Transactions on Computer Science and Intelligent Systems Research
ISSN: 2960-1800, eISSN: 2960-2238 | Volume 5
AIDML 2024

Mixed Text and Formula Recognition Using ResNet and Transformer

Chenhe Yang*

Department of Computer science and technology, South China University of Technology, Guangzhou, China

*Corresponding author: 202130192352@mail.scut.edu.cn

Abstract. Recent advancements in deep learning have enabled the recognition of images, objects, observations, texts, and other complex structures with increasing accuracy. In recent years, scene text recognition has stimulated a lot of researchers in the computer vision community. However, it still needs improvement due to the poor performance of existing scene recognition algorithms. This paper aims to combine deep learning research to recognize texts and formulas in images and convert them into LaTeX format. More accurately and efficiently recognizing and converting images containing mixed text and mathematical formulas can improve the quality of the generated LaTeX code. In terms of the model, this paper utilizes ResNet as an Encoder for feature extraction and Transformer as a Decoder for text generation, improving the accuracy of the generated text. Two types of datasets were used in the process: pure formulas and a mixture of text and formulas. This paper concludes that the effects on both datasets are good, with CERs of less than 0.05 and a loss of only 0.10. Besides, the accuracy of pure formula recognition is higher than that of mixed types, possibly because there is more data on pure formulas and it is difficult to accurately extract text features in mixed types. This method can automatically parse and generate content containing mathematical formulas, promoting the development of the education industry and improving the efficiency and quality of digital content generation.

Keywords: Deep Learning; LaTeX Conversion; Mathematical Formula Recognition; Transformer; ResNet.

1. Introduction

Language and text have always been a focus of research in various fields. In recent years, scene text recognition has stimulated a lot of researchers in the field of computer vision. However, due to the poor performance of existing scene recognition algorithms, there is still room for improvement [1]. There has been a continuous effort from predecessors on how to generate text accurately, which is also applied in academia. In academic research, especially in fields such as mathematics, physics, engineering, and economics, many research papers and technical documents contain complex mathematical formulas. Therefore, it is crucial to quickly, efficiently, and correctly extract formulas and convert them into usable LaTeX code.

The task of generating text from images involves two processes: feature extraction from images and text generation. Before the rapid development of deep learning, feature extraction methods, such as Connected Component Analysis (CCA), were mostly performed manually. Methods based on CCA first extract candidate components in various ways (for example, through color clustering or extreme region extraction) and then use manually designed rules or classifiers automatically trained on handcrafted features to filter out non-text components [2]. With the development of deep learning, there came methods based on CNN for sliding window feature extraction and RNN or LSTM for text generation. RNN is a unique type of neural network that can utilize past feature information and handle sequential input [1].

However, the drawbacks of both methods are obvious. Traditional manual feature extraction is highly sensitive to noise when the computational load is significant. Minor noise points in an image may be misidentified as separate connected components, leading to over-segmentation. Regarding text

generation, RNN has limitations in parallel processing and handling long sequences. Therefore, this paper employs ResNet as an Encoder for feature extraction, which can effectively solve the problems of gradient vanishing and explosion. With the development of deep learning, the Transformer emerged and became a focus in the field. Transformer represents a significant breakthrough in the field of natural language processing, quickly becoming widely used in the domain. It is entirely based on the attention mechanism, completely eliminating the need for recursion and convolution [3]. The Transformer enables parallelization and performance improvement through a self-attention mechanism and uses a Transformer-based Decoder for text generation. The combination of these two models effectively improves the accuracy of the generated results.

Currently, many commercial tools can convert images to text, especially LaTeX code. For example, Mathpix supports cross-platform use and can recognize mathematical formulas in images and convert them into LaTeX code; the web-based MyScript Math Sample can convert handwritten mathematical formulas into LaTeX code. MyScript Math Sample uses MyScript's interactive ink technology, among others. These tools are undoubtedly of great importance for academic publishing, online educational content creation, digital document management, and other fields.

The main focus of this paper is on how to efficiently and correctly convert images containing formulas into texts containing LaTeX codes using deep learning. In terms of models, it adopts an Encoder-decoder approach with strong representational capabilities and more flexibility, combining ResNet and Transformer for feature extraction and text generation tasks. It uses AdamW and cross-entropy functions, among others, for training to improve the efficiency of converting images containing formulas into texts containing LaTeX codes.

2. Dataset and Method

2.1. Dataset

As shown in Fig. 1, Fig. 1 is an example of the task and represents our dataset. The data annotated in Fig. 1 are above, and the labels are below.

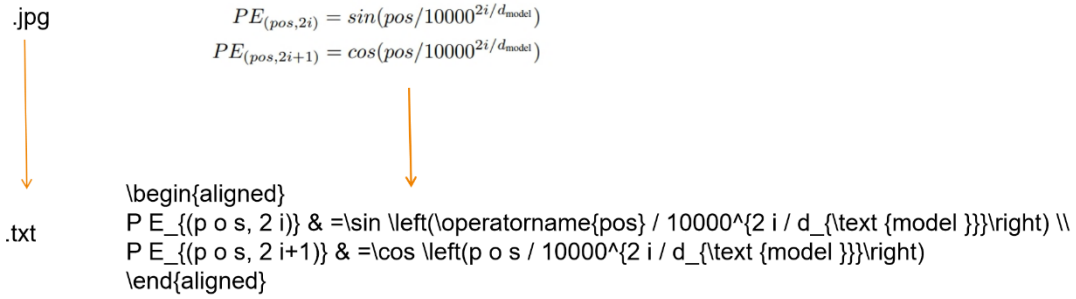

Fig. 1 The Example of tasks

The data source used in this article comes from the formulas annotated by students of South China University of Technology in middle and high school students' exam papers through LabelImg. Mathpix generates the labels after annotation and then preprocesses them to obtain the final dataset. LabelImg is a popular graphical image annotation tool that supports annotating image data for machine learning projects. This tool is mainly used to create annotation files for training object detection models, such as annotating the bounding boxes of objects in images. Fig. 2 displays the process of using LabelImg for annotation, during which several thousand exam papers were annotated.

The annotated dataset is divided into two categories, one is a pure formula dataset, as shown in Fig. 3 above, which contains only formulas, and the other is a mixed text and formula dataset, as shown in Fig. 3 below, which contains both Chinese text and formulas.

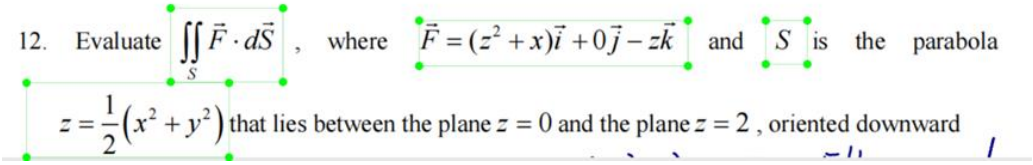

Fig. 2 The annotation process using LabelImg. and generates XML files

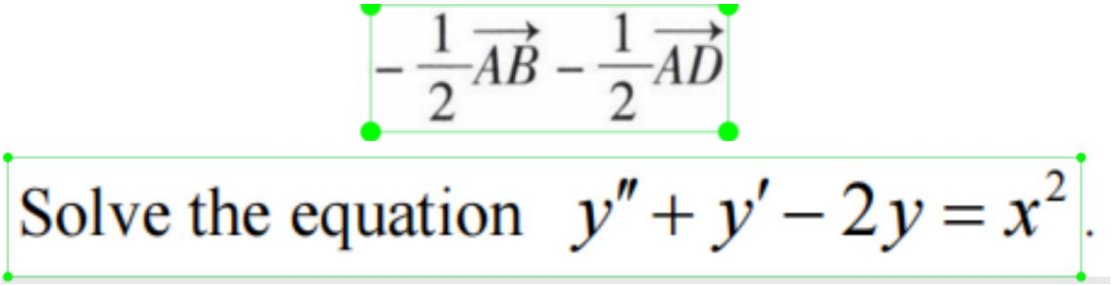

Fig.3 two types of datasets, with the top part being pure formulas and the bottom part being a mixture of text and formulas

Besides dataset creation, data preprocessing is also very important. Data obtained directly from the source may have inconsistencies and errors. Data preprocessing can make the impossible possible, making data adapt to the input requirements of each data mining algorithm [4].

During the cleaning process, it is necessary to segment the data according to a vocabulary list, preliminarily filter it based on the vocabulary list, and filter out multi-line and wrongly recognized data. Entries that do not meet the criteria are removed from the dataset. This step ensures the quality of the data and prevents the model from learning incorrect information. Then, align the filtered data to ensure that each image's label is accurate. Then, the final adjustments to the data will be made according to the input and output formats of the project. Finally, based on the needs of the neural network model, decide whether the images need to be augmented and to what size, ensuring that all input data dimensions are consistent for easy model processing.

2.2. Method

The method used in this paper consists of two parts: the model structure and the function settings.

Regarding model structure, a pre-trained model, ResNet18, is adopted as the Encoder. The Transformer serves as the Decoder for feature extraction and text generation. The model first uses the predefined ResNet architecture to extract features from the input image. ResNet extracts high-level features of the image through multiple convolutional layers, as shown in Table 1. Only the first three layers of residual structure are used in this study. Because the output channel number of ResNet (e.g., 256) is usually different from the input dimension (d_model) expected by the Transformer encoder (decoder), a convolutional layer is used at the end of ResNet to convert the output channel number of ResNet to the dimension required by the Transformer. Since convolution operations do not naturally handle the positional information in sequences as self-attention mechanisms do, positional information is added before passing features to the Transformer. Two-dimensional positional encoding is used to encode image features, enabling the model to understand the relative or absolute positions of pixels in the image. After positional encoding, the feature maps are flattened and rearranged to meet the input requirements of the Transformer. The processed image feature sequence is then fed into the Transformer decoder. In the decoding process, positional encoding and masking operations are also performed on the input of the decoder according to the required output sequence length (to ensure the autoregressive property of decoding) to generate the final output sequence.

Table 1. The network structure of ResNet-18, where only up to the first three residual structure layers [5]

Layer Name	Output Size	ResNet-18
conv1	112 x 112 x 64	7 x 7, 64, stride 2
conv2_x	56 x 56 x 64	3 x 3 max pool, stride 2
		$\begin{bmatrix} 3\times 3, & 64 \\ 3\times 3, & 64 \end{bmatrix} \times 2$
conv3_x	28 x 28 x 128	$\begin{bmatrix} 3\times 3, & 128 \\ 3\times 3, & 128 \end{bmatrix} \times 2$
conv4_x	14 x 14 x 256	$\begin{bmatrix} 3\times 3, & 256 \\ 3\times 3, & 256 \end{bmatrix} \times 2$
conv5_x	7 x 7 x 512	$\begin{bmatrix} 3\times 3, & 512 \\ 3\times 3, & 512 \end{bmatrix} \times 2$
average pool	1 x 1 x 512	7 x 7 average pool
fully connected	1000	512 x 1000 full connections
softmax	1000	

Regarding function usage, three main functions are utilized: the loss function, the optimization function, and the learning rate adjustment function.

The loss function adopted is the classic cross-entropy function, which measures the difference between two probability distributions. In machine learning, cross-entropy is commonly used to measure the difference between the probability distribution predicted by the model and the probability distribution of the actual labels, guiding the optimization process of the model [6]. This is the standard method for evaluating the performance of multi-classification tasks. The cross-entropy loss effectively captures the difference between the model's predicted probability distribution and the actual distribution, which is crucial for model training.

For learning rate adjustment, the MultiStepLR learning rate scheduler is utilized. MultiStepLR is a strategy for dynamically adjusting the learning rate by reducing it at predetermined training "milestones," allowing for more refined weight updates at different stages of the training process. By reducing the learning rate at specific training cycles (milestones), this scheduler helps the model escape local minima while making fine adjustments with lower learning rates in the later stages of training to avoid overfitting. This method ensures that the learning rate can be adaptively adjusted at different stages of model training, promoting more effective and stable model convergence.

The optimization function employs a variant of the AdamW optimizer. AdamW is an improved version of the Adam optimization algorithm, designed to resolve Adam's potential conflict between weight decay and adaptive learning rate adjustment. Adam adjusts the learning rate for each parameter by computing the first and second moments of the gradients, achieving faster convergence and better stability, especially in deep-learning applications. However, applying weight decay directly on the gradients in Adam could lead to an unstable optimization path. AdamW separates weight decay from the gradient updates, directly decaying the parameters, effectively utilizing weight decay for regularization, and reducing the risk of overfitting. This approach retains Adam's advantages, such as adaptive learning rate adjustment, while enhancing the model's generalization ability through an improved weight decay strategy [7].

The combination of these three strategies plays a key role in the model's training process. The improved weight decay mechanism of AdamW, the explicit objective of the cross-entropy loss

function, and the flexible learning rate adjustment strategy of MultiStepLR together provide a stable and effective optimization path for the model.

2.3. Performance Evaluation

In terms of performance evaluation, the growth of translation studies necessitates the development of accurate automatic evaluation metrics, allowing us to track machine translation performance [8].

We adopted three main metrics: Edit Distance Score, Exact Match Score, and BLEU Score.

The Edit Distance Score is based on the Levenshtein distance to calculate the difference between predicted and actual labels, with a higher score indicating fewer modifications needed. The formula for calculating the Edit Distance Score is as follows:

$$Edit\ Distance = \left(1 - \frac{\sum distance.\text{levenshtein}(\text{references}_i, \text{hypotheses}_i)}{\sum \max(\text{length}(\text{references}_i), \text{length}(\{\backslash rm\ hypotheses\}\_i))}\right) \times 100 \quad (1)$$

The Exact Match Score checks whether the prediction and the real label are completely identical; if they are, the score is 100%. Otherwise, it is 0. The formula for the Exact Match Score is as follows:

$$\text{Exact Match Score} = \frac{\text{Number of perfect matches}}{|D|} \times 100 \quad (2)$$

For the BLEU Score its use as an evaluation metric is based on the assumption that it is related to the real-world utility of these systems and predicts the actual utility of these systems, which can be measured by external measurements (for example, through task performance) or user satisfaction [9]. Therefore, it evaluates the quality of translation by calculating the n-gram overlap between predictions and reference labels, with a higher score indicating a higher quality of translation.

Additionally, there is an overall score metric, which is the arithmetic mean of these three scoring metrics, as shown in Formula 3, and it serves as the basis for final ranking.

$$\text{Overall Score} = \frac{\text{BLEU Score} + \text{Edit Distance Score} + \text{Exact Match Score}}{3} \quad (3)$$

In summary, evaluating using the BLEU Score, Edit Distance Score, and Exact Match Score comprehensively measures the performance of generating LaTeX code. This method considers vocabulary match, similarity between texts, and cases of complete consistency, providing a more comprehensive and detailed performance evaluation. Combining these metrics can more accurately reflect the model's strengths and weaknesses in different aspects.

3. Result

During the training process, as can be seen from Fig.4 and Fig.5, a significant number of iterations were involved, aimed at reducing the model's loss on the training set and improving its performance on the validation and test sets. Furthermore, from Tables 2 and 3, which display training data on two types of datasets, regardless of the dataset type, from the initial sharp decrease in loss to a gradual stabilization, the model exhibited a good learning trend. The training loss showed good convergence, with the loss on both training sets decreasing and stabilizing between 0.05 and 0.07. The performance on the test sets was also satisfactory, with a character error rate of only 0.035 on both training sets.

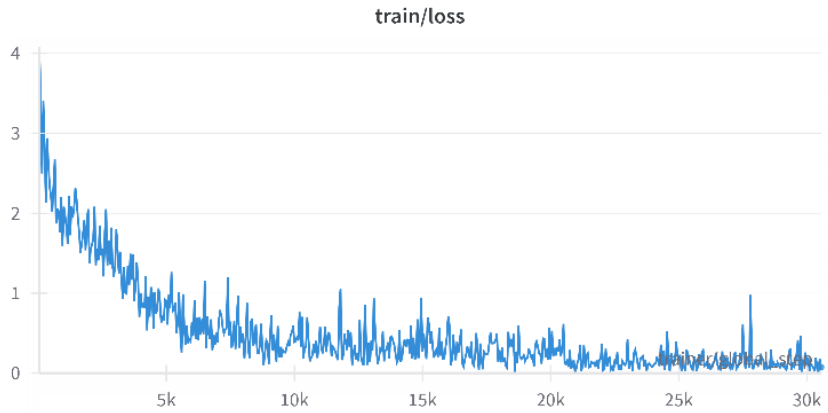

Fig 4. Loss variation over iteration numbers for the task on a pure mathematical expression dataset, along with training information on the test set, training set, and validation set

Fig. 4 illustrates the change in loss over iteration times on a pure mathematical expression dataset. Table 2 shows the specific value of variables during training.

Table 2. Variable Values for Training Data on pure formulas

Variable	Value
Test cer	0.03441
Train loss	0.07515
Train step number	30570
Val cer	0.03518
Val loss	0.11582

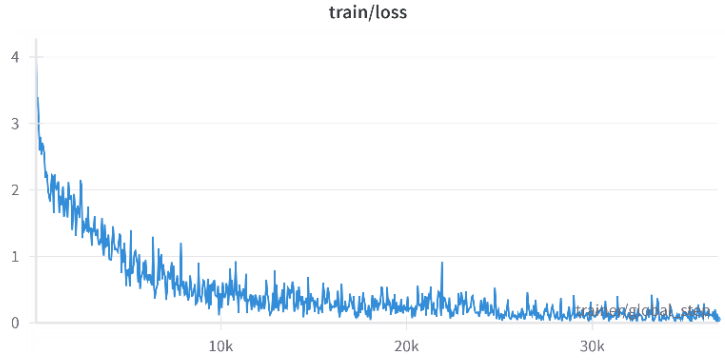

Fig 5. Loss variation over iteration numbers for the task on a mixed mathematical expression dataset, along with training information on the test set, training set, and validation set

Table 3. Variable Values for Training Data on mixed formulas

Variable	Value
Test cer	0.03510
Train loss	0.05466
Train step number	36765
Val cer	0.04838
Val loss	0.12262

Fig. 5 illustrates the change in loss over iteration times on a mixed mathematical expression dataset. Table 3 shows the specific value of variables during training.

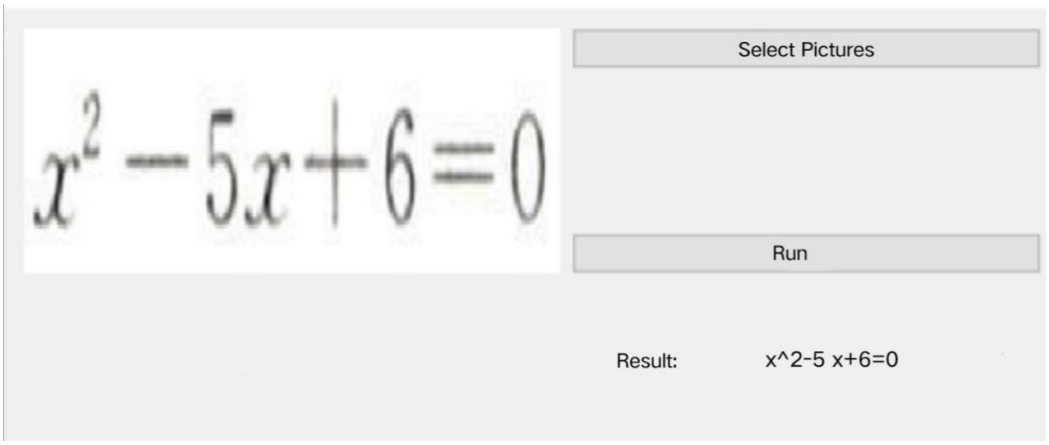

Fig. 6 User interface for recognizing

Additionally, for visualization and to test the accuracy of the results, as shown in Fig. 6, we conducted a test on a simple equation. This demonstrates the implementation of a user interface (UI) for visualization.

Overall, the model performed well during the training process, with steadily declining loss values and ultimately low character error rates on both the test and validation sets, indicating the model's good generalization ability.

4. Discussion

Figures 5 and 6 showed that the error in mixed text and formula scenarios is greater than that in pure formulas. The reason for this could be attributed to imbalanced data distribution, possibly because the proportion of formulas far exceeds that of text and the set of characters used in formulas is fixed, making the model more sensitive to formulas. Additionally, the diversity in text types makes it challenging to capture text features accurately.

To optimize and improve the model further, future research could start from the aspect of data augmentation. The more data the algorithm can access, the higher its efficiency. Even if the data quality is low, as long as the model can extract useful information from the original dataset, the algorithm can perform better [10]. A novel method of data augmentation, MeshCut, involves overlaying a grid mask during the training phase, transforming the image into a mosaic composed of multiple image segments, as shown in Fig. 1. Since there is no gap between the segments, the overall features of the target are effectively decomposed into multiple local features, providing more diverse information for the network [11].

In addition, extending the training duration might also help the model achieve a better state of convergence. More training iterations mean the model has more opportunities to learn patterns in the data, especially those rare formula structures or symbols. However, it's important to be mindful of overfitting during the process.

5. Conclusion

This study concludes that the use of deep learning models, especially the Transformer, significantly enhances the accuracy of converting images to text. After completing over 30,000 training steps on two datasets, the training loss decreased to approximately 0.075 for pure formulas and 0.055 for mixed formulas, with the validation loss being slightly higher than the training loss. This discrepancy is normal since the validation data was not used during training, reflecting the model's generalization ability on unseen data. The Character Error Rate (CER) on the test set is 0.0344, with the validation

sets for pure and mixed formulas showing CERs of 0.0351 and 0.0483, respectively, indicating that the model achieved a low error rate on both independent datasets.

The core goal of this project is to enhance the recognition capabilities for text and mathematical formulas in images, which is crucial for automated document processing and information extraction. The model demonstrated good performance in converting images to LaTeX code tasks, characterized by a low character error rate and a stable trend of decreasing loss.

However, any machine learning project should consider the risk of overfitting. Training loss being lower than validation loss could be an early sign of overfitting. Therefore, even though the current results are encouraging, it is essential to continue monitoring the model's performance by introducing more validation and test data to ensure the model's generalization ability. Additionally, exploring regularization techniques such as Dropout or L1/L2 regularization could help mitigate the risk of overfitting.

Lastly, for broader applications, the deployment and integration of the model should also be considered. This involves ensuring that the model can operate quickly and accurately across different hardware and software environments and can be seamlessly integrated with other systems, such as document management and content analysis tools.

References

[1] Kantipudi, M., Kumar, S., & Jha, A. (2021). Scene Text Recognition Based on Bidirectional LSTM and Deep Neural Network. Computational Intelligence and Neuroscience, 2021, Article 2676780. https://doi.org/10.1155/2021/2676780

[2] Long, S., He, X., & Yao, C. (2018). Scene Text Detection and Recognition: The Deep Learning Era. International Journal of Computer Vision, 129(1-2), 161-184. https://doi.org/10.1007/s11263-020-01369-0

[3] Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. Neural Information Processing Systems.

[4] García, S., Luengo, J., & Herrera, F. (2014). Data Preprocessing in Data Mining. Springer. https://doi.org/10.1007/978-3-319-10247-4

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 770-778). Las Vegas, NV, USA. https://doi.org/10.1109/CVPR.2016.90

[6] Chen, X., Kar, S., & Ralescu, D.A. (2012). Cross-entropy measure of uncertain variables. Information Sciences, 201, 53-60. https://doi.org/10.1016/j.ins.2012.02.049

[7] Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

[8] Lee, S., Lee, J., Moon, H., Park, C., Seo, J., Eo, S., Koo, S., & Lim, H. (2023). A Survey on Evaluation Metrics for Machine Translation. Mathematics, 11(4), Article 1006. https://doi.org/10.3390/math11041006

[9] Reiter, E. (2018). A Structured Review of the Validity of BLEU. Computational Linguistics, 44(3), 393–401. https://doi.org/10.1162/coli_a_00322

[10] Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621.

[11] Jiang, W., Zhang, K., Wang, N., & Yu, M. (2020). MeshCut data augmentation for deep learning in computer vision. PLOS ONE, 15(12), e0243613. https://doi.org/10.1371/journal.pone.0243613