Recognition of Hand Written Mathematical Expression using Scale Augmentation and Drop Attention

Anish Ghosh,Bivek Panthi,Sishir Sunar

Abstract

This is a review paper. We start by explaining about how handwritten mathematical expressions have a unstable scale.Then we show how augmented layers are used for scaling those mathematical expression.We continue by explaining how an attention based encoder-decoder is used for extracting features and generating predictions.The drop attention is used when the attention distribution of the decoder is not precise.This method achieves better performance than any other existing method.

1 Introduction

In this section we will discuss the problem we are addressing and how we are going to tackle the problem. So we are trying to convert Hand Written Mathematical Expressions to .tex format. One of the main problem that arises when converting Hand Written Mathematical Expression is scaling the image of the Mathematical Expression.

The main thing in this process of converting the Hand Written Mathematical Expression to .tex is the **encoder-decoder network**.We enhance this network by using **Scale Augmentation** at the beginning and **Drop Attention** at the end to enchance the accuracy which increases quality of this output.

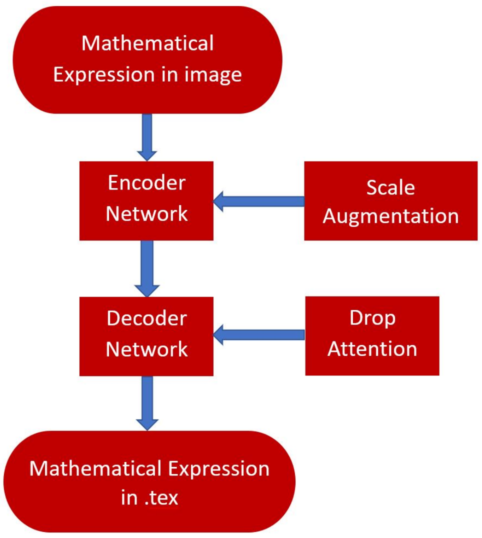
Sketch of the Paper

- Encoder-Decoder Network
- Scale Augmentation
- Drop Attention
- Experiment and Implementation
- Conclusion
- References

Sketch of the process

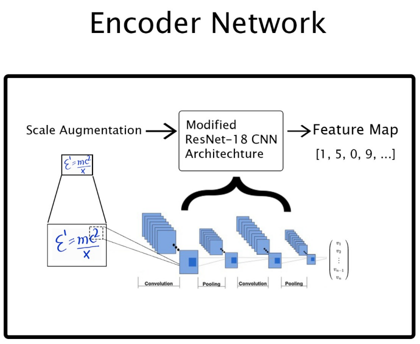
- Scale Augmentation
- Encoder-Decoder network
- Drop Attention

As we can see the Scale Augmentation is used at the begining before the Encoder-Decoder Network but it is used to enhance the Encoder-Decoder Network. So we need to understand what is the Encoder-Decoder Network then we can move on with Scale Augmentation.


Mathematical Expression in image
Encoder Network
Scale Augmentation
Decoder Network
Drop Attention
Mathematical Expression in .tex

2 Encoder-Decoder Network

The encoder is build by modifying the CNN architechture ResNet-18 and we set the stride of all convolutional layers in ResNet-18 to 1 in order to extract the more precise features and avoid the neglecting features.The other settings are same as in [1] .Max pooling layers are used for down-sampling and dropout layers are used to lessen the network over fitting.The detailed network configuration is shown in the table I.



Variables and their Meanings

- c - Number of filters
- k - Kernel number
- s - Kernel Size
- pa - Stride Padding
- n - Block Number
- p - Dropout Probability

Table I(detailed Network Configuration)

Layer/Block	*Setting*
Convolution	$c = 64, k = 3 \times 3, s = 1, pa = 1$
Maxpooling	$k = 2 \times 2, s = 2$
Batchnorm	$-$
ReLU	$-$
Buildingblock	c = 64, n = 2
Maxpooling	$k = 2 \times 2, s = 2$
Dropout	p = 0.1
Buildingblock	c = 128, n = 2
Maxpooling	$k = 2 \times 2, s = 2$
Dropout	p = 0.2
Buildingblock	c = 256, n = 2
Maxpooling	$k = 2 \times 2, s = 2$
Dropout	p = 0.3
Buildingblock	c = 512, n = 2
Maxpooling	$k = 2 \times 2, s = 2$
Dropout	p = 0.3

The attention-based decoder is based on RNN and iteratively generates the target sequence Y from features F.

Definition(Softmax Function) :The softmax function is a function that turns a vector of K real values into a vector of K real values that sum to 1. The input values can be positive, negative, zero, or greater than one, but the softmax transforms them into values between 0 and 1, so that they can be interpreted as probabilities.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

- $\sigma = softmax$
- $\vec{z}$ = input vector
- e^{z_i} = standard exponential function for input vector
- K = number of classes in the multi-class classifier
- e^{z_j} = standard exponential function for output vector
- e^{z_j} = standard exponential function for output vector

Variables and their Meanings

- t - time step
- y_t - A member of the target sequence Y at time step t
- c_t - context at time step t
- h_t - hidden state at time step t
- H - Height of the feature F
- W - Width of the feature F
- L - the size of the features $L = H \cdot W$
- $\alpha_{t,l}$ - weight of the l^{th} features of F at time step t
- f_l - the l^{th} features of F
- tanh - chosen activation function
- $e_{t,l}$ - attention weight of the l^{th} features of F at time step t
- W^e, W^h, W^f, W^q, W^s - trainable Parameters

- q_l - position embedding of the l^{th} features of F
- s_l - coverage features of the l^{th} features of F
- $E^{p,h}$ - absolute position embedding in horizontal direction
- $E^{p,v}$ - absolute position embedding in vertical direction

At t the probability of generating y_t from the RNN output is shown in the equation 1 where $g(\cdot)$ is a linear function followed by a softmax function.

$$p\left(y_t\right)=g\left(c_t, h_t\right) \tag{1}$$

Context c_t is computed as a weighted sum of features in equation 2.

$$c_t=\sum_{l=0}^{L-1} \alpha_{t,l} \cdot f_l \tag{2}$$

Similar to human eyes the attention based decoder concentrates on only a subset of features at a time.Trained using back propagation algorithm using gradient descent,the decoder can be given the ability to determine which features are important for generating the word y_t.We choose a linear function with the activation function tanh.We compute the attention weights and normalization using softmax function as shown in equation 3 and equation 4 respectively.

$$e_{t,l}=W^e \tanh\left(W^h h_t+W^f f_l+W^q q_l+W^s s_l\right) \tag{3}$$

$$\alpha_{t,l}=\frac{\exp\left(e_{t,l}\right)}{\sum_{i=1}^{L} \exp\left(e_{t,i}\right)} \tag{4}$$

Refer to [2] when calculating the attention weights, position embeddings make the decoder position sensetive.The calculation of position embedding is shown in equation 5.

$$q_l=E^{p,h}(\lfloor l/H \rfloor)+E^{p,v}(l \bmod H) \tag{5}$$

Referring to [3] the coverage features are used to lessen over-attention and under-attention.$s_l$is given as the sum of past attention weights in equation 6.

$$s_l=\sum_{\tau=0}^{t-1} \alpha_{\tau,l} \tag{6}$$

The hidden state is calculated using a RNN with long short- term memory (LSTM) cells. At time step t, the RNN input is the word embedding at t1 concatenated with context at t1, which is a weighted sum of CNN features F with position embeddings at last time step is represented by equation 8.

$$h_t=RNN\left(\left[E^d\left(y_{t-1}\right), c_{t-1}^{\prime}\right], h_{t-1}\right) \tag{7}$$

$$c_t^{\prime}=\sum_{l=1}^{L} \alpha_{t,l} \cdot \left(f_l+q_l\right) \tag{8}$$

3 Scale Augmentation

Since in a Mathematical Expression the expression consists of various symbols some smaller than the others which increases the recognition difficulty. So we apply augmentation to the Mathematical Expression according to equation 9 to improve the recognition.

$$\left\{\begin{array}{l} x^{\prime}=k x \\ y^{\prime}=k y \end{array}\right. \tag{9}$$

k is the scaling factor and we keep the aspect ratio unchanged.The encoder-decoder network is trained to adapt to Mathematical Expression of various scales in order to generate correct predictions.

4 Drop Attention

The normalised attention weight α has a great impact on recognition performance so when the attention neglects the key features, the model will generate an incorrect prediction which will result in worse performance.From [4] a different idea can be proposed that is the Drop Attention Module in the decoder.First, we randomly supress the features where the α is the highest.Then we randomly abandon spots according to equation 10.

Variables and their meanings

- γ - Supress Factor
- r_p, r_s - Random Variables that obey Bernoulli Distribution
- f_l' - features (replaces f_l in equation 2 and 8 in training phase to predict the correct symbol when the attention is imprecise)

$$f_l' = \begin{cases} \gamma \cdot f_l, & if\ l = argmax\left(\alpha_l\right) and r_p = 0 \\ 0, & if\ l \neq argmax\left(\alpha_l\right) and r_s = 0 \\ f_l, & otherwise \end{cases} \tag{10}$$

5 Experiment and Implementation

The proposed model was implemented using PyTorch and optimized on Nvidia TITAN X_p Graphical Processing Unit.For simultaneous calculations to be carried out at the same time the batch size was set to 8.The Mathematical Expression images underwent scale augmentation in the training phase and were zero padded to a fixed size 256×1024.Few of the mathematical expression were downsized because they were larger than the size before zero padding.The scaling factor(equation 9) k was set randomly within the bound $[0.5, 2]$ during each training iteration.Using CNN the features were extracted and fed to the attention based decoder with configurations according to **Table I**.The suppress factor γ and the Bernoulli probabilities r_p, r_s(equation 10) during the training phase were set to $0.1, 0.8\ and\ 0.4$ respectively.The cross-entropy loss was calculated between the generated symbols and the ground truth.Adam optimizer was used to train the model, and the learning rate was 0.0001.

5.1 Effect of Scale Augmentation

Comparative experiments for processing Mathematical Expression images were performed without applying the drop attention module.The three methods used are as follows :

- Normalization to a fixed height of 256 without changing the aspect ratios.
- Zero Padding to a fixed size.
- Scale Augmentation

The expression recognition rate for the above mentioned methods is shown in **Table II**.

(Expression Recognition Rate % of different processing methods for Mathematical Expression Images)

$Method$	$CROHME2014$	$CROHME2016$
$Normalized\ to\ fixed\ height$	48.78	49.26
$Zero-padded\ to\ fixed\ size$	50.00	47.95
$Scale\ Augmentation$	55.17	52.48

Table II[5]

As it can be seen from **Table II**,the results with a fixed height normalization are slightly better than zero padding on the CHROME 2016 data set but worse on CHROME 2014.Whereas Scale Augmentation outperforms the other two methods which shows that Scale Augmentation is more effective than the other two methods.The model trained with Scale Augmentation could extract particular features from various scales which helps us to get a more accurate prediction.Hence Scale Augmentation is applied in the subsequent experiments.

5.2 Effect of Drop Attention

Comparative experiments were performed with and without the Drop Attention to verify its effectiveness.The results are shown in **Table III**.

(Expression Recognition Rate % of processing methods with or without Drop Attention)

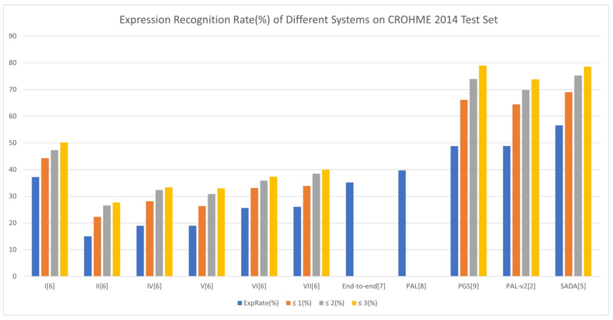
Drop Attention	*CROHME*2014	*CROHME*2016
without	55.17	52.48
with	56.59	54.58

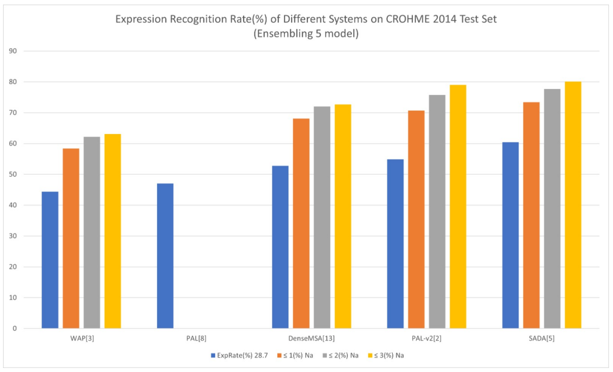
Table III[5]

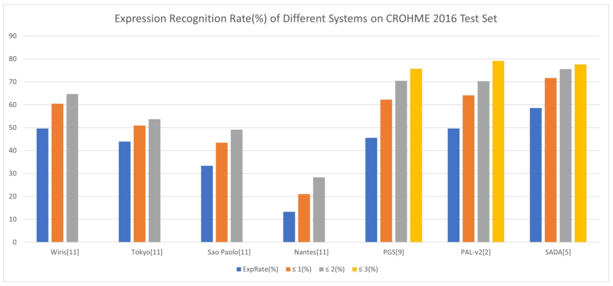
As shown above in **Table III** when the experiment was performed with the Drop Attention Module, the accuracy improves by 1.42% on CROHME 2016 and 2.1% on CROHME 2014.The drop attention module trains model generating better predictions when attention neglects the key features.The Drop Attention model was implemented during the training phase of subsequent experiments to improve the performance of the proposed model.

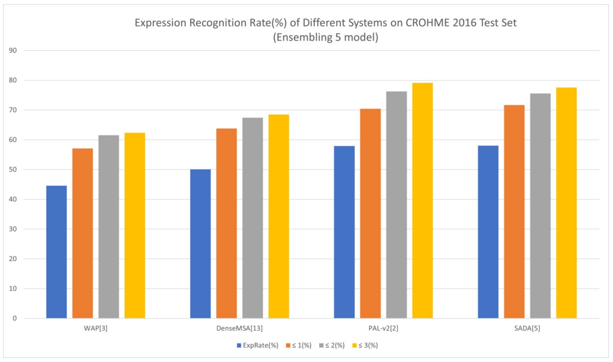
5.3 Comparison with proposed methods

The graphs below shows the expression recognition rate(ExpRate) taken out using recent offline attention based Hand Written Mathematical Expression models on CROHME 2014 and CROHME 2016 datasets. Among all the Hand Written Mathematical Expression models, the Scale Augmentation and Drop Attention model (SADA[5]) achieves the best performing results.









6 Conclusion

In this paper Scale Augmentation method is used to solve the problem of recognition of Mathematical Expression of various scales. The problem is caused due to its two dimensional structure.A new Drop Attention module is used to enhance the training of the model which helps to generate better results when the attention is imprecise.The experimental results indicated that these two technologies assist our method achieving state-of-the-art performance, compared with the previous methods without an ensemble.

7 References

1. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVRP), pp. 770–778, 2016

2. J.-W. Wu, F. Yin, Y.-M. Zhang, X.-Y. Zhang, and C.-L. Liu, "Handwrit- ten mathematical expression recognition via paired adversarial learning," International Journal of Computer Vision, pp. 1–16, 2020.

3. J. Zhang, J. Du, S. Zhang, D. Liu, Y. Hu, J. Hu, S. Wei, and L. Dai, "Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition," Pattern Recognition, vol. 71, pp. 196–206, 2017.

4. G. Sun, H. Cholakkal, S. Khan, F. S. Khan, and L. Shao, "Fine-grained recognition: Accounting for subtle differences between similar classes," arXiv preprint arXiv:1912.06842, 2019.

5. Z. Li, L. Jin, S. Lai and Y. Zhu, "Improving Attention-Based Handwritten Mathematical Expression Recognition with Scale Augmentation and Drop Attention," 2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR), Dortmund, Germany, 2020, pp. 175-180, doi: 10.1109/ICFHR2020.2020.00041.

6. H. Mouchere, C. Viard-Gaudin, R. Zanibbi, and U. Garain, "Icfhr 2014 competition on recognition of on-line handwritten mathematical expressions (crohme 2014)," in 2014 14th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 791–796, IEEE, 2014.

7. A. D. Le and M. Nakagawa, "Training an end-to-end system for handwritten mathematical expression recognition by generated patterns," in 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, pp. 1056–1061, IEEE, 2017.

8. J.-W. Wu, F. Yin, Y.-M. Zhang, X.-Y. Zhang, and C.-L. Liu, "Image- to-markup generation via paired adversarial learning," in Joint Euro- pean Conference on Machine Learning and Knowledge Discovery in Databases, pp. 18–34, Springer, 2018.

9. A.D.Le,B.Indurkhya,andM.Nakagawa,"Patterngenerationstrategies for improving recognition of handwritten mathematical expressions," Pattern Recognition Letters, vol. 128, pp. 255–262, 2019.

10. Z. Xie, Z. Sun, L. Jin, H. Ni, and T. Lyons, "Learning spatial-semantic context with fully convolutional recurrent network for online handwritten chinese text recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 8, pp. 1903–1917, 2017.

11. H. Mouche‘re, C. Viard-Gaudin, R. Zanibbi, and U. Garain, “Icfhr2016 crohme: Competition on recognition of online handwritten mathematical expressions,” in 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 607–612, IEEE, 2016.

12. Y.Deng,A.Kanervisto,andA.M.Rush,“Whatyougetiswhatyousee: A visual markup decompiler,” arXiv preprint arXiv:1609.04938, vol. 10, pp. 32–37, 2016.

13. J. Zhang, J. Du, and L. Dai, “Multi-scale attention with dense encoder for handwritten mathematical expression recognition,” in 2018 24th International Conference on Pattern Recognition (ICPR), pp. 2245– 2250, IEEE, 2018.