A Deep Learning Method for Mathematical Formulas Detection in PDF Documents

Nghia Vo Trong
Faculty of Information Technology
University of Science, VNU−HCM
Ho Chi Minh City, Vietnam
20120536@student.hcmus.edu.vn

Van-Loc Nguyen
Faculty of Information Technology
University of Science, VNU−HCM
Ho Chi Minh City, Vietnam
20120131@student.hcmus.edu.vn
ORCID: 0000-0001-9351-3750

Minh-Tam Nguyen Kieu
Faculty of Information Technology
University of Science, VNU−HCM
Ho Chi Minh City, Vietnam
20120572@student.hcmus.edu.vn

Dang Nguyen Hai
Faculty of Information Technology
University of Science, VNU−HCM
Ho Chi Minh City, Vietnam
nhdang@selab.hcmus.edu.vn

Minh-Triet Tran
Faculty of Information Technology
University of Science, VNU−HCM
Ho Chi Minh City, Vietnam
tmtriet@fit.hcmus.edu.vn

Quan Vu Hai
Vietnam National University HCM
Ho Chi Minh City, Vietnam
vhquan@vnuhcm.edu.vn

***Abstract*—In this paper, we provide a deep learning method to detect mathematical formulas in scientific PDF documents. This task is quite different from the extraction of mathematical expressions in images. The task of mathematical formulas detection has three main challenges: a large scale span, a large variation of the ration between the width and the height, and rich character set and mathematical expressions. Considering theses challenges, we use Faster R-CNN, a real-time object detection model, with ResNet50, and a suitable level of Feature Pyramid Network. Our model is trained, tested and evaluated on the IBEM dataset and provides significant results on both embedded and isolated formulas.**

***Index Terms*—Mathematical Formulas Detection, PDF Documents, Deep Learning, Faster R-CNN**

I. Introduction

II. Related Workk

For many years, the detection of mathematical formulas has been recognized as a difficult task [2]. There are several existing methods for detecting mathematical expressions in PDF documents using formatting information, for example, page layout, character labels, character locations, font sizes, etc. However, there are many different tools for generating PDF documents, and there is a variants in their character quality. Lin et al. [7] show that mathematical formulas can be a composition of some object types. For instance, the square root sign in a PDF generated by LaTeXconsists of the text object representing a radical sign and a graphical object for the horizontal line, which results in the fact that some symbols must be identified from multiple drawings elements [9].

Lin et al. [7] categorized the methods of formula detection into three types, based on the features they used, which are: character-based, image-based and layout-based methods. The first type, character-based methods use OCR to identify characters, and those which are not recognized by the OCR engine are considered candidates for mathematical expressions. The second category uses image segmentation, which we will not mention is this paper because our method process PDF documents only. The last one, layout-based methods use features such as line height, line spacing, alignment, etc to detect formulas. A lot of published papers use a combination of character, layout, and context features [9].

A. Traditional Methods

Chaudhuri and Garain investigated over 10000 document pages and found out the frequencies of each mathematical character in formulas [3]. These frequencies are used to develop a detector for embedded expressions, which scans each text line and decides if the line consists of one of the 25 most frequent characters. After finding the leftmost word that contains a mathematical symbol, the detector enlarges the region around the word on the left and right with rules to find the formula region.

There is another method based on mathematical symbols location, and then growing formula regions around the symbols. This method used fuzzy logic, which was developed by Kacem et al. [5].

In 2011, Lin et. al proposed a four-step detection process, which finds embedded math formulas by merging characters tagged as math characters [7]. SVM classification was used for both character classification into math and non-math, and isolated math formulas detection.

B. CRF and Deep Learning Techniques

For digital PDF documents, in 2017, Iwatsuki et al. [4] developed a manually annotated dataset and applied conditional random fields (CRF) for detecting the zone of math expressions using both layout features (such as font types) and linguistic features (such as n-grams) extracted from PDF documents.

(to be continued)

III. METHOD

We use the Faster R-CNN model with ResNet50 as the backbone for our model. Moreover, the Feature Pyramid Network (FPN) module is used to improve our solution. The Faster R-CNN is a real-time object detection model, which consists of 2 modules. The first module of the Faster R-CNN model is a deep fully convolutional network that proposes regions. The second module is a detector that uses proposed regions from the first one [10]. This is a single, unified network for object detection. By using the recently popular terminology of neural networks as the 'attention' mechanisms, the Region Proposal Networks (RPN) tells the Faster R-CNN where to look.

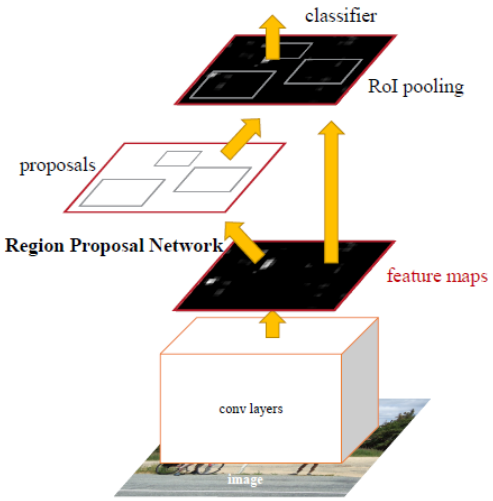

Fig. 1. Faster R-CNN is a single, unified network for object detection. The RPN module plays the role of the 'attention' of this network.

A. Region Proposal Networks

An RPN takes an image as input and outputs a set of rectangular object proposals each with an objectness score, which measures membership to a set of object classes. This process is modeled with a fully convolutional network. The structure of the RPN is shown in Figure 2, and figure 3 shows some examples of object detection using RPN proposals, on the PASCAL VOC 2007 test. The method introduced in [10] detects objects in a wide range of scales and aspect ratios.

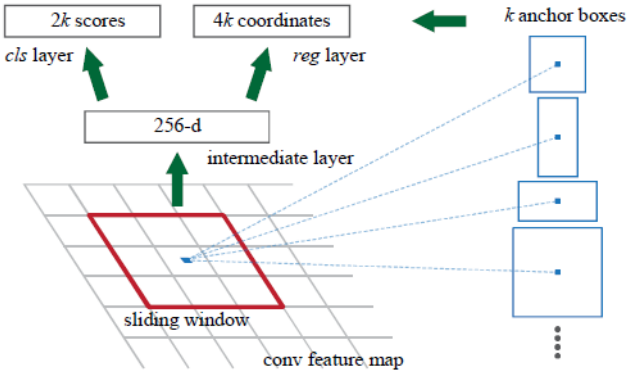

Fig. 2. Region Proposal Network (RPN)

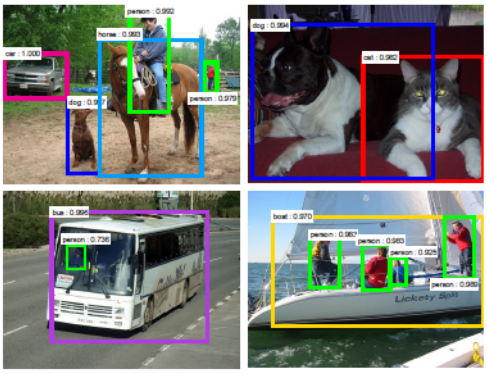

Fig. 3. Examples of object detection using RPN proposals

Loss function: From [10], the loss function of the RPN is:

$$L\left(\{p_i\},\{t_i\}\right) = \frac{1}{N_{cls}}\sum_i L_{cls}\left(p_i, p_i^*\right)$$

$$+\lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}\left(t_i, t_i^*\right).$$

B. Sharing Features for RPN and Fast R-CNN

In the Faster R-CNN model, they use a 4-step training algorithm to learn shared features via alternating optimization. In the first step, the RPN is trained end-to-end by back-propagation and stochastic gradient descent (SGD). In the second step, they train a separate detection network by Fast R-CNN. In the third step, they use the detector network to initialize RPN training, and they let the two networks share convolutional layers. Finally, they keep the shared convolutional layers fixed, they fine-tune the unique layers of Fast R-CNN.

C. ResNet50

ResNet50 is a variant of the ResNet model, consisting of 48 convolution layers with 1 MaxPool and 1 Average Pool layer. It has 3.8×10^9 floating points operations. The ResNet50 is a widespread ResNet model.

Table I shows the architecture of the ResNet50. We can see

TABLE I
RESNET50 ARCHITECTURE

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

that [6], in a ResNet50 architecture, there are:

- A convolution with a kernel size of 7×7 and 64 different kernels all with a stride of size 2 there is 1 layer.
- Next there is a max pool with also a stride size of 2.
- In the next convolution there is a 1×1, 64 kernel following this a 3×3, 64 kernel and at last a 1×1, 256 kernel. These three layers are repeated in total 3 times so there are 9 layers in this step.
- Next we see a kernel of 1×1, 128 after that a kernel of 3×3, 128 and at last a kernel of 1×1, 512 this step was repeated 4 times so there are 12 layers in this step.
- After that there is a kernel of 1×1, 256 and two more kernels with 3×3, 256 and 1×1, 1024 and this is repeated 6 times there are a total of 18 layers.
- And then again a 1×1, 512 kernel with two more of 3×3, 512 and 1×1, 2048 and this was repeated 3 times there are a total of 9 layers.
- After that we do an average pool and end it with a fully connected layer containing 1000 nodes and at the end a softmax function so this gives us 1 layer.

There are $1 + 9 + 12 + 18 + 9 + 1 = 50$ layers in total.

D. Feature Pyramid Network

The Feature Pyramid Network (FPN) is used to address the problem of the large-scale span. The task of Mathematical Formulas Detection (MFD) contains a large number of extremely small formulas, which brings great challenges for us. As shown in figure 4 [14], for a single extremely small character formula, its short side is usually about 16 pixels. We discover that for any layer of FPN, the limit of the detector is 3 pixels, which means that if we use the default (3-7), the short side needs to be at least 24 pixels to be detected. It can be seen clearly that there are many small embedded formulas that do not satisfy this condition. As a result, we have changed the selection of the FPN level to (2-6), so that our model can overcome this challenge.

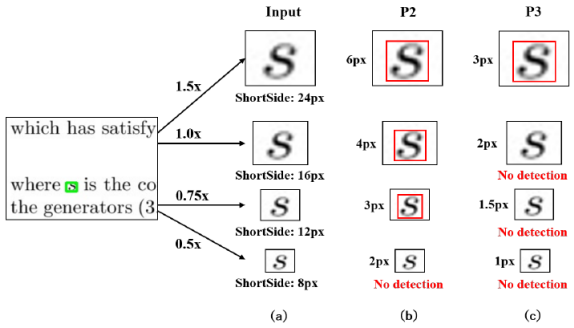

Fig. 4. The significance of FPN in the task of MFD. (a) Input size; (b): Corresponding size in P2, (c) Corresponding size in P3. Under some small cases, some positive samples will be missed

IV. EXPERIMENTS

In this section, we will describe the implementation of our mathematical formula detection system and dataset in detail.

A. Dataset

Our data is from the IBEM dataset. This originally comprises 600 documents, with 8273 pages in total. Those documents are parsed from mathematical papers, then each page is annotated with a bounding box of 2 types: isolated and embedded. The dataset is then split into various sets for ICDAR 2021 Competition on Mathematical Formula Detection, including Training, Test, and Validation sets.

Training

- Tr00: 4082 pages.
- Tr01: 760 pages.
- Tr10: 329 pages.

Validation

- Va00: 577 pages.
- Va01: 380 pages.

Test

- Ts00: 736 pages.
- Ts01: 380 pages.

- Ts10: 699 pages.
- Ts11: 329 pages.

Our experiment uses Tr01, Tr10, Ts01 for training, Va01 for validation, and Ts11 for testing with 2178 pages in total ($\sim 26.33\%$ of the original dataset), and an approximate ratio of $4.47 : 1.16 : 1$. The reason for this small subset is for the purpose of evaluating the ability of the model on small subsets, and the performance it gives (F1-score) through time (minutes).
The numbers of embedded and isolated bounding boxes in each set are:

- **Training:** 31074 embedded bounding boxes, 6617 bounding boxes.
- **Validation:** 6595 embedded bounding boxes, 1346 bounding boxes.
- **Test**: 5702 embedded bounding boxes, 1074 bounding boxes.

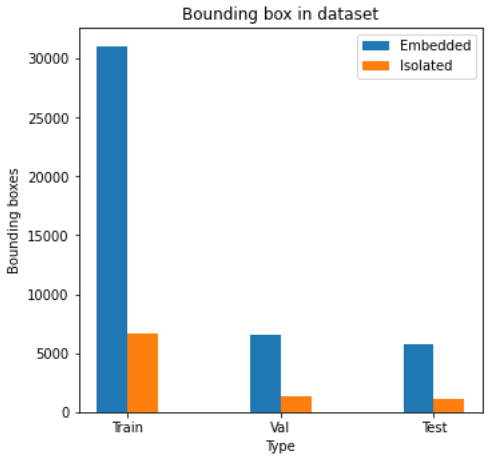

Fig. 5. The numbers of embedded and isolated bounding boxes

B. Implementation Details

Our baseline model is Faster R-CNN with ResNet50 as the backbone. We have trained on Kaggle with a 4-core CPU, 12GB RAM, and a NVIDIA Tesla P100 GPU [1]. The images are resized to 1447×2048 with the same ratio. The size of the region crops from the image is 1200×1120 to fit the limitation of the machine. They are also flipped and padded for data augmentation. For the feature aggregation, we use FPN (2-6). The loss function for the classifier is Cross-Entropy Loss and for the bouding box is L1 Loss. Test images are resized to 1583×2048 due to the distribution of the test dataset, flip augmentation is also applied. For post-processing, Non-Maximum Suppression (NMS) with 0.5 IoU threshold to remove redundant boxes. All models are trained based on the MMDetection toolbox and config given by Yuxiang Zhong. The optimizer for this baseline is Stochastic Gradient Descent (SGD) with a learning rate of 0.02.

[1] https://www.kaggle.com/docs/notebooks

C. Remarks

We have tested on 3 configs: Faster R-CNN with schedule 1x (12 epochs), Dynamic R-CNN with schedule 1x (12 epochs) to check if it is better than the faster one and Faster R-CNN with schedule 2x (24 epochs) to check if the model is underfitting with low epochs.
The results are given in the figures below.

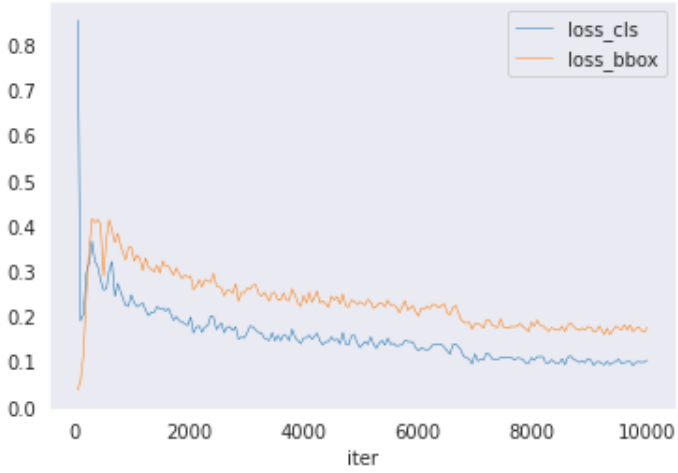

Fig. 6. Faster R-CNN with schedule 1x

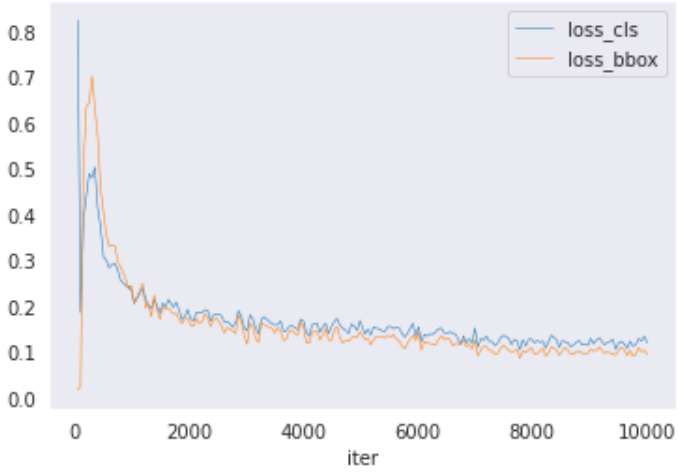

Fig. 7. Dynamic R-CNN with schedule 1x

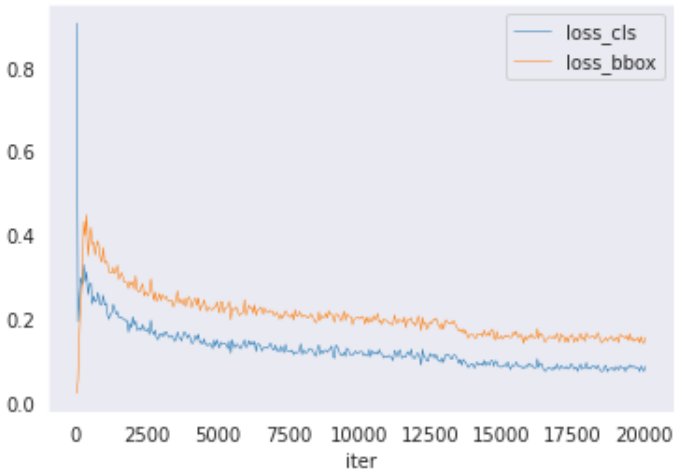

Fig. 8. Faster R-CNN with schedule 2x

The F1-score gained from the model is as follow.

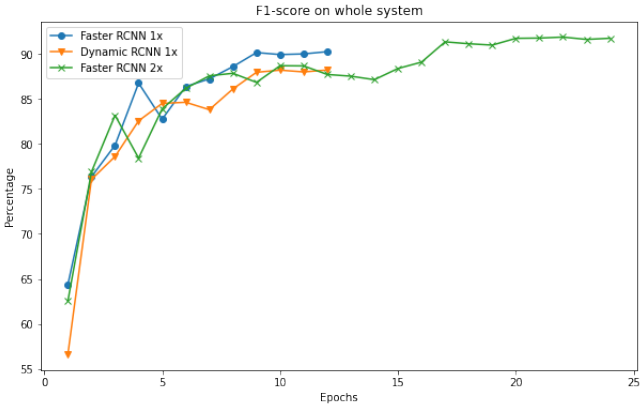

Fig. 9. F1-score on whole system

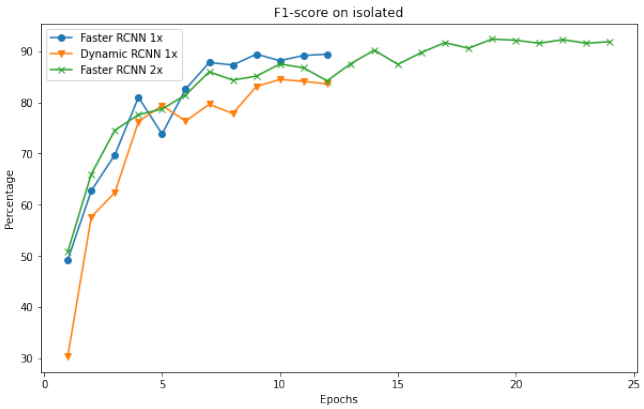

Fig. 10. F1-score with isolated bounding box

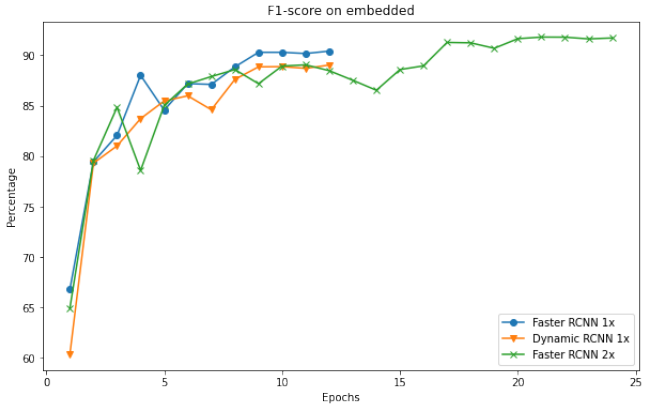

Fig. 11. F1-score with embedded bounding box

It can be seen from the graphs that on the whole system, with the same schedule 1x, the F1-scores given by the Faster R-CNN model are higher than the one by Dynamic R-CNN if we use the same number of epochs, except in the case of 5 epochs. The difference gets higher when we increase the number of epochs. Compared to the scores by Faster R-CNN with schedule 2x (24 epochs), although it gives a lower percentage when trained with a small number of epochs, the score becomes increasing to around 90%. Moreover, on the isolated bounding box, the Faster R-CNN model shows its benefit when compared with the number of Dynamic R-CNN, the F1-score of Faster R-CNN is nearly 90% while the one of Dynamic R-CNN reaches about 80% when they are both trained with 12 epochs. Considering the Faster R-CNN with schedule 2x, it gives the same F1-score with Dynamic R-CNN 1x at the point of 12 epochs, however, the score is about 90% at the point of 24 epochs. Besides that, it can be inferred from the figures of the embedded bounding box that with the same number of epochs (12 epochs), the Faster R-CNN model always provides better results than the Dynamic R-CNN, in spite of the fact that the difference is not large. When we increase the number of epochs to 24, we can observe that the F1-score of Faster R-CNN can reach the milestone of nearly 95%.

From the result given above, we can conclude that the Faster R-CNN model gives better F1-score than the Dynamic R-CNN model.

V. Conclusion and Future Work

Acknowledgment

References

[1] Chungkwong Chan. “Stroke Extraction for Offline Handwritten Mathematical Expression Recognition”. In: *IEEE Access* 8 (2020), pp. 61565–61575. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2984627.

[2] Kam Fai Chan and Dit Yan Yeung. “Mathematical expression recognition: A survey”. In: *International Journal on Document Analysis and Recognition* 3.1 (2000), pp. 3–15. ISSN: 14332833. DOI: 10.1007/PL00013549.

[3] Bidyut. B. Chaudhuri and Utpal Garain. “An Approach for Processing Mathematical Expressions in Printed Document”. In: *Document Analysis Systems*. 1998.

[4] Kenichi Iwatsuki et al. “Detecting In-Line Mathematical Expressions in Scientific Documents”. In: DocEng ’17. Valletta, Malta: Association for Computing Machinery, 2017, pp. 141–144. ISBN: 9781450346894. DOI: 10.1145/3103010.3121041. URL: https://doi.org/10.1145/3103010.3121041.

[5] Afef Kacem, Abdel Belaïd, and Mohamed Ben Ahmed. “Automatic Extraction of Printed Mathematical Formulas Using Fuzzy Logic and Propagation of Context”. In: *International Journal on Document Analysis and Recognition* 4 (Dec. 2001), pp. 97–108. DOI: 10.1007/s100320100064.

[6] Aakash Kaushik. *Understanding ResNet50 architecture*. URL: https://iq.opengenus.org/resnet50-architecture/.

[7] Xiaoyan Lin et al. “Mathematical formula identification in PDF documents”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR* (2011), pp. 1419–1423. ISSN: 15205363. DOI: 10.1109/ICDAR.2011.285.

[8] Suyu Ma et al. “Latexify Math: Mathematical Formula Markup Revision to Assist Collaborative Editing in Math Q & A Sites”. In: *Proceedings of the ACM on Human-Computer Interaction* 5.CSCW2 (2021). ISSN: 25730142. DOI: 10.1145/3479547.

[9] Parag Mali et al. “ScanSSD: Scanning Single Shot Detector for Mathematical Formulas in PDF Document Images”. In: (2020). URL: http://arxiv.org/abs/2003.08005.

[10] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149. ISSN: 01628828. DOI: 10.1109/TPAMI.2016.2577031.

[11] Amit Schester, Norah Borus, and William Bakst. “Converting Handwritten Mathematical Expressions into LaTeX”. In: (2017).

[12] Zelun Wang and Jyh Charn Liu. “Translating math formula images to LaTeXsequences using deep neural networks with sequence-level training”. In: *International Journal on Document Analysis and Recognition* 24.1-2 (2021), pp. 63–75. ISSN: 14332825. DOI: 10.1007/s10032-020-00360-2.

[13] Jianshu Zhang, Jun Du, and Lirong Dai. “Multi-Scale Attention with Dense Encoder for Handwritten Mathematical Expression Recognition”. In: *Proceedings - International Conference on Pattern Recognition* 2018-Augus (2018), pp. 2245–2250. ISSN: 10514651. DOI: 10.1109/ICPR.2018.8546031.

[14] Yuxiang Zhong et al. “1st Place Solution for ICDAR 2021 Competition on Mathematical Formula Detection”. In: (2021), pp. 1–8. URL: http://arxiv.org/abs/2107.05534.