Math Formula Images To LaTeX Code Based On End-to-End Approach With Attention Mechanism

by

Trung Hoang Quoc, Bao Thai Duy

THE FPT UNIVERSITY HO CHI MINH CITY

Math Formula Images To LaTeX Code Based On End-to-End Approach With Attention Mechanism

by

Trung Hoang Quoc, Bao Thai Duy
Supervisor: Trung Nguyen Quoc, Tien Nguyen Quoc

A final year capstone project submitted in partial fulfillment of the requirement for the Degree of Bachelor of Artificial Intelligent in Computer Science

DEPARTMENT OF ITS
THE FPT UNIVERSITY HO CHI MINH CITY
April 2024

Acknowledgements

In the acknowledgment section, we express our gratitude to the individuals who have contributed their support, guidance, and encouragement to this project.

- Our esteemed supervisors, Trung Nguyen Quoc and Tien Nguyen Quoc, for their invaluable guidance, unwavering assistance, and meticulous review throughout the entirety of this research endeavor.

We are grateful to all those who have helped us in ways both big and small, and without whom this project would not have been possible. Thank you all.

Author Contributions

- **Conceptualization:** Trung Hoang Quoc.
- **Methodology & Software:** Trung Hoang Quoc.
- **Data Curation:** Trung Hoang Quoc, Bao Thai Duy.
- **Writing - Original Draft Preparation:** Bao Thai Duy.
- **Writing - Review and Editing:** Trung Hoang Quoc.
- **Supervision:** Trung Nguyen Quoc, Tien Nguyen Quoc.
- **Project Administration:** Trung Hoang Quoc (Lead), Bao Thai Duy.
- **Tools & Resources:** Trung Hoang Quoc.
- **Funding Acquisition:** Bao Thai Duy.

All authors have read and agreed to the Final Capstone Project document.

Abstract

Recognizing mathematical formulas in images and translating them into LaTeX sequences, both printed and handwritten, is challenging due to the complexity of two-dimensional formulas and lack of training data. Traditional methods can only handle simple formulas and are not effective for complex formulas. In this paper, we introduce the Sumen (**S**caling **U**p I**m**age-to-LaT**e**X Performa**n**ce) model, an encoder-decoder architecture based on Transformer with attention mechanism trained on the largest dataset from previous works. The model achieves a BLEU score of 95.59, Edit Distance (ED) of 97.3, and Exact Match (EM) of 69.23 on the img2latex100k benchmark. On the CROHME 2014/2016/2019 benchmark, the corresponding results on Expression Recognition Rates (ExpRate) are 58.01/82.39/78.99 and Word Error Rate (WER) are 9.46/2.55/4.51. All of our metrics outperform state-of-the-art methods on both printed and handwritten formulas.

Keywords: Image to LaTeX, printed formula recognition, handwritten formula recognition, CROHME, Img2latex-100k

Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview

1.1.1 Context

Mathematical formulas are essential for life and are widely used in various fields. They are used to describe algorithms, illustrate ideas, clarify complex concepts, and create clear and consistent content. To express mathematical formulas in documents, we need to use markup languages. LaTeX is a language used for typesetting and formatting mathematical formulas professionally. Its application is spreading from science to technology, from papers to websites. In the scientific field, LaTeX is widely used to write articles, scientific research reports, blogs, and books. It can show the intuition of formulas and is a powerful tool to represent them clearly, accurately, and easily readable. In addition, in the field of education, LaTeX is often used to write mathematical formulas in lectures, teaching materials, and tests. Mathematical formulas can be used to explain complex concepts more clearly and attractively to students.

Representing mathematical formulas with LaTeX is a real challenge, especially for non-professionals. Even for experts, typing long and complex mathematical formulas is not easy. Mathematical expressions that are too long or too complex can make users easily make some minor mistakes, which can lead to time loss and inconvenience in editing. And these errors happen frequently. If a formula is too long, repairing or adjusting it can be very complicated, we may need to scan each part of the formula to be able to fix the error, which can make us confused and impatient when we have to fix each character. And for those who are new to LaTeX, typing each complex code can be a big challenge, requiring a lot of time and patience.

1.1.2 Math Formula Images To Latex

To help people apply LaTeX to write mathematical formulas easily and quickly, we have built a model that supports converting mathematical formula images to corresponding LaTeX sequences. This way, users can easily convert handwritten mathematical formulas on paper and convert them into LaTeX format, saving time and effort compared to manual typing. In addition, our model can also be used to support users in practicing how to use LaTeX, helping users to easily understand the structure of writing LaTeX code and

learn to use it faster. This is especially useful for beginners with LaTeX, as they can practice writing mathematical formulas and see the results immediately through our model. It helps users learn effectively in learning LaTeX, thereby promoting the learning process and approaching this language flexibly and efficiently.

Converting math formula images to LaTeX, also known as img2latex, is the process of translating mathematical formulas in images into LaTeX sequences. It was created to help scientists, teachers, and non-professionals easily convert mathematical formulas from images into LaTeX code and display them in their documents. Formula recognition has two problems: recognizing images containing handwritten mathematical expressions or printed mathematical expressions. Handwritten formula recognition is more difficult than printed formula recognition due to many reasons such as lack of large enough data, confusion and lack of clarity due to bad handwriting.

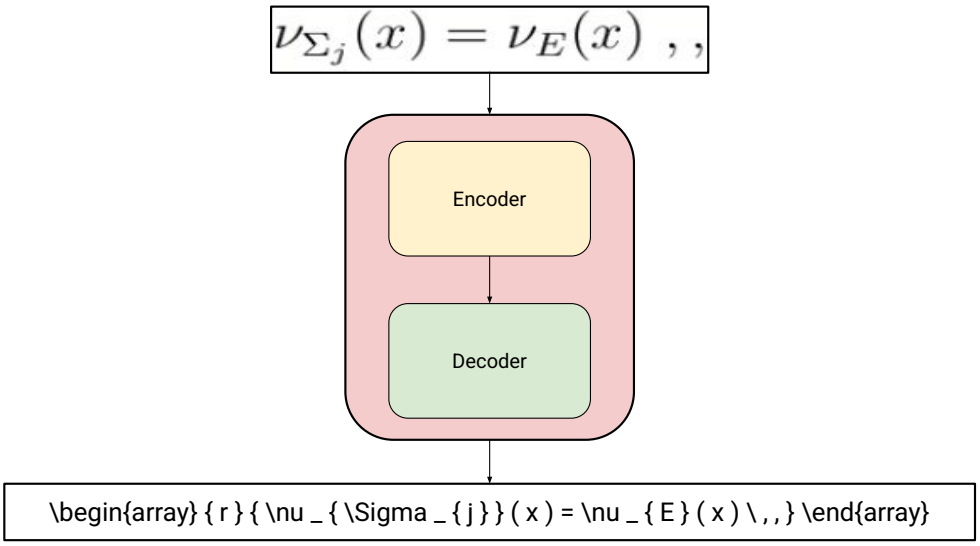


Figure 1: Image-to-latex workflow

An end-to-end approach means that the model receives image input and directly generates a LaTeX sequences without going through the intermediate steps illustrated in Figure 1. Specifically, the model takes raw image input without the need for an OCR engine, bypassing the need for an OCR engine to help prevent images from being segmented.

1.2 Background

Previously, traditional methods such as the INFTY system [34] created rules based on structural analysis methods and syntactic parsers. These methods could solve simple mathematical formulas such as superscript, subscript, special symbols, and fractions. However, they could not solve complex and long formulas because they only recognized characters and did not understand the relationship between characters and multi-line formulas.

In recent years, the advent of deep neural networks has led to the proposal of methods based on the encoder-decoder architecture. These methods are widely used and have achieved good results in many areas such as image captioning, machine translation, text to image, and speech to text. The encoder-decoder architecture combines computer vision and natural language processing. The encoder is responsible for processing the image and capturing important information features. The decoder then receives these feature information and generates the corresponding LaTeX string for the formulas in the image. The encoder-decoder architecture simplifies in the process and has achieved many successes in this field. The model receives the input and directly generates the output without the need to complicate the data, such as segmenting the location of text segments in the image.

Although the encoder-decoder architecture has been working well in recent years, previous methods have only focused on the Printed Mathematical Expression Recognition (PMER) [13, 14, 32, 38, 39, 48] or Handwritten Mathematical Expression Recognition (HMER) [46, 9, 47, 20, 11, 42, 44]. However, they have not been very good at both tasks in the same method.

1.3 Specific Works

We use the encoder-decoder architecture and continue to improve and develop it. We propose a large dataset of about 3.4 million image-text pairs to achieve good generalization. We train both handwritten mathematical expressions and printed mathematical expressions data at the same time, which helps the model share LaTeX representation knowledge. We consider handwritten mathematical expressions as a data augmentation. Our model can support users in converting images to LaTeX for both printed and handwritten formulas, and it achieves good results on various benchmarks. The primary contributions are:

- Introducing a large-scale dataset for mathematical formula recognition.

- Proposing the Sumen model, which achieves state-of-the-art results compared to existing methods on both PMER and HMER tasks.
- Releasing the source code and checkpoints of the Sumen model for the research community to use and further develop.

2 Related Works

The advent of deep neural networks has replaced classical methods with encoder-decoder architectures and brought about good results and real success in the field of Computer Vision & Natural Language Processing. The encoder uses models such as Convolutional Neural Networks (CNN) and Vision Transformer (ViT) [16]. CNN is one of the most popular models used in image processing, with the ability to automatically extract features from image inputs through convolution and pooling layers, CNN has proven its superiority in object recognition and image classification. Some popular CNNs today include EfficientNet [35], which is designed to optimize computational efficiency and accuracy by increasing the model size in depth, width, and resolution. DenseNet [18] is a densely connected CNN architecture where each layer is connected to every other layer in a dense manner. This enhances information flow and addresses the vanishing gradient problem. Meanwhile, ViT is a neural network architecture that uses Transformer, originally developed for natural language processing, but has been applied to the field of computer vision. In ViT network, images are divided into patches and represented as vectors, similar to how Transformer represents words in NLP. Each vector represents a part of the image and is fed into the Transformer network to extract information and perform tasks such as classification or object detection. ViT has demonstrated the power of Transformer in computer vision, which is no less than that of Transformer in natural language processing.

Decoder based on Recurrent Neural Network (RNN) has the disadvantages of vanishing gradient and exploding gradient, making it challenging to work with long sequences. It is also slow to compute and can only learn short-term memory [29]. For complex mathematical formula markup chains that can exceed hundreds of LaTeX tokens, the hidden state vector in RNN is not enough to compress all the information from the encoder. Long Short Term Memory (LSTM) was introduced to solve the problems of RNN and to scale up to capture long-term memory when needed. It mitigates the vanishing gradient problem and can forget irrelevant information through the forget gate. However, LSTM is complex, requires significant time and computational resources, and cannot be parallelized. Transformer [37] was introduced based on the self-attention mechanism and has gradually replaced methods like RNN and LSTM due to its convenient parallel processing and ability to overcome the limitations mentioned above. It is widely used with Large Language

Models (LLMs) and has achieved state-of-the-art results.

Methods using encoder-decoder architecture such as [14] propose neural encoder-decode with coarse-to-fine attention mechanism, the author uses CNN encoder to extract feature in the image and RNN decoder implements a conditional language model over the vocabulary. [32] introduces a neural transducer model with visual attention, which uses CNN as encoder and RNN as decoder combined with using beam search in the inference process. [39] proposed a method including CNN combined with positional encoding used in encoder to extract features, features will be augmented with 2D positional encoding before being unfolded into a vector and fed into the LSTM decoder to translate into a sequence of LaTeX tokens. [48] has proposed a model applying Transformer-based encoder-decoder architecture, in which the encoder uses ViT and takes the idea of machine translation applied to img2latex task, in addition this method combines using the YOLO model [31] for the preprocessing step to separate single-line formulas from multi-line formulas to improve the model's accuracy.

Autoregressive-decoder methods like [44, 43] idea of learning and generating text strings in left-to-right direction. Since then, the BTTR [47] and ABM [9] methods were born and introduced the novel bidirectional training strategy with the purpose of being able to learn LaTeX sequences in left-to-right and right-to-left directions on the Transformer decoder, helping the model learn well and achieve higher accuracy. But entails parameters and longer training time. Inspired by the coverage mechanism in RNN, CoMER [46] proposes a model to improve the Transformer's shortcomings in terms of sufferings from the lack of coverage problem, using Attention Refinement Module (ARM) to fine-tune attention weights with past alignment information without hurting its parallelism and performs better than vanilla transformer decoder and RNN decoder in the HMER task.

3 Project Management Plan

In the 14 weeks of project implementation, our team has completed our project through activities listed:

- In the first two weeks, we brainstormed ideas and then searched for relevant literature, papers, and the most suitable tools to complete the project. We researched and explored methods appropriate for our current conditions.
- From weeks 3 to 4, we researched and searched for datasets suitable for the model we wanted to build. We found datasets for handwritten mathematical expressions and printed mathematical expressions. These datasets were then normalized by us to a common standard, preparing them for our training step.
- In week 5, our team delved deeper into the models, architectures, and identified the most suitable training methods.
- From week 6 to 9, we focused on training and fine-tuning certain parameters to improve the results. Eventually, our model was completed and yielded out performance compared to the papers we research.
- During weeks 9 to 11, we evaluate the results and write the final report, and at the same time we write paper for the ICIT scientific conference in Da Nang, September 2024.
- During the remaining weeks, we evaluated the results, completed the report, and developed a demo website for better visualization.

Task Name	Priority	Owner	Start Date	End Date	Status
Find documents	High	Trung Hoang Quoc Bao Thai Duy	01/01/2024	07/01/2024	Completed
Review papers	High	Trung Hoang Quoc Bao Thai Duy	08/01/2024	14/01/2024	Completed
Find dataset	High	Bao Thai Duy	15/01/2024	18/01/2024	Completed
Collect and process data	High	Trung Hoang Quoc Bao Thai Duy	19/01/2024	28/01/2024	Completed
Research on models and architectures	High	Trung Hoang Quoc	29/01/2024	04/02/2024	Completed
Find different approaches and choose the most suitable one	High	Bao Thai Duy	29/01/2024	04/02/2024	Completed
Experiment	High	Trung Hoang Quoc Bao Thai Duy	05/02/2024	17/03/2024	Completed
Compare results	Low	Trung Hoang Quoc	11/03/2024	17/03/2024	Completed
Writing paper	High	Trung Hoang Quoc	11/03/2024	31/03/2024	Completed
Writing report	Medium	Trung Hoang Quoc Bao Thai Duy	18/03/2024	31/03/2024	Completed
Review report revision	Medium	Bao Thai Duy	01/04/2024	07/04/2024	Completed
Create website demo	Medium	Trung Hoang Quoc	08/04/2024	14/04/2024	Completed
Future work	Low	Trung Hoang Quoc Bao Thai Duy	29/04/2024	...	Pending

Table 1: Project plan

4 Materials and Methods

4.1 Materials

4.1.1 Project Management Tool

Here are the tools we use for project management:

- **Notion** is an online application developed by Notion Labs Inc., renowned for its freemium business model. It is a versatile application that supports note-taking, project management, planning, and collaboration. With Notion, users can create notes in various formats, manage projects using Kanban boards, plan personal tasks, work, and study with calendars and to-do lists, and collaborate with team members in real-time. Notion is a flexible and user-friendly tool, suitable for various purposes.

- **Google Docs** is a free online word processing application provided by Google. With Google Docs, we can create, edit, and share documents with team members in real-time. Google Docs provides basic tools for text editing, allowing you to easily format fonts, font sizes, colors, etc. It also integrates with other Google apps such as Drive, Gmail, Calendar, etc., helping you work more efficiently.

- **Google Sheets** is a free online spreadsheet application provided by Google. With Google Sheets, you can create spreadsheets with various types of data, easily edit data, share spreadsheets with others in real-time, use formulas to calculate and analyze data, and create visual charts to illustrate data. Google Sheets is a powerful tool that helps you manage and analyze data efficiently.

- **Overleaf** is an online text editing platform based on LaTeX, widely used in academic and research communities. With Overleaf, users can easily and effectively create scientific documents, technical reports, theses, and other text documents without needing to install or configure LaTeX on their personal computers. The intuitive interface of Overleaf allows users to create and edit documents conveniently. This platform integrates many useful features such as syntax error checking, version control, and easy document sharing with colleagues. This helps enhance collaboration and speed up work in group projects.

4.1.2 Hardware

Here are the hardware we used throughout this project:

- **Kaggle** is an online platform that provides a community for data scientists to collaborate, share code, and compete in AI competitions. It offers powerful computational resources for free, including GPU (2 NVIDIA Tesla T4 GPUs; 1 NVIDIA Tesla P100 GPU), TPU, CPU, and 29GB of RAM. However, there is a weekly usage limit of 30 hours, which resets on Saturday morning. Kaggle also offers unlimited free storage for public data and 107GB of storage for private data. To avoid the usage limit, we created multiple Kaggle accounts for the training process.

- **Personal Computer:** We use personal laptops with 10th generation Intel core i7 CPU and 8GB RAM for model inference and demo.

4.2 Dataset

In the formula recognition task, we divide it into two domains: Handwritten and printed mathematical formulas. We use different datasets to train and evaluate the model based on these two domains. In it, we collect and build the largest dataset to date from online sources, creating a robust and well-generalizable dataset. This dataset consists of approximately 3.4 million image-text pairs, including both handwritten mathematical expressions (200,330 samples) and printed mathematical expressions (3,237,250 samples). The percentage of data is illustrated in Figure 2 and data details are shown in Figure 3. We publish the dataset at [17].

Printed mathematical expressions: We collect from Im2latex-100k dataset [13], I2L-140K Normalized dataset and Im2latex-90k Normalized dataset [32], Im2latex-170k dataset [2], Im2latex-230k dataset [3], latex-formulas dataset [8] and Im2latex dataset [4].

Handwritten mathematical expressions: We collected data from the Competition on Recognition of Online Handwritten Mathematical Expressions (CROHME) dataset [26, 27, 25], Aida Calculus Math Handwriting Recognition Dataset [1] and Handwritten Mathematical Expression Convert LaTeX [6]

Pre-processing: Due to the large dataset and the fact that the same mathematical formula can be represented in different LaTeX string formats in an image, it is easy to cause polymorphic ambiguity. To address this issue,

we use the normalization method with KaTeX parser [7]. We convert the raw LaTeX strings into an abstract syntax tree, and then apply safe normalizing tree transformation to eliminate ambiguity in the LaTeX markup strings. Some previous works using this method include [13, 39].

Items	**Amount of data (train)**
Im2latex-100k dataset [13]	93,925 image-text pairs
I2L-140K Normalized dataset and Im2latex-90k Normalized dataset [32]	132,500 image-text pairs
Im2latex-170k dataset [2]	165,477 image-text pairs
Im2latex-230k dataset [3]	234,311 image-text pairs
latex-formulas dataset [8]	1,035,945 image-text pairs
Im2latex dataset [4]	1,586,584 image-text pairs
CROHME [26, 27, 25]	10,968 image-text pairs
Handwritten Mathematical Expression Convert LaTeX [6]	11,181 image-text pairs
Aida Calculus Math Handwriting Recognition Dataset [1]	100,000 image-text pairs

Table 2: Source of dataset

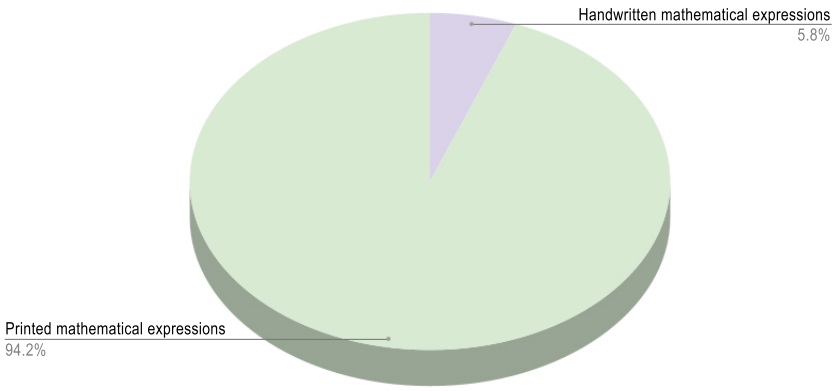


Figure 2: Overall dataset

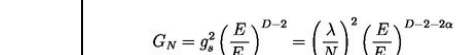
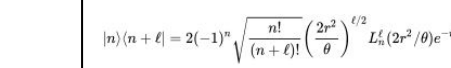
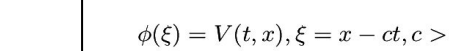
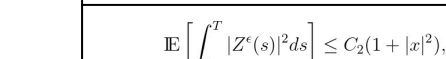
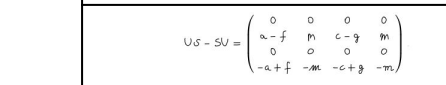
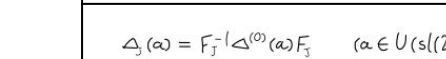
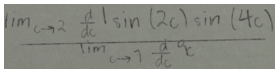
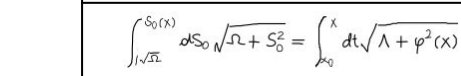
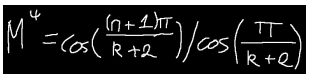
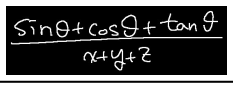
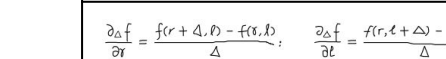
Images	Labels
	y = \left(\begin{array} { c c c c } { { 0 } } & { { q _ { 1 } } } & { { 0 } } & { { 0 } } \\ { { 0 } } & { { 0 } } & { { q _ { 2 } } } & { { 0 } } \\ { { 0 } } & { { 0 } } & { { 0 } } & { { q _ { 3 } } } \\ { { q _ { 4 } } } & { { 0 } } & { { 0 } } & { { 0 } } \end{array} \right)
	*G _ { N } = g _ { s } ^ { 2 } \left(\frac { E } { E _ { s } } \right) ^ { D - 2 } = \left(\frac { \lambda } { N } \right) ^ { 2 } \left(\frac { E } { E _ { s } } \right) ^ { D - 2 - 2 \alpha }*
	*| n \rangle \langle n + \ell | = 2 (- 1) ^ { n } \sqrt { \frac { n ! } { (n + \ell) ! } } \left(\frac { 2 r ^ { 2 } } { \theta } \right) ^ { \ell / 2 } L _ { n } ^ { \ell } (2 r ^ { 2 } / \theta) e ^ { - r ^ { 2 } / \theta } e ^ { i \ell \varphi } | ,*
	*\begin{array} { r } { \phi (\xi) = V (t , x) , \xi = x - c t , c > 0 } \end{array}*
	*\displaystyle { \mathrm { I \! E } } \left[\int _ { 0 } ^ { T } | Z ^ { \epsilon } (s) | ^ { 2 } d s \right] \leq C _ { 2 } (1 + | x | ^ { 2 }) ,*
	*U S - S U = \left(\begin{array} { c c c c } { 0 } & { 0 } & { 0 } & { 0 } \\ { a - f } & { m } & { c - g } & { m } \\ { 0 } & { 0 } & { 0 } & { 0 } \\ { - a + f } & { - m } & { - c + g } & { - m } \end{array} \right) .*
	*\Delta _ { - { \! J } } (a) = F _ { - { J } } ^ { - 1 } \Delta ^ { (0) } (a) F _ { - { \! J } } ^ { } \qquad (a \in U (s l (2))) | .*
	*\frac { \operatorname* { l i m } _ { c \to 2 } \frac { d } { d c } 1 \sin { \left(2 c \right) } \sin { \left(4 c \right) } } { \operatorname* { l i m } _ { c \to 7 } \frac { d } { d c } 9 c }*
	*\int _ { i \sqrt \Omega } ^ { S _ { 0 } (x) } d S _ { 0 } \sqrt { \Omega + S _ { 0 } ^ { 2 } } = \int _ { x _ { 0 } } ^ { x } { d t \sqrt { \Lambda + \varphi ^ { 2 } (x) } }*
	*M ^ { 4 } = \cos (\frac { (n + 1) \pi } { R + 2 }) / \cos (\frac { \pi } { R + e })*
	*\frac { \sin \theta + \cos \theta + \tan \theta } { x + y + z }*
	*\frac { \partial _ { \Delta } f } { \partial r } = \frac { f (r + \Delta , \ell) - f (r , \ell) } { \Delta } ; \qquad \frac { \partial _ { \Delta } f } { \partial \ell } = \frac { f (r , \ell + \Delta) - f (r , \ell) } { \Delta } | .*

Figure 3: Data examples

Data Format: We collected the dataset from many sources, which resulted in various formats. Therefore, we needed to process and store them in a common format. The dataset consists of four folders: train, test, val, and root, as shown in Figure 4. The root folder contains images in gray, png, jpg, and bmp formats. For the train, test, and val folders, we store the image paths and corresponding LaTeX codes in CSV (Comma-Separated Values) files with two columns. CSV is widely used in AI/ML and Data Science. It is stored in a simple tabular format, which is flexible and widely supported. In the train and val folders, there are files such as handwritten mathematical expressions, printed mathematical expressions and total for both datasets.

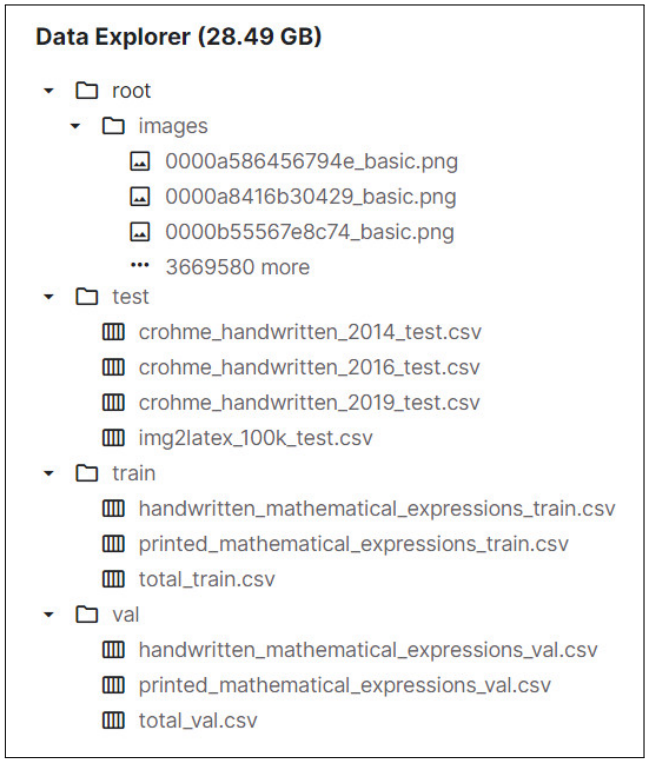


Figure 4: Data format

4.3 Methods

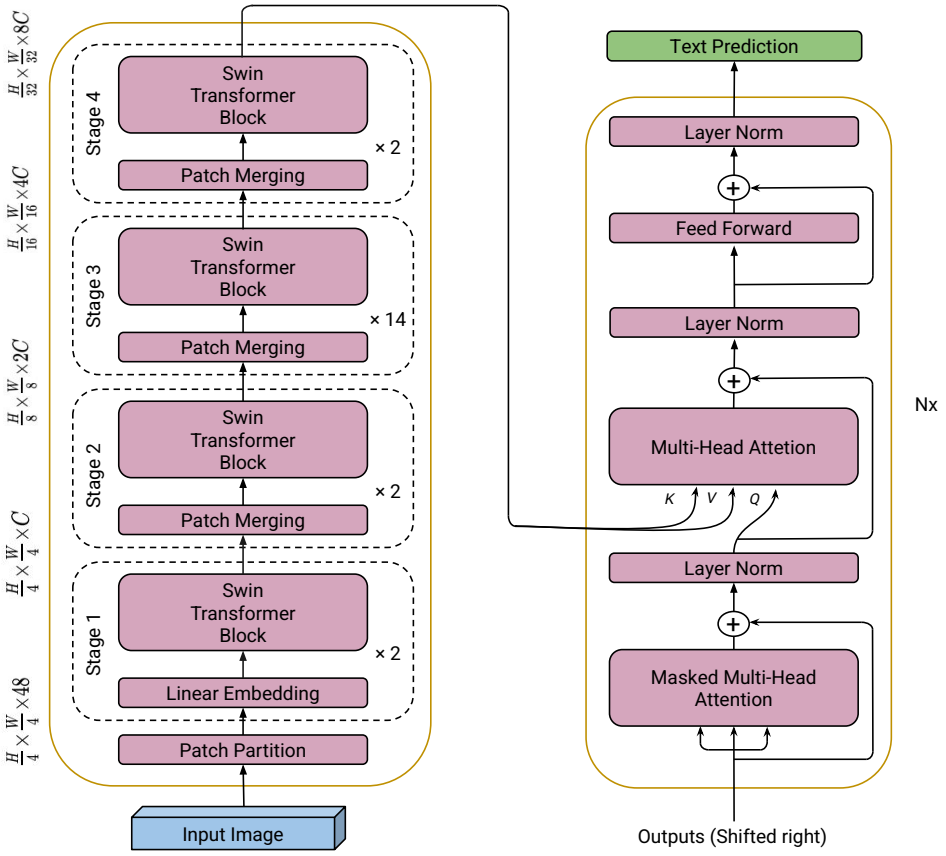


Figure 5: Overview of Sumen architecture (left: encoder, right: decoder)

The architecture we use in this project is Nougat [10], which is a transformer-based encoder-decoder architecture. Nougat is a Visual Transformer model that performs an Optical Character Recognition (OCR) task to understand scientific documents and translate them into markup language. The encoder is Swin Transformer [24] and and the decoder is decoder-Transformer [37]. The entire architecture of the model is shown in Figure 5. We initialize the weights from Nougat.

4.3.1 Encoder

The advent of Swin Transformer is a breakthrough in the field of computer vision, combining the power of the attention mechanism and the hierarchical architecture of CNN. In which width and height are reduced and channels are increased in the later layers, providing flexibility to scale different image sizes. Some notable features are shown in Figure 6 as follows: Window-based self-attention is used to attend to patches or tokens in a window (local attention). However, sharing information across different windows is crucial to understanding the relationships between objects in the image. Therefore, shifted windows allow windows to communicate, which is an idea based on stride in CNN but with a different variation called cyclic shift. The model takes an input of an image with size $H \times W \times 3$, which is then used to extract important information from the image and output the encoder's KV feature vectors.

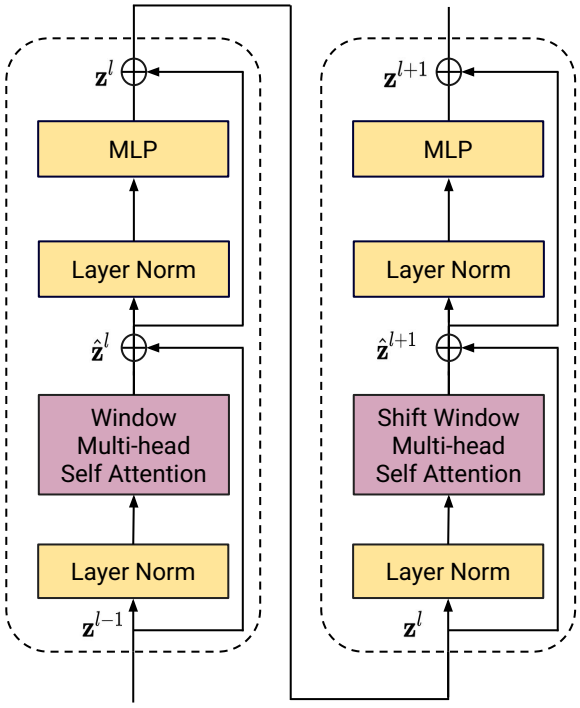


Figure 6: Swin transformer block

4.3.2 Decoder

The decoder uses decoder-Transformer, which is an auto-regressive language modeling (unidirectional), meaning that the model reads words only in the left-to-right direction. The decoder receives important information in the image represented by feature vectors KV from the encoder through cross multi-head attention and generates LaTeX sequences corresponding to the formulas in the image.

4.3.3 Attention Mechanism

Introduced to improve the performance of encoder-decoder models, the Attention mechanism is an indispensable part of modern models today, achieving high accuracy in most NLP tasks. The Attention mechanism was proposed in [37] with a parallel processing mechanism that works by comparing words with each other pairwise to find the importance of the words it attends to, including itself (self).

Self-Attention:

- To compute self-attention, three vectors Query, Key, Value (Q, K, V) are created by three neural networks with different weights W.

$$Q = W_q * x + b$$
$$K = W_k * x + b$$
$$V = W_v * x + b$$

- The scaled dot-product attention function, illustrated in the Figure 7, will perform the attention computation. During training, the vectors $W_q, W_k,$ and W_v will be learned to be adjusted for the best attention.

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

Multi-Head Attention:

- Each Self-Attention module is called a Head.

$$Head_i = Attention_i(x) = softmax\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right)V_i$$

- Using multiple heads simultaneously is called Multi-Head Attention. It is a collection of multiple self-attention layers, where each head performs a different function. These heads operate in parallel and produce the final result together at the same time. This helps to learn different aspects of the relationship of a word with the other words in the sentence and improves performance and converges faster during training.

$$MultiHead(Q, K, V) = Concat(Head_1, Head_2, \ldots, Head_h)W^0$$

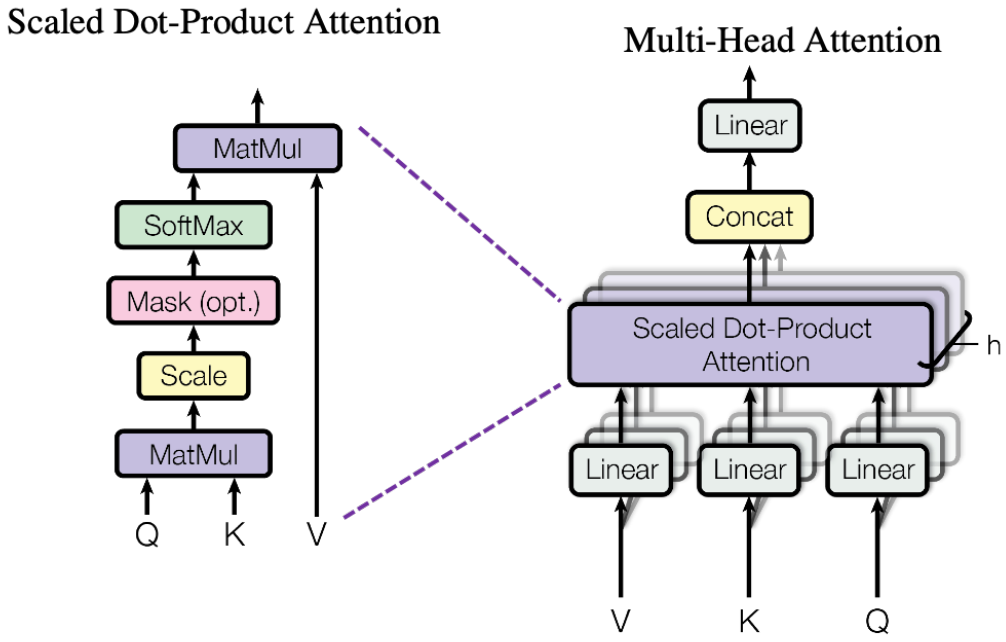


Figure 7: Scaled Dot-Product Attention (left) and Multi-Head Attention (right) [37]

4.3.4 Data Augmentation

We use image augmentation methods similar to Nougat during training to apply random changes including RGB shift, bitmap, erosion, dilation, shift scale rotation, grid distortion, affine, elastic transform, random brightness

contrast, image compression, gauss noise, gaussian blur. This increases the amount of variation in a more diverse dataset and improves the generalization ability of models, an example is illustrated in Figure 8.

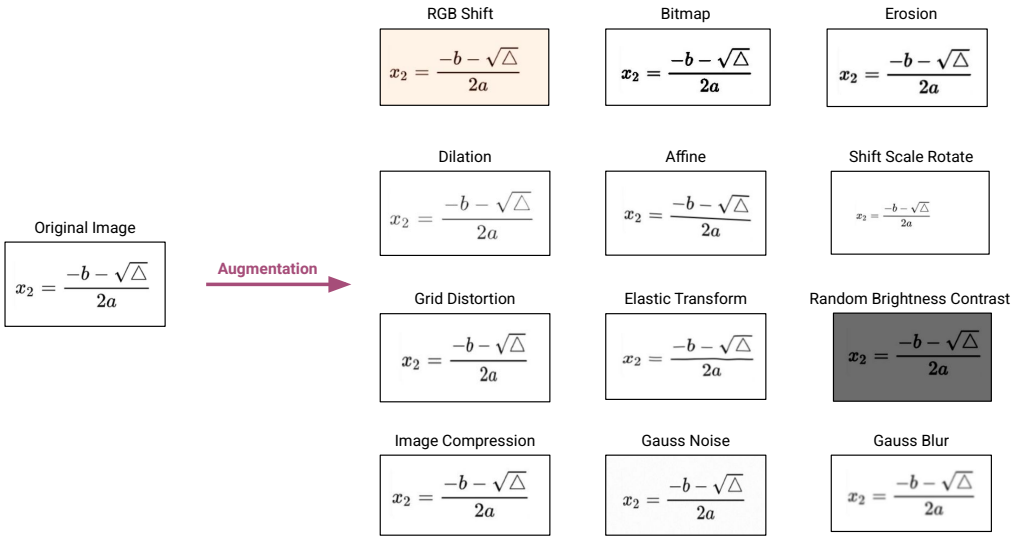


Figure 8: Some examples of transformations of the original image that will create a new training sample.

4.3.5 Beam Search Algorithm

Beam Search is a popular search algorithm used in natural language processing to generate text. The algorithm works by maintaining a finite set k of strings representing the best sequences the algorithm has found so far. From these strings, Beam Search generates the next token with the highest probability that fits the initial string found. Beam Search works as follows:

1. Initialize k: representing the number of strings with the highest probability.

2. Expand the strings:

    - Take each string in the k strings.

- Find the k next strings with the highest probability to concatenate, creating k new strings.

3. Keep the best k strings: Sort the $k * k$ candidate strings by probability, taking the k strings with the highest probability.

4. Repeat until termination: Repeat steps 2 and 3 until an end-of-string token is encountered or a maximum length is reached.

5. Return the result: Return the string with the highest probability among all strings in the last step.

4.4 Implementation Details

The entire model is used using Pytorch & Transformer framework, during training we use Lion optimizer [12] with weight decay = 1e-2, $\beta_1 = 0.95$ and $\beta_2 = 0.98$. We use a max length of 512, image resolution of 224×768 (height $\times$ width), learning rate of 1e-4 with 1k warmup steps based on cosine schedule. The entire model was trained in 10 epochs with batch size of 2048 with gradient accumulation on 1 Tesla P100-PCIE-16GB GPU.

5 Results

5.1 Metrics

The following metrics are used for the test dataset:

- **Bilingual Evaluation Understudy (BLEU):** Compares the similarity between the machine translation and the reference translation. BLEU uses n-grams (combinations of n words) to measure the level of overlap between text strings [28].
- **Edit Distance (ED)**: Or Levenshtein distance [22], measures the minimum number of edit operations required to transform one text string into another. Edit operations include insertion, deletion, and substitution of characters. The result is calculated by dividing the total number of operations performed by the total number of words in the reference. To compare with other methods, we take 1− minimum ED.
- **Exact Match (EM)**: EM checks whether the machine translation or model outputs match the references exactly (100%). If it matches the references, the result is calculated by dividing the total number of correct outputs by the total number of references.
- **Word Error Rate (WER)**: WER uses edit distance to calculate the accuracy percentage. Instead of using the distance between phonemes, it uses the distance between words. The final result is calculated by dividing the total number of substitutions + deletions + insertions by the total number of words in the reference.
- **Expression Recognition Rates (ExpRate):** ExpRate uses edit distance to calculate the accuracy of the model. For each line with distance = 0, the total number of correct lines is increased by 1. The final result is the number of correct lines divided by the total number of lines.

5.2 Printed Mathematical Expressions Benchmark

We use the Im2latex-100k dataset for our experiments on PMER task, which is a collection of about 100,000 real-world mathematical expressions rendered from public papers on the arxiv.org server. This is a popular dataset dedicated

to the printed formula recognition task, and the test set consists of 10,285 samples. We use metrics such as BLEU score (4-gram), EM score, and ED score to evaluate on this dataset. The results shown in the Table 3 indicate that our model performs very well overall in terms of the entire string sequence on BLEU and ED, with the highest result, while EM is only at an average level and not the highest. This is affected by one or more characters being incorrect, as we observed, leading to zero results on an entire string.

Model	BLEU	Edit Distance	Exact Match
INFTY [34]	66.65	96.4	59.6
CNNENC [13]	75.01	61.17	53.53
WYGIWYS [13]	87.73	87.60	77.46
MI2LS w/o Reinforce [39]	89.08	91.09	79.39
MI2LS with Reinforce [39]	90.28	92.28	82.33
DoubleAttention [39]	88.42	88.57	79.81
DenseNet [38]	88.25	91.57	-
MathBERT [30]	90.45	90.11	**87.52**
Zhou *et al.* [48]	92.11	90.0	60.2
Wang *et al.* [38]	85.71	90.25	28.68
DenseNet(2 blocks)	85.82	91.38	35.68
C-S attention	86.54	90.75	31.79
DenseNet + C-S	88.25	91.57	37.09
Sumen-base (ours)	**95.59**	**97.3**	69.23

Table 3: Comparison of performance printed formula recognition task with previous methods on Im2latex-100k test set.

5.3 Handwritten Mathematical Expressions Benchmark

We use the CROHME dataset to demonstrate the effectiveness of our model on the HMER task, which is a large open dataset for handwritten mathematical expressions. The test set consists of three versions: CROHME 2014 (986 samples), CROHME 2016 (1,147 samples), CROHME 2019 (1,199 samples). We choose expression level metrics: ExpRate (%), ExpRate ≤ 1 error (%), ExpRate ≤ 2 error (%), and ExpRate ≤ 3 error (%) provided by the CROHME 2019 organizers [25]. Furthermore, we use the WER metric to evaluate errors at the word level. The results shown in the Table 5 & 4 and indicate that

we achieve the best results on 4/4 ExpRate metrics on CROHME 16&19. However, on CROHME 14 we only achieve the best result on 1/4 ExpRate metrics. Overall, our model outperforms other models.

Dataset	Model	ExpRate	WER
CROHME 14	Dense [44]	50.1	13.9
	Dense+MSA [44]	52.8	12.9
	ABM [44]	56.85	10.1
	Sumen-base (ours)	**58.01**	**9.46**
CROHME 16	Dense [44]	47.5	15.4
	Dense+MSA [44]	50.1	13.7
	Sumen-base (ours)	**82.39**	**2.55**
CROHME 19	Sumen-base (ours)	**78.99**	**4.51**

Table 4: Comparison of performance handwritten formula recognition task with previous methods on CROHME 2014/2016/2019 test sets based on word error rate.

Dataset	Model	ExpRate	$\leq$ 1 error	$\leq$ 2 error	$\leq$ 3 error
CROHME 14	DenseWAP [44]	43.0	57.8	61.9	-
	DenseWAP-TD [45]	49.1	64.2	67.8	-
	WS-WAP [36]	53.65	-	-	-
	Li *et al.* [23]	56.59	69.07	75.25	78.60
	Ding *et al.* [15]	58.72	-	-	-
	BTTR [47]	53.96	66.02	70.28	-
	BTTR (CoMER) [46]	55.17	67.85	72.11	74.14
	CoMER [46]	**59.33**	71.70	75.66	77.89
	PAL [40]	39.66	56.80	68.51	-
	WAP [43]	46.55	61.16	65.21	66.13
	PGS [21]	48.78	66.13	73.94	-
	PGS-v2 [41]	48.88	64.50	69.78	-
	DLA [19]	49.85	-	-	-
	ABM [9]	56.85	**73.73**	**81.24**	-
	WYGIWYS [13]	36.4	-	-	-
	Sumen-base (ours)	58.01	72.11	80.22	**85.04**
CROHME 16	DenseWAP [44]	40.1	54.3	57.8	-
	DenseWAP-TD [45]	48.5	62.3	65.3	-
	WS-WAP [36]	51.96	64.34	70.10	72.97
	Li *et al.* [23]	54.58	69.31	73.76	76.02
	Ding *et al.* [15]	57.72	70.01	76.37	78.90
	BTTR [47]	52.31	63.90	68.61	-
	BTTR (CoMER) [46]	56.58	68.88	74.19	76.90
	CoMER [46]	59.81	74.37	80.30	82.56
	WAP [43]	44.55	57.10	61.55	62.34
	PGS [21]	36.27	-	-	-
	PGS-v2 [41]	49.61	64.08	70.27	-
	DLA [19]	47.34	-	-	-
	ABM [9]	52.92	69.66	78.73	-
	Sumen-base (ours)	**82.39**	**89.97**	**94.42**	**95.99**
CROHME 19	DenseWAP [44]	41.7	55.5	59.3	-
	DenseWAP-TD [45]	51.4	66.1	69.1	-
	Ding *et al.* [15]	61.38	75.15	80.23	82.65
	BTTR [47]	52.96	65.97	69.14	-
	BTTR (CoMER) [46]	59.55	72.23	76.06	78.40
	CoMER [46]	62.97	77.40	81.40	83.07
	ABM [9]	53.96	71.06	78.65	-
	Sumen-base (ours)	**78.99**	**86.22**	**90.5**	**92.07**

Table 5: Comparison of performance handwritten formula recognition task with previous methods on CROHME 2014/2016/2019 test sets based on expression recognition rates score.

6 Discussion

During the implementation of this project, with a large amount of data, we were limited in time and computing resources on Kaggle. Our model only trained for 10 epochs, with the best results achieved on many metrics, we believe that continued training of the model on our dataset can improve the accuracy further and Sumen promises will be a useful model for applications related to mathematical formula recognition.

7 Conclusions And Perspectives

We have proposed a transformer-based encoder-decoder architecture to convert images containing mathematical formulas into LaTeX code. We also introduce a new large dataset that helps scale up the accuracy of img2latex performance. Compared with other models, we have succeeded in solving the major challenge of recognizing mathematical formulas, including printed and handwritten formulas in the same model, the results our model achieves state-of-the-art performance on the CROHME 2016/2019 and Im2latex-100k test sets .

We have succeeded in solving the major challenge of recognizing mathematical formulas, including both printed and handwritten formulas in the same model. Previous studies have had difficulty handling complex formulas and achieving state-of-the-art results on only handwritten mathematical expressions or only printed mathematical expressions. Our new method has defeated previous methods by using the Sumen architecture with improvements in the Vision module and a large dataset that contributes to improving the generalization ability of the model. This result is a clear demonstration of the excellent performance of the Sumen model compared to previous methods. In particular, the BLEU and ED indices on the img2latex100k benchmark have demonstrated the model's diverse and accurate capabilities in identifying and converting images into LaTeX format. At the same time, the WER and Exprate indexes on the CROHME benchmark are the highest, demonstrating Sumen's flexibility and efficiency for many types of mathematical formulas. On the contrary, for the EM metric, it is not high but BLEU and ED are the highest. The reason for this is that EM is used to evaluate individual characters, while BLEU and ED are used to evaluate the overall string. string, the wrong number of characters in the code will lead to bad results of the metric.

Future work will concentrate on creating a large handwritten mathematical expressions dataset to balance the current dataset using GAN as proposed in [33] to convert printed mathematical expression images into handwritten mathematical expression images.

A Appendix

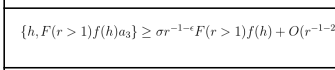
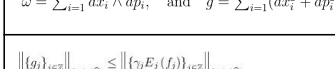
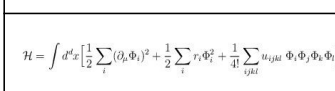
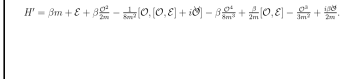
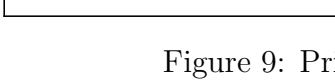
Images	Predicts	Labels
	= \int d x \left(\pi \left(x , t \right) \frac \delta { \delta \phi \left(x , t \right) } - \left(\left(m ^ { 2 } - \nabla _ { x } ^ { 2 } \right) \phi \left(x , t \right) \right) \, \frac \delta { \delta \pi \left(x , t \right) } \right)	= \int d x \left(\pi \left(x , t \right) \frac { \delta } { \delta \phi \left(x , t \right) } - \left(\left(m ^ { 2 } - \nabla _ { x } ^ { 2 } \right) \phi \left(x , t \right) \right) \, \frac { \delta } { \delta \pi \left(x , t \right) } \right)
	\left[\partial _ { \mu } \partial ^ { \mu } - \partial _ { y } ^ { 2 } + e ^ { - 2 \sigma } \left\{ m ^ { 2 } + \left(\frac { 3 } { 1 6 } - \xi \right) R \right\} \right] \tilde { \varphi } + e ^ { - \frac { 7 } { 2 } \sigma } V ^ { \prime } (e ^ { \frac { 3 } { 2 } \sigma } \tilde { \varphi }) = 0 .	\left[\partial _ { \mu } \partial ^ { \mu } - \partial _ { y } ^ { 2 } + e ^ { - 2 \sigma } \left\{ m ^ { 2 } + \left(\frac { 3 } { 1 6 } - \xi \right) R \right\} \right] \tilde { \varphi } + e ^ { - \frac { 7 } { 2 } \sigma } V ^ { \prime } (e ^ { \frac { 3 } { 2 } \sigma } \tilde { \varphi }) = 0 .
	\begin{array} { r } { F _ { [j , j + 1] } = \frac { 1 } { 1 - g _ { L } g _ { R } - \alpha g _ { L } + \overline { \alpha } g _ { R } } \left(\begin{array} { l l } { 1 + g _ { L } g _ { R } + \alpha g _ { L } + \overline { \alpha } g _ { R } } & { 2 \rho g _ { L } } \\ { 2 \rho g _ { R } } & { 1 + g _ { L } g _ { R } - \alpha g _ { L } - \overline { \alpha } g _ { R } } \end{array} \right) , } \\ { \alpha = \alpha _ { j } , \rho = \rho _ { j } , g _ { L } (z) = z b _ { j } (z) , g _ { R } (z) = z f _ { j + 1 } (z) . } \end{array}	\begin{aligned} F_ { [j , j + 1] } = \frac { 1 } { 1 - g _ { L } g _ { R } - \alpha g _ { L } + \overline{\alpha} g _ { R} } \begin{pmatrix} 1 + g _ { L } g _ { R } + \alpha g _ { L } + \overline{\alpha} g_R & 2 \rho g_L \\ 2 \rho g_R & 1 + g _ { L } g _ { R } - \alpha g _ { L } - \overline{\alpha} g _ { R } \end{pmatrix}, \\ \noalign{\vskip5pt} \alpha = \alpha _ { j, \rho = \rho _ { j } , g _ { L } (z) = z b _ { j } (z) , g _ { R } (z) = z f _ { j + 1 } (z) . \end{aligned}
	f _ { i j } = \sum _ { j = \pm } f _ { i j } g _ { i } + \sum _ { j = \pm } f _ { i } g _ { - i - 2 } + \sum _ { j = \pm } g _ { i } f _ { - i - 2 } .	\begin{array} { r } { f g = \sum _ { i \in \mathbb Z } f _ { i } \tilde { g } _ { i } + \sum _ { i \in \mathbb Z } f _ { i } g _ { \le i - 2 } + \sum _ { i \in \mathbb Z } g _ { i } f _ { \le i - 2 } , } \end{array}
	\{ h , F (r > 1) f (h) a _ { 3 } \} \geq \sigma r ^ { - 1 - \epsilon } F (r > 1) f (h) + O (r ^ { - 1 - 2 \epsilon })	\{ h , F (r > 1) f (h) a _ { 3 } \} \geq \sigma r ^ { - 1 - \epsilon } F (r > 1) f (h) + O (r ^ { - 1 - 2 \epsilon })
	\omega = \sum _ { i = 1 } ^ { 2 m } d x _ { i } \wedge d p _ { i } , \quad \mathrm { a n d } \quad g = \sum _ { i = 1 } ^ { 2 m } (d x _ { i } ^ { 2 } + d p _ { i } ^ { 2 }) .	\omega = \sum _ { i = 1 } ^ { 2 m } d x _ { i } \wedge d p _ { i } , \ \ \ \mathrm { a n d } \ \ \ g = \sum _ { i = 1 } ^ { 2 m } (d x _ { i } ^ { 2 } + d p _ { i } ^ { 2 }) .
	\begin{array} { r l } { \left| \{ g _ { j } \} _ { j \in \mathbb { Z } } \right| _ { L ^ { p } \ell ^ { q } (\widehat { P }) } } & { \lesssim \left| \{ \gamma _ { j } E _ { j } (f _ { j }) \} _ { j \in \mathbb { Z } } \right| _ { L ^ { p } \ell ^ { q } (\widehat { P }) } } \\ & { \lesssim \left| \{ E _ { j } (f _ { j }) \} _ { j \in \mathbb { Z } } \right| _ { L ^ { p } \ell ^ { q } (\widehat { P }) } = \left| \{ f _ { j } \} _ { j \in \mathbb { Z } } \right| _ { L ^ { p } \ell ^ { q } (\widehat { P }) } , } \end{array}	\begin{array} { r l } { \left| \{ g _ { j } \} _ { j \in \mathbb { Z } } \right| _ { L ^ { p } \ell ^ { q } (\widehat { P }) } } & { \leq \left| \{ \gamma _ { j } E _ { j } (f _ { j }) \} _ { j \in \mathbb { Z } } \right| _ { L ^ { p } \ell ^ { q } (\widehat { P }) } } \\ & { \lesssim \left| \{ E _ { j } (f _ { j }) \} _ { j \in \mathbb { Z } } \right| _ { L ^ { p } \ell ^ { q } (\widehat { P }) } = \left| \{ f _ { j } \} _ { j \in \mathbb { Z } } \right| _ { L ^ { p } \ell ^ { q } (\widehat { P }) } , } \end{array}
	\begin{array} { r } { \mathcal { H } = \int d ^ { d } x \Big[\frac { 1 } { 2 } \sum _ { i } (\partial _ { \mu } \Phi _ { i }) ^ { 2 } + \frac { 1 } { 2 } \sum _ { i } r _ { i } \Phi _ { i } ^ { 2 } + \frac { 1 } { 4 ! } \sum _ { i j k l } u _ { i j k l } \Phi _ { i } \Phi _ { j } \Phi _ { k } \Phi _ { l } \Big] , } \end{array}	\begin{array} { r } { \mathcal { H } = \int d ^ { d } x \Bigl [{ \frac { 1 } { 2 } } \sum _ { i } (\partial _ { \mu } \Phi _ { i }) ^ { 2 } + { \frac { 1 } { 2 } } \sum _ { i } r _ { i } \Phi _ { i } ^ { 2 } + { \frac { 1 } { 4 ! } } \sum _ { i j k l } u _ { i j k l } \; \Phi _ { i } \Phi _ { j } \Phi _ { k } \Phi _ { l } \Bigr] , } \end{array}
	H ^ { \prime } = \beta m + \mathcal { E } + \beta \frac { \mathcal { O } ^ { 2 } } { 2 m } - \frac { 1 } { 8 m ^ { 2 } } [\mathcal { O } , [\mathcal { O } , \mathcal { E }]] + i \mathcal { \dot { O } }] - \beta \frac { \mathcal { O } ^ { 4 } } { 8 m ^ { 3 } } + \frac { \beta } { 2 m } [\mathcal { O } , \mathcal { E }] - \frac { \mathcal { O } ^ { 3 } } { 3 m ^ { 2 } } + \frac { i \beta \mathcal { \dot { O } } } { 2 m } \dots \, ,	H ^ { \prime } = \beta m + \mathcal { E } + \beta \frac { \mathcal { O } ^ { 2 } } { 2 m } - \frac { 1 } { 8 m ^ { 2 } } [\mathcal { O } , [\mathcal { O } , \mathcal { E }]] + i \mathcal { \dot { O } }] - \beta \frac { \mathcal { O } ^ { 4 } } { 8 m ^ { 3 } } + \frac { \beta } { 2 m } [\mathcal { O } , \mathcal { E }] - \frac { \mathcal { O } ^ { 3 } } { 3 m ^ { 2 } } + \frac { i \beta \mathcal { \dot { O } } } { 2 m } \dots \, ,
	f (W _ { l _ { 1 } } , \ldots , W _ { l _ { K } }) = \sum _ { l _ { 1 } , \ldots , l _ { K } = 0 } a _ { l _ { 1 } , \ldots , l _ { K } } W _ { t _ { 1 } } ^ { l _ { 1 } } \diamond \cdots \diamond W _ { t _ { K } } ^ { l _ { K } } .	\begin{array} { r } { f (W _ { t _ { 1 } } , \ldots , W _ { t _ { K } }) = \sum _ { l _ { 1 } , \ldots , l _ { K } \geq 0 } a _ { l _ { 1 } , \ldots , l _ { K } } W _ { t _ { 1 } } ^ { l _ { 1 } } \diamond \cdots \diamond W _ { t _ { K } } ^ { l _ { K } } . } \end{array}

Figure 9: Printed formula recognition results

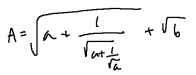
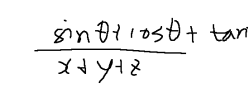
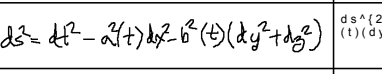
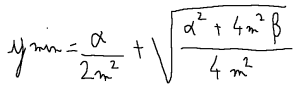
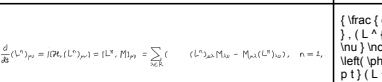
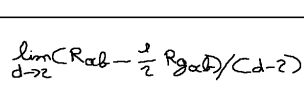
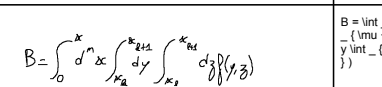
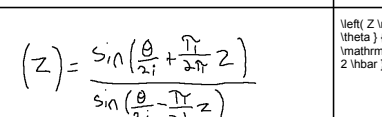
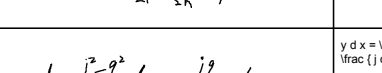
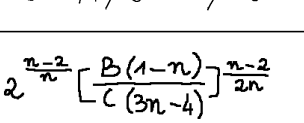
Images	Predicts	Labels
	A = \sqrt { a + \frac { 1 } { \sqrt { a + \frac { 1 } { \sqrt { a } } } } } + \sqrt { b }	A = \sqrt { a + \frac { 1 } { \sqrt { a + \frac { 1 } { \sqrt { a } } } } } + \sqrt { b }
	\frac { \sin \vartheta + i \cos \vartheta } { x + y + z } + \tan \vartheta	\frac { \sin \theta + \cos \theta + \tan \theta } { x + y + z }
	d s ^ { 2 } = d t ^ { 2 } - a ^ { 2 } (t) d x ^ { 2 } - b ^ { 2 } (t) (d y ^ { 2 } + d z ^ { 2 })	d s ^ { 2 } = d t ^ { 2 } - a ^ { 2 } (t) d x ^ { 2 } - b ^ { 2 } (t) (d y ^ { 2 } + d z ^ { 2 })
	{ \cal N } \, m = \frac { \alpha } { 2 \, m ^ { 2 } } + \sqrt { \frac { \alpha ^ { 2 } + 4 \, n ^ { 2 } \beta } { 4 \, \, n ^ { 2 } } }	y _ { m i n } = \frac { \alpha } { 2 m ^ { 2 } } + \sqrt { \frac { \alpha ^ { 2 } + 4 m ^ { 2 } \beta } { 4 m ^ { 2 } } }
	{ \frac { d } { d t } } (L ^ { n }) _ { \mu \nu } = i [{ \cal H } , (L ^ { n }) _ { \mu \nu }] = [L ^ { n } , M] _ { \mu \nu } \nonumber \, = \sum _ { \lambda \in { \cal R } } \left(\phantom { \mathrm { \huge ~ H } } \hspace { - 1 5 p t } (L ^ { n }) _ { \mu \lambda } M _ { \lambda \nu } - M _ { \lambda \nu } (L ^ { n }) _ { \mu \lambda } \right) , \quad n = 1 , \ldots .	{ \frac { d } { d t } } (L ^ { n }) _ { \mu \nu } = i [{ \cal H } , (L ^ { n }) _ { \mu \nu }] = [L ^ { n } , M] _ { \mu \nu } \nonumber \, = \sum _ { \lambda \in { \cal R } } \left(\phantom { \mathrm { \huge ~ H } } \hspace { - 1 5 p t } (L ^ { n }) _ { \mu \lambda } M _ { \lambda \nu } - M _ { \mu \lambda } (L ^ { n }) _ { \lambda \nu } \right) , \quad n = 1 , \ldots .
	\operatorname* { l i m } _ { d \rightarrow 2 } (R _ { a b } - \frac { 1 } { 2 } R g _ { a b }) / (d - 2)	\operatorname* { l i m } _ { d \rightarrow 2 } (R _ { a b } - \frac { 1 } { 2 } R g _ { a b }) / (d - 2)
	B = \int _ { 0 } ^ { x } d ^ { 4 } x \int _ { x _ { \mu } } ^ { x _ { \mu + 1 } } d y \int _ { x _ { \mu } } ^ { x _ { \mu } } d y \int (y _ { \mu }) ^ { x _ { \mu } } d y \int (y _ { \mu })	B = \int _ { 0 } ^ { x } d ^ { n } x \int _ { x _ { k } } ^ { x _ { k + 1 } } d y \int _ { x _ { l } } ^ { x _ { l + 1 } } d z f (y , z)
	\left(Z \right) = \frac { \mathrm { S i n } \left(\frac { \theta } { 2 i } + \frac { \Pi } { 2 \Pi } \, Z \right) } { \mathrm { S i n } \left(\frac { \theta } { 2 i } - \frac { \Pi } { 2 \hbar } \, Z \right) }	(z) = \frac { \sin (\frac \theta { 2 i } + \frac \pi { 2 \pi } z) } { \sin (\frac \theta { 2 i } - \frac \pi { 2 h } z) }
	y d x = \frac { j ^ { 2 } - q ^ { 2 } } { 1 + q ^ { 2 } } d y x - \frac { j q } { 1 + q ^ { 2 } } d x y	y d x = \frac { j ^ { 2 } - q ^ { 2 } } { 1 + q ^ { 2 } } d y x - \frac { j q } { 1 + q ^ { 2 } } d x y
	2 ^ { \frac { n - 2 } { n } } \biggl [\frac { B \left(1 - n \right) } { C \left(3 n - 4 \right) } \biggr] ^ { \frac { n - 2 } { 2 n } }	2 ^ { \frac { n - 2 } { n } } [\frac { B (1 - n) } { C (3 n - 4) }] ^ { \frac { n - 2 } { 2 n } }

Figure 10: Handwritten formula recognition results

References

[1] https://www.v7labs.com/open-datasets/aida.

[2] . https://www.kaggle.com/datasets/rvente/im2latex170k.

[3] . https://www.kaggle.com/datasets/gregoryeritsyan/im2latex-230k.

[4] . https://huggingface.co/datasets/AlFrauch/im2latex.

[5] What is image augmentation and how it can improve the performance of deep neural networks. URL https://albumentations.ai/docs/introduction/image_augmentation/#image-augmentation-to-the-rescue.

[6] https://huggingface.co/datasets/Azu/Handwritten-Mathematical-Expression-Convert-LaTeX.

[7] Katex. https://katex.org/.

[8] https://huggingface.co/datasets/OleehyO/latex-formulas.

[9] Xiaohang Bian, Bo Qin, Xiaozhe Xin, Jianwu Li, Xuefeng Su, and Yanfeng Wang. Handwritten mathematical expression recognition via attention aggregation based bi-directional mutual learning. *CoRR*, abs/2112.03603, 2021. URL https://arxiv.org/abs/2112.03603.

[10] Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents. *CoRR*, abs/2308.13418, 2023. doi: 10.48550/ARXIV.2308.13418. URL https://doi.org/10.48550/arXiv.2308.13418.

[11] Chungkwong Chan. Stroke extraction for offline handwritten mathematical expression recognition. *IEEE Access*, 8:61565–61575, 2020. doi: 10.1109/ACCESS.2020.2984627.

[12] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and

Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/9a39b4925e35cf447ccba8757137d84f-Abstract-Conference.html.

[13] Yuntian Deng, Anssi Kanervisto, and Alexander M. Rush. What you get is what you see: A visual markup decompiler. *CoRR*, abs/1609.04938, 2016. URL http://arxiv.org/abs/1609.04938.

[14] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M. Rush. Image-to-markup generation with coarse-to-fine attention. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 980–989. PMLR, 2017. URL http://proceedings.mlr.press/v70/deng17a.html.

[15] Haisong Ding, Kai Chen, and Qiang Huo. An encoder-decoder approach to handwritten mathematical expression recognition with multi-head attention and stacked decoder. In Josep Lladós, Daniel Lopresti, and Seiichi Uchida, editors, *Document Analysis and Recognition – ICDAR 2021*, pages 602–616, Cham, 2021. Springer International Publishing. ISBN 978-3-030-86331-9.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[17] Trung Hoang and Bao Thai. Fushion image to latex dataset, 2024. https://www.kaggle.com/datasets/hongtrung/image-to-latex-dataset.

[18] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL http://arxiv.org/abs/1608.06993.

[19] Anh Duc Le. Recognizing handwritten mathematical expressions via paired dual loss attention network and printed mathematical expressions. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2413–2418, 2020. doi: 10.1109/CVPRW50498.2020.00291.

[20] Anh Duc Le, Bipin Indurkhya, and Masaki Nakagawa. Pattern generation strategies for improving recognition of handwritten mathematical expressions. *CoRR*, abs/1901.06763, 2019. URL http://arxiv.org/abs/1901.06763.

[21] Anh Duc Le, Bipin Indurkhya, and Masaki Nakagawa. Pattern generation strategies for improving recognition of handwritten mathematical expressions. *Pattern Recognit. Lett.*, 128:255–262, 2019. doi: 10.1016/J.PATREC.2019.09.002. URL https://doi.org/10.1016/j.patrec.2019.09.002.

[22] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, 10:707–710, 1965. URL https://api.semanticscholar.org/CorpusID:60827152.

[23] Zhe Li, Lianwen Jin, Songxuan Lai, and Yecheng Zhu. Improving attention-based handwritten mathematical expression recognition with scale augmentation and drop attention. In *17th International Conference on Frontiers in Handwriting Recognition, ICFHR 2020, Dortmund, Germany, September 8-10, 2020*, pages 175–180. IEEE, 2020. doi: 10.1109/ICFHR2020.2020.00041. URL https://doi.org/10.1109/ICFHR2020.2020.00041.

[24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, abs/2103.14030, 2021. URL https://arxiv.org/abs/2103.14030.

[25] Mahshad Mahdavi, Richard Zanibbi, Harold Mouchère, Christian Viard-Gaudin, and Utpal Garain. ICDAR 2019 CROHME + TFD: competition on recognition of handwritten mathematical expressions and typeset formula detection. In *2019 International Conference on Document Analysis and Recognition, ICDAR 2019, Sydney, Australia, September 20-25,*

2019, pages 1533–1538. IEEE, 2019. doi: 10.1109/ICDAR.2019.00247. URL https://doi.org/10.1109/ICDAR.2019.00247.

[26] Harold Mouchère, Christian Viard-Gaudin, Richard Zanibbi, and Utpal Garain. ICFHR 2014 competition on recognition of on-line handwritten mathematical expressions (CROHME 2014). In *14th International Conference on Frontiers in Handwriting Recognition, ICFHR 2014, Crete, Greece, September 1-4, 2014*, pages 791–796. IEEE Computer Society, 2014. doi: 10.1109/ICFHR.2014.138. URL https://doi.org/10.1109/ICFHR.2014.138.

[27] Harold Mouchère, Christian Viard-Gaudin, Richard Zanibbi, and Utpal Garain. ICFHR2016 CROHME: competition on recognition of online handwritten mathematical expressions. In *15th International Conference on Frontiers in Handwriting Recognition, ICFHR 2016, Shenzhen, China, October 23-26, 2016*, pages 607–612. IEEE Computer Society, 2016. doi: 10.1109/ICFHR.2016.0116. URL https://doi.org/10.1109/ICFHR.2016.0116.

[28] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL, 2002. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040/.

[29] Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1310–1318. JMLR.org, 2013. URL http://proceedings.mlr.press/v28/pascanu13.html.

[30] Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. Mathbert: A pre-trained model for mathematical formula understanding. *CoRR*, abs/2105.00377, 2021. URL https://arxiv.org/abs/2105.00377.

[31] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL http://arxiv.org/abs/1506.02640.

[32] Sumeet S. Singh. Teaching machines to code: Neural markup generation with visual attention. *CoRR*, abs/1802.05415, 2018. URL http://arxiv.org/abs/1802.05415.

[33] Matthias Springstein, Eric Müller-Budack, and Ralph Ewerth. Unsupervised training data generation of handwritten formulas using generative adversarial networks with self-attention. *CoRR*, abs/2106.09432, 2021. URL https://arxiv.org/abs/2106.09432.

[34] Masakazu Suzuki, Fumikazu Tamari, Ryoji Fukuda, Seiichi Uchida, and Toshihiro Kanahori. INFTY: an integrated OCR system for mathematical documents. In *Proceedings of the 2003 ACM Symposium on Document Engineering, Grenoble, France, November 20-22, 2003*, pages 95–104. ACM, 2003. doi: 10.1145/958220.958239. URL https://doi.org/10.1145/958220.958239.

[35] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 2019. URL http://proceedings.mlr.press/v97/tan19a.html.

[36] Thanh-Nghia Truong, Cuong Tuan Nguyen, Khanh Minh Phan, and Masaki Nakagawa. Improvement of end-to-end offline handwritten mathematical expression recognition by weakly supervised learning. In *17th International Conference on Frontiers in Handwriting Recognition, ICFHR 2020, Dortmund, Germany, September 8-10, 2020*, pages 181–186. IEEE, 2020. doi: 10.1109/ICFHR2020.2020.00042. URL https://doi.org/10.1109/ICFHR2020.2020.00042.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

[38] Jian Wang, Yunchuan Sun, and Shenling Wang. Image to latex with densenet encoder and joint attention. In Rongfang Bie, Yunchuan Sun, and Jiguo Yu, editors, *2018 International Conference on Identification,*

Information and Knowledge in the Internet of Things, IIKI 2018, Beijing, China, October 19-21, 2018, volume 147 of *Procedia Computer Science*, pages 374–380. Elsevier, 2018. doi: 10.1016/J.PROCS.2019.01.246. URL https://doi.org/10.1016/j.procs.2019.01.246.

[39] Zelun Wang and Jyh-Charn Liu. Translating math formula images to latex sequences using deep neural networks with sequence-level training. *Int. J. Document Anal. Recognit.*, 24(1):63–75, 2021. doi: 10.1007/S10032-020-00360-2. URL https://doi.org/10.1007/s10032-020-00360-2.

[40] Jin-Wen Wu, Fei Yin, Yan-Ming Zhang, Xu-Yao Zhang, and Cheng-Lin Liu. Image-to-markup generation via paired adversarial learning. In Michele Berlingerio, Francesco Bonchi, Thomas Gärtner, Neil Hurley, and Georgiana Ifrim, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2018, Dublin, Ireland, September 10-14, 2018, Proceedings, Part I*, volume 11051 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2018. doi: 10.1007/978-3-030-10925-7_2. URL https://doi.org/10.1007/978-3-030-10925-7_2.

[41] Jin-Wen Wu, Fei Yin, Yan-Ming Zhang, Xu-Yao Zhang, and Cheng-Lin Liu. Handwritten mathematical expression recognition via paired adversarial learning. *Int. J. Comput. Vis.*, 128(10):2386–2401, 2020. doi: 10.1007/S11263-020-01291-5. URL https://doi.org/10.1007/s11263-020-01291-5.

[42] Jianshu Zhang, Jun Du, and Li-Rong Dai. A gru-based encoder-decoder approach with attention for online handwritten mathematical expression recognition. *CoRR*, abs/1712.03991, 2017. URL http://arxiv.org/abs/1712.03991.

[43] Jianshu Zhang, Jun Du, Shiliang Zhang, Dan Liu, Yulong Hu, Jin-Shui Hu, Si Wei, and Li-Rong Dai. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognit.*, 71:196–206, 2017. doi: 10.1016/J.PATCOG.2017.06.017. URL https://doi.org/10.1016/j.patcog.2017.06.017.

[44] Jianshu Zhang, Jun Du, and Lirong Dai. Multi-scale attention with dense encoder for handwritten mathematical expression recognition. *CoRR*, abs/1801.03530, 2018. URL http://arxiv.org/abs/1801.03530.

[45] Jianshu Zhang, Jun Du, Yongxin Yang, Yi-Zhe Song, Si Wei, and Lirong Dai. A tree-structured decoder for image-to-markup generation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 11076–11085. PMLR, 2020. URL http://proceedings.mlr.press/v119/zhang20g.html.

[46] Wenqi Zhao and Liangcai Gao. Comer: Modeling coverage for transformer-based handwritten mathematical expression recognition. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 392–408, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-19815-1.

[47] Wenqi Zhao, Liangcai Gao, Zuoyu Yan, Shuai Peng, Lin Du, and Ziyin Zhang. Handwritten mathematical expression recognition with bidirectionally trained transformer. *CoRR*, abs/2105.02412, 2021. URL https://arxiv.org/abs/2105.02412.

[48] Mingle Zhou, Ming Cai, Gang Li, and Min Li. An end-to-end formula recognition method integrated attention mechanism. *Mathematics*, 11 (1), 2023. ISSN 2227-7390. doi: 10.3390/math11010177. URL https://www.mdpi.com/2227-7390/11/1/177.