Handwritten Math Formula Recognition

Keechang Choi
POSTECH
Dept. Mathematics
keechang@postech.ac.kr

Seunghyun Chae
POSTECH
Dept. Computer Science
shchae7@postech.ac.kr

Abstract

In this paper we present results of applying pre-existing methods designed for purposes of recognizing printed texts to handwritten math formula recognition. Experimental results show that despite being used for a different purpose, it is a viable approach. We combined scene text detection methods and image-to-markup language methods to recognize and predict multiple formulas in a given handwritten image. Existing methods are based on printed images, so we modified some parts to apply it on handwritten data. We used CROHME (Competition on Recognition of Online Handwritten Mathematical Expressions) 2013 dataset to generate synthesized formula scene images, and to train and test the image to markup language model. The proposed scene detection method achieves an F1-score of 0.70 on our own data set. The proposed image to markup language methods achieved 32.86% of accuracy on the generated test set with the optimal condition.

1. Introduction

Despite advances in digital media and the ability to write in digital format, some people still prefer writing on paper. There are many OCR techniques for characters, English sentences, but not many for math formulas, and especially not many for handwritten math formulas.

In this paper, we define math formula recognition as two problems of localizing handwritten formulas from an image and converting the localized formulas into markup language such as Latex. We divide recognizing formula problem into two subproblems, math formula detection and image-to-markup conversion.

For detection, we used a scene text detection approach. Purpose of it is to detect bounding box or bounding polygons of text in a various circumstance. We apply the basic idea of it to detect bounding boxes of math formulas.

For conversion to markup, we used a neural encoder-decoder model provided by Harvard NLP originally used for conversion from rendered images of LaTeX code to LaTeX markup. To train and evaluate the model, dataset of handwritten formulas of varying complexity

2. Problem: Math Formula Detection

2.1. Model

We used fully convolutional neural network. It is based on EAST [2] model. It consists of Feature extractor, Feature merging, and output layer. For feature extractor, pre-trained convolutional network on ImageNet dataset can be used. We first used VGG16 as a backbone feature extractor. Feature merging branch gradually merge extracted features, by unpooling last layer and concatenating it with current feature map. We used modified output layer of EAST model. It originally predicts score map, RBOX geometry, QUAD geometry. Instead, we predicted score map, side end vertex score map, side vertex geometry. It requires less complexity and it was efficient for long and complex texts.

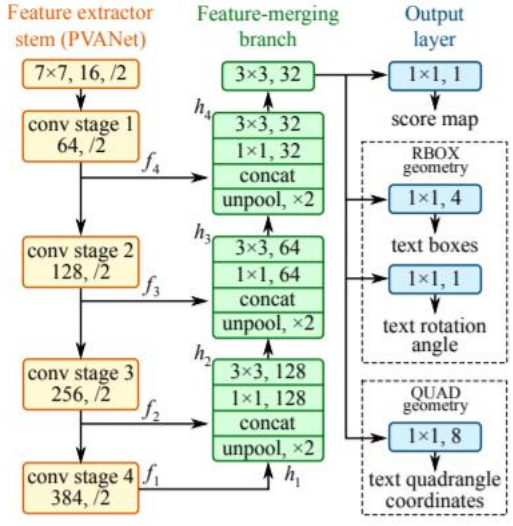

Figure 1. Network model of EAST [2]

2.2. Data

To detect handwritten math formulas, we generated our dataset by scattering each formula into background images. We could easily calculate the ground truth label of the dataset by writing the scattered position of each images. To get each formula image, we pre-processed CROHME (Competition on Recognition of Online Handwritten Mathematical Expressions) 2013 dataset. It consists of approximately 8000 handwritten math formula. inkml file (ink trace data of handwritten formulas, and the ground truth markup code for the formula). Ink traces are converted to png and images of formula are extracted.

Then 1~10 images among whole image set are randomly decided and distributed into background image files. We first generated and trained the model on 1000 (1024 x 1024) synthesized images in an empty background with larger space between formulas (dataset1). Next, we trained and tested on 1000 (1280 x 720) another synthesized images scattered densely on three different backgrounds: blackboard, illuminated paper, empty (dataset2).

2.3. Process

We first examined existing methods of scene text detection techniques. First, we tried to apply CRAFT (Character Region Awareness for Text Detection) [3] methods into our problem. It predicts confidence map of each character in various circumstances with affinity score map which describes connection of consecutive characters.

CRAFT was able to detect text of polygonal shape and we though that it would work well with math formulas. However, compared to most of common texts where all characters are connected linearly, math formulas are also connected vertically, and some expressions are nested inside others. Because of such properties, the complex connections and relations on math formulas were hard to label, and difficult to train by using CRAFT.

We found EAST model seems to capture text bounding boxes well in various circumstances without considering the orientation of connection, but not the math formulas. So, we decided to train this model with our dataset.

However, original EAST model showed low efficiency on finding bounding boxes of long math formula. We analyzed it to be because of the complexity of its last output layer. We decided to delete rotation prediction and to predict both sides ends of the formula. It decreased computational complexity and increased performance on long math formulas.

2.4. Implementation

Preprocess and labeling processes are the same the EAST model. We used VGG16 as the backbone of the feature extractor. For loss function, we adopted balanced cross entropy to predict score map of each point and side end scope map, introduced by the EAST paper. For side vertex coordinates, we used smoothed-L1 loss which was used for QUAD region in the original paper.

We first converted all the data images to have horizontal size 512 and labeled on each pixel with output scores. For dataset 1, training was done 12 epoch and dataset 2 in 40. In both cases, 10% of the training data was used for validation set.

2.5. Evaluation

For evaluation, we use the same strategy used for COCO-text. We determined the detected box is True Positive (TP) when its IoU (Intersection over Union) with the ground truth box is greater than the IoU threshold, and False Positive (FP) when it IoU is less than the IoU threshold. The bounding boxes that should be detected but not found are marked as False Negative (FN). Based on the number of TP, FP, FN, we calculated precision score, recall score, and F1-score. Then F1 score can estimate precision and accuracy of the results. Because all three scores are dependent on IoU threshold, we calculated 10 scores corresponding to IoU thresholds in a range of 0.5 to 0.95 increased by 0.05. By averaging those 10 scores, we calculated final averaged scores.

2.6. Result

2.6.1 Qualitative results.

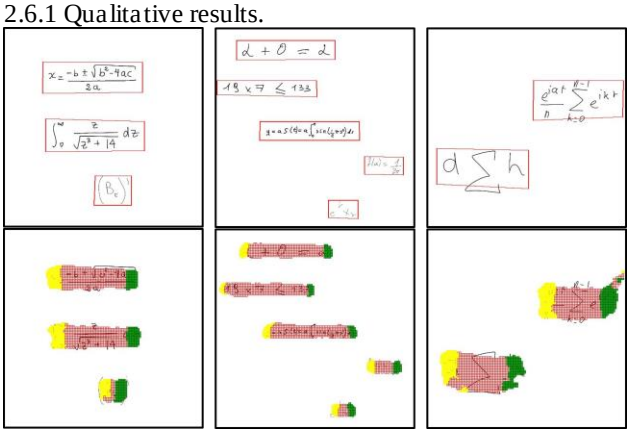

Figure 2. Qualitative results on the dataset1

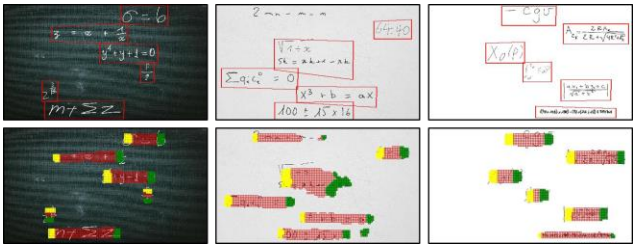

Figure 3. Qualitative results on the dataset2

Upper part of Figure 2 and 3. shows detected boxes, and

lower part of the figures shows visualized score map. Red part means score of the pixel is higher than the threshold. Yellow and green part means each side (left and right) end score of the pixel is higher than the threshold.

On first dataset, it seems to detect better on longer formula and the formulas whose characters are in a similar size. Fractions, subscripts, and superscripts are difficult to detect and be recognized as one formula. In other models, they were not contained in one bounding box, but were separated into several bounding boxes.

However, with the model trained by dataset 1, we could see that fractions, subscript, and superscript are detected and included in one bounding box.

On the second dataset, it detected well on empty backgrounds, but images on other background showed problems. Some formulas are not detected or detected in a merged form because its side end is not well detected.

2.6.2 Quantitative results

	Precision	Recall	F1 scoe
Dataset1	0.7059	0.7035	0.7047
Dataset2	0.5558	0.3763	0.4488

Table 1. Average precision, recall, F1 score of each dataset

On dataset1, when IoU threshold is less than 0.90, we got more than 62% well-matched bounding boxes. As compared with qualitative results, its performance was quite desirable. However, on dataset2, performance decreased rapidly.

2.7. Conclusion

We got the desirable result on the simpler data, dataset1. However, the data set that we generated is that of restricted circumstances. Model trained with dataset2, containing more generalized data, showed rapidly decreased performance. We concluded that we need to augment the data set rationally, and we may need to change models to increase the performance. Unlike detecting individual characters or linearly connected texts, detecting math formulas has several properties and difficulties. Despite needing modification to increase performance, we showed that bounding boxes of math formula can be detected by using existing scene text detection methods and by modifying them to be applied on formulas.

3. Problem: Image-to-Markup Conversion

We define the image-to-markup conversion problem as converting detected handwritten math formulas into LaTeX or MathML math formula markup.

3.1. Model

Our model is based on the encoder-decoder model [1]. The model is comprised of three components, convolutional network (CNN), row encoder, and decoder. CNN extracts visual features from the input image. Then RNN row encoder is applied to each row in the final feature map. The RNN decoder with a visual attention mechanism then uses the encoded features to produce the final outputs.

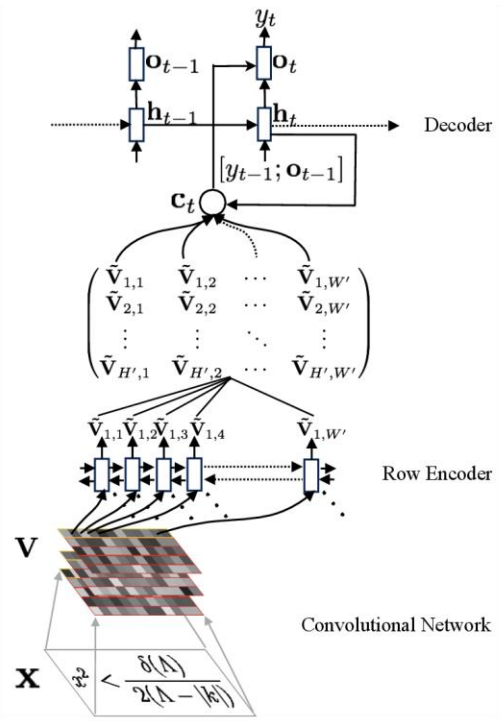

Figure 4. Network structure for image-to-markup conversion.

3.2. Data

We used two publicly available handwritten math formula datasets. One, dataset provided by CROHME (Competition on Recognition of Online Handwritten Mathematical Expressions) 2013 was used. Of the CROHME 2013 dataset, KAIST and MathBrush dataset were primarily used. KAIST dataset is comprised of approximately 1k images and MathBrush, approximately 3k.

Dataset	Included symbols				
	Numbers	Alphabet	Operator	Parenthesis	script
Level0	O	Eng	+, -, =	X	X
Level1	O	Eng, Greek	Linear connection	X	X
Level2	O	Eng, Greek	Linear connection	()	X
Level3	O	Eng, Greek	Nested operation	() { }	X
Level4	O	Eng, Greek	Nested operation	() { }	sub, super

Table 2. CROHME dataset categorization standards

The whole CROHME dataset is also used and divided according to complexity of the formulas. Table 1. Shows the standard in which the dataset was categorized.

As data were stored in InkML file format, ground truth math formulas were extracted, and ink traces were converted to png image files. The ground truth from INKML is in MathML format, different from that of LaTeX. However, as there seemed to be no significant difference between the two, no further preprocessing was done to convert MathML to LaTeX.

Two, Im2latex-100k-handwritting dataset provided by Harvard NLP was used. From 100k data, datasets of size 1k, 2k, 5k, denoted, Im2latex-1k, Im2latex-3k, Im2latex-5k are extracted for use.

Both dataset images are preprocessed to reduce computation and needed memory. Original images are cropped and padded with 8 pixels on all sides. Then are down sampled to half of their original sizes.

3.3. Process

OCR techniques existed for rendered images of math formula, but not many for handwritten math formulas. We decided to directly apply the model provided by [1]. Harvard NLP provided an official implementation of the paper in Torch. Training the official implementation in Google Colab proved to be a challenge, and to enable training, we decided to implement the paper in Pytorch.

There already existed an unofficial implementation of the paper in GitHub. However, the implementation was incomplete, and needed fixing. We created an evaluation code and fixed other miscellaneous bugs. Then trained using the CROHME 2013 and Im2latex-100k-handwriting dataset.

Training with Im2latex-100k-handwriting proved difficult with the small datasets providing no training at all. Deducing that it was because of the complexity of the math formulas along with small number of training data, we decided to focus on training with CROHME dataset which contained simple math formulas.

We decided to further separate the CROHME dataset according to math formulas' complexity. Training with separated CROHME dataset according to complexity, proved to be productive and resulted in successful conversion from image to markup.

3.4. Implementation

Given that Google Colab disconnect after a certain period of time, Time-consuming long-term training was impossible and hence the epoch was fixed at 25.

The initial learning rate is set to 3e-4, and we halve it once the validation average loss does not decrease. Weights for the model are saved after each epoch and weight with the minimum validation loss was chosen for evaluating.

3.5. Evaluation

The evaluating code stores the reference and result markup in separate files. Our core evaluation method is to compare them. Spatial arrangements and blank spaces in formula images in ground truth is denoted as <pad>. Sequences of <pad> has no formula meaning. Character by character comparison couldn't be done and comparisons were manually done. Markups with less than two different components were deemed to be match.

3.6. Results

Dataset	Evaluation Score
Im2latex-1k	-
Im2latex-3k	-
Im2latex-5k	-

Table 3. Im2latex dataset experiments.

The experimental results on Im2latex dataset can be seen in Table 2. These results compare evaluation results of several datasets of different sizes, 1k, 2k, 5k. For each size, the model was unable to convert every image to the correct markup and returned sequences of <unk>.

Dataset	Evaluation Score
KAIST	13.89 %
MathBrush	32.86 %

Table 4. CROHME 2013 KAIST, MathBrush dataset experiments.

$$\text{Result} : \cdots - \sum \text{f}$$
$$\text{Ref} \quad : \cdots - \sum \text{T}$$

$$\text{Result} : (\sum G)$$
$$\text{Ref} \quad : (S - 1_C)$$

Figure 5. Example output of model trained with MathBrush dataset. The two upper markups result in a match and the two lower markups result in a mismatch.

The experimental results on CROHME 2013 KAIST and MathBrush dataset can be seen in Table 3. The CROHME 2013 dataset is comprised of math formulas simpler than that of Im2latex-100k-handwritten dataset. That is why despite KAIST and Im2latex-1k dataset being near size, performances when dealing with KAIST dataset is better.

For the same reason performance of model trained on MathBrush dataset is better than one trained with Im2latex-3k.

Dataset	Evaluation Score
Level0	3.3 %
Level1	-
Level2	-
Level3	-
Level4	-

Table 5. CROHME level-0~4 dataset experiments

Table 4 shows results on varying degrees of math formula complexity. Unexpectedly, even with much simpler datasets, model was not able to convert image to markup. Only model trained with CROHME level-0 returned some semblance of a result and achieve 3.3% accuracy.

Model	Accuracy
TUAT	25.66 %
U NATES	26.06 %
UPV	37.22 %
IM2TEX	38.74 %
C2F-REINFORCE	33.87 %
WAP	46.55 %
IM2LATEX (MathBrush dataset trained)	32.86 %

Table 6. Handwritten image to markup experiments. Compares accuracy between different models used.

Comparison between best performance achieved by our method and other handwritten math formula image-to-markup results can be seen in Table 5. All other models were trained with the full CROHME 2013 and 2014 dataset.

3.7. Conclusion

As expected, when used for handwritten images, the model performed worse compared to when used for rendered images. Im2latex dataset contained complex formulas and it was difficult to train with small datasets. Compared to the amount of possible input's formula spatial arrangement and used symbol, the datasets were too small.

As for CHROME level-0~4 dataset, even though they were comprised of much simpler formulas, by extracting them from the whole CROHME dataset, the number of data was also too small. We deduce the reason for training with KAIST and MathBrush data being successful, as a balanced mix of formula complexity and number of data.

Despite being limited on computing resources, scale of dataset and training, we were able to directly use models for rendered-image-to-latex to handwritten images.

Even without additional preprocessing to approximate handwriting images to rendered images, it was shown it is possible. With increased scale of dataset and more computing resources, we expect the performance to increase and the model to work for Im2latex data as well.

3. Final Conclusion

We have presented that pre-existing models and method used for general OCR and rendered image-to-markup is a viable approach to math formula recognition. We separated the recognition task into two parts, detecting formula bounding boxes from an image containing multiple formulas and predicting a latex code from a formula image. We planned to develop one integrated model when we managed to solve the two subproblems, but faced technical difficulties, and were unable to do so.

One of the most serious difficulty was preparing the dataset. We generated handwritten math formula data by processing CROHME 2013 data set and labeled it with position of bounding boxes and ground truth latex code. Taking into account that rendered image-to-markup approach by Harvard NLP used im2latex dataset with 100k images, we assumed getting generally working result with the CROHME dataset which is roughly less than 10k would be difficult. So, we divided dataset with various criterion, and trained and tested on them to get the results.

We used Google Colab to train the model due to the absence of computing resources. Since our approach required computationally heavy tasks, training the model took extremely long and was the most difficult part of the project. If we can train the model on bigger data set with more powerful computing resource, we expect our divided data set would produce reliable results and perform well enough to be used on general handwritten math formula images.

References

[1] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M. Rush. Image-to-markup generation with coarse-to-fine attention. ICML, 2017

[2] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. EAST: An Efficient and Accurate Scene Text Detector. CVPR, 2017

[3] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee. Character region awareness for text detection. CVPR, 2019.