Optical Character Recognition For Mathematical Equations

Rex Stockham
Massachusetts Institute of Technology
rexstock

Abstract

This paper describes an approach for reading off-line handwritten mathematical equations and converting them to LaTeX. This approach consists of four steps: image processing, segmentation, classification, and merging individual characters into a properly formatted LaTeX output. We utilize a Convolutional Neural Network to provide accurate classifications for 22 classes of symbols from the MNIST and CROHME datasets, with the ability to extrapolate four additional symbols from spatial relationships between the characters during the merging step.

1. Introduction

LaTeX is a high quality typesetting system which includes features designed for the production of technical and scientific documentation.[1] It is widely utilized in academia and is capable of producing properly formatted mathematical equations. The recognition of mathematical equations using Optical Character Recognition (OCR) can be classified as either on-line or off-line, [5] where on-line OCR processes stroke data produced as characters are being written, and off-line OCR processes a static document. The problem of recognizing handwritten mathematical equations has gained importance in recent years [5] due to the interest in converting written documents into a digital form.

For this paper we are interested in the off-line detection of handwritten mathematical equations and their conversion to LaTeX. As a research problem, the recognition of mathematical equations is different from typical OCR text recognition problems as it poses the challenge of not only identifying and classifying characters, but also performing a structural analysis of the equation. [11] When there are errors detecting characters in the input image or during classification, those errors are propagated forwards to the structural analysis.

In this paper, we propose a four step system for the conversion of off-line mathematical equations into LaTeX. The system analyzes the input image using a bottom-up approach to detect individual characters. Characters are classified into one of 22 classes using a Convolutional Neural Network (CNN) trained on a subset of the MNIST and CROHME datasets. After classification, the system utilizes a rule based system based on spatial information to recursively format the equation into LaTeX. Finally, we assess the capabilities of our proposed model by looking at the classification accuracy of the CNN and the results of image segmentation.

2. Related Work

This project has benefited from a substantial amount of research on text detection and OCR for mathematical symbols and equations. For image processing, Kamel *et al* [4] proposes an adaptive global thresholding technique to extract a binary image from an original grayscale image. This is intended to remove noise in the image. The work by Wu *et al* [9] notes that applying a series of Gaussian filters to the image before thresholding increases it's performance, as text generally has a stronger response to the filters.

When training a classifier to detect mathematical equations, nearly all approaches rely on a subset of the MNIST and CROHME datasets, or a combination of both. [2][5][6][11]. As our system should be able to classify both digits and mathematical symbols, we will be using a combination of the MNIST and CROHME datasets. Nazemi *et al* [6] lays out a process for the combination, normalization, and augmentation of these two datasets for use in training a classifier.

Several different approaches to the task of classification have been proposed, with varying degrees of success. Hamid *et al* [2] examines the accuracy of Support Vector Machines and K-Nearest Neighbors for classification. Additionally, Wu *et al* [10] lays out an approach that relies on a Self-Organizing Map (SOM) for classification, but it is very computationally expensive to train. While these models all had a high classification accuracy on the MNIST dataset, we felt that a Convolutional Neural Network would be more effective on the modified dataset we will be using given the results described by Nazemi *et al*.[6] Nazemi *et al* tests the effectiveness of two different CNN architectures, and we base the architecture of our classifier off of the techniques they describe.

Another method used in the classification of characters is the hierarchical approach laid out by Ranger *et al* [7]. This relies on contextual data such as the size and color of symbols and their spatial relationships with other symbols in the equation. This proved to be an adaptive and efficient approach, and it influenced our construction of a rule based system to merge the outputs of our classifier into a single LaTeX equation.

3. Approach

We approached this system iteratively, designing and testing each of the four sub-sections before moving on to the next. The four sections of the system are: image processing, segmentation, classification, and merging.

3.1. Image Processing

When given an input image of any size, the first step is to process the image so that individual characters can be recognized. Following the work in [4][9] the image is converted to grayscale, ensuring that it has only one channel. We then apply a 5x5 Gaussian filter to the image: 1

$$G(x,y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

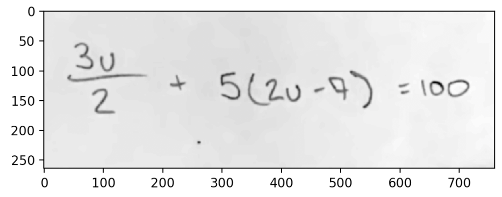

Figure 1. Grayscale Gaussian Blur

Next, a binary threshold is applied to the image. [4] If $f(x,y)$ represents the grayscale image of size MxN with pixel values in the range [0, 255] where $1 \leq x \leq M$ and $1 \leq y \leq N$, then $t(x,y)$ the image with a binary threshold T is:

$$t(x,y) = \begin{cases} 1 & f(x,y) \leq T \\ 0 & otherwise \end{cases}$$

The threshold value T is set adaptively by comparing the value of a pixel to the mean of the pixel values in an n x n neighborhood around it. The results of thresholding 2 will enable us to detect individual characters. In its current state, this system is limited to detecting chracters in images with white backgrounds.

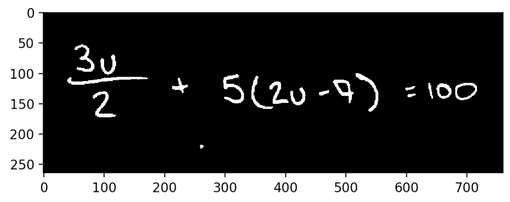

Figure 2. Adaptive Thresholding

3.2. Segmentation

For our system to work, we must perform character level segmentation in order to detect the individual symbols in the input image, preprocess them, and pass them to the classifier. To perform segmentation, our model utilizes the border following algorithm outlined by Suzuki [8]. For this step we provide as an input the processed image which is made up of 0's and 1's.2 The algorithm is limited to binary images and rectangular bounding boxes, which are appropriate limitations for our system. At a high level, border following works by deriving a sequence of (x, y) pixel coordinates from the border between a connected component of 1-pixels and a connected component of 0-pixels. A bounding box is then placed around the coordinates such that individual characters are segmented. The results of this process can be seen in Figure 3. 3

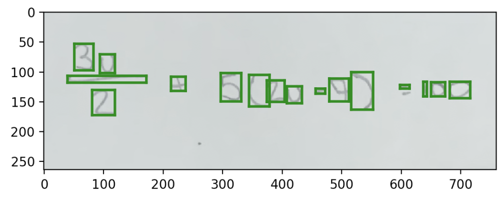

Figure 3. Bounding Boxes

Suzuki's algorithm [8] generates a heirarchical ordering of bounding boxes such that we can determine if one bounding box lies within another, such as might occur with the inner and outer borders of the number zero. Our system then removes all bounding boxes that lie inside of another and computes the mean and standard deviation for the area of the bounding boxes. Given a mean area μ, a standard deviation of σ, and a bounding box with width area a, the bounding box is removed if $a \leq \mu - 1.5\sigma$. We found that this effectively removed bounding boxes that resulted from noise in the image.

Next, we iteratively crop the thresholded image at the locations specified by the bounding boxes to retrieve binary images of each character in the mathematical equation.4 Our CNN classifier was trained on 224 x 224 pixel images, with a mean pixel value of .1942 and a standard deviation of .2202. Before these images are passed to the CNN for classification, they are padded such that resizing them to the necessary 224 x 224 pixels will maintain their original aspect ratio. Finally, they are then resized and normalized using the mean and standard deviation of the dataset. [6]

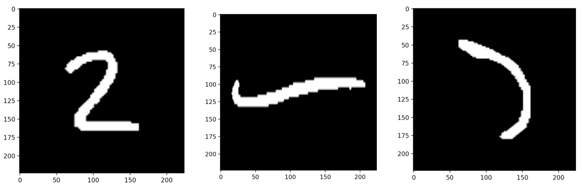

Figure 4. Cropped Characters

3.3. Classification

Dataset

The dataset used to train our CNN classifier consists of 22 classes: [0-9], +, -, $<$, $>$, =, *,), (, u, v, $\sum$, and ∞. In total, each class had 6000 images. The dataset is the reuslt of combining the MNIST dataset of handwritten digits and a subset of the CROHME dataset of mathematical equations. The images in the CROHME dataset were converted from InkML files to binary grayscale .png files to match the formatting of the MNIST dataset. Images from both datasets were resized to 224 x 224, and the mean and standard deviation of the dataset was computed and used for normalization. We performed a stratified test-train split of the dataset, setting aside 20% of the images for testing and maintaing the distribution of classes over both the test and training sets.

Augmentation

Before being passed to the CNN, images were randomly augmented, in order to prevent over-fitting. When performing augmentation on this dataset, we avoided performing horizontal or vertical flips of the image so as not to change the symbol meaning. [6]. For example, flipping the character $<$ horizontally makes it become $>$. The augmentations used were random rotations about the center of the image by up to ± 10 degrees, and random pixel erasing.

Architecture

We implemented our CNN using the Resnet18 architecture described by He *et al* [3]. A description of each of the layers in this architecture is shown in Figure 5.5 ResNet18 is a residual neural network, and utilizes skip connections to add the output of the previous layer to the layer ahead alloing us to train a deeper network without degrading performance.[3] In addition to the ResNet18 architecture, we experimented with SqueezeNet, VGG, K-Nearest Neighbors, and a random forest classifier. All of

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

Figure 5. Resnet18 Architecture

these models were found to have lower classification accuracy than ResNet18 as shown in Table 1.1 In order to reduce overfitting of the model, we added a dropout layer between the average pooling and fully connected layers. The dropout layer had a probability $p = .5$. As we are training on 22 classes, there are 22 nodes in the fully connected layer which has a softmax activation function to predict class probabilities. The softmax function is defined as: $softmax(x_i) = \frac{e^{x_i}}{\sum_{i=1}^{N} e^{x_i}}$

Optimization and Loss

The final model was trained using Stochastic Gradient Descent (SGD) with a batch size of 128 images. The learning rate was initialized to .001, and we used the Adam optimization algorithm. Given the softmax activation function of the networks fully connected layer, we chose to use cross entropy loss. The network was trained for a total of 100 epochs, and the results of training can be seen in Figure 7 7 and Figure 8. 8

3.4. Merging

After all of the identified characters have been classified by the CNN, the structure of the equation must be analysed to generate properly formatted LaTeX. To do this, we rely on a recursive rule based system which compares the spatial relationships of the characters as defined by their bounding boxes. Using the Euclidian distance $d(p, q) = \sqrt{(p_2 - q_2)^2 + (p_1 - q_1)^2}$ between the center of bounding box p and bounding box q given threshold T, we calculate the result of the binary relationship $d(p, q) < T$. While our CNN was trained on 22 classes, the method defined above allows us to add four more: $\leq$, $\geq$, $\div$, and $=$. For instance, if images p and q are classified as the - symbol and satisfy $d(p, q) < T$, they are interpreted as $=$. The same logic holds for $\leq$ and $\geq$. To detect the $\div$ symbol, we look for images that are classified as - that have at least image above and below it with a center that falls between the left and right edges of its bounding box.

After the classification of each symbol has been rectified using the process described above, we recursively split the equation in half horizontally, splitting vertically only when the image we are splitting on is classified as $\div$. Using a dictionary mapping classifier outputs to their proper LaTeX formatting, we recursively construct the full equation. 6

```
$\frac{3u}{2}+5(2u-7)=100$
```

Figure 6. Formatted LaTeX Equation

4. Experimental Results

Due to the rule based nature of the merging step, the overall accuracy of our system is reliant upon the accuracy of segmentation and classification. Errors in either of these steps are propagated forwards throughout the rest of the system. As the accuracy of character segmentation is largely subjective, [9] we will focus our analysis on the classification accuracy of our CNN. We experimented with a variety of classifiers and hyperparameter values, the results of which can be seen in Table 1.1 As is evident, the three CNN models were far superior to K-Nearest Neighbors and Random Forest classifiers for our dataset. While SqueezeNet had the highest training accuracy with 89.4%, it is evident that it severely overfit the training data.

Model	Train Acc.	Test Acc.
ResNet18	73.6%	70.2%
Squeezenet	89.4%	55.8%
VGG	64.5%	60.6%
KNN	N/A	32.4%
Random Forest	N/A	48.2%

Table 1. Classifier Accuracy

Ultimately, the ResNet18 architecture produced the highest classification accuracy on the test set at 70.2%. However, these are sub-par results as compared to what is reported in the literature cited. [2] [6] [11] As we can see in Figure 7, 7 the classification accuracy of the training and testing datasets is very close, indicating that the model is not overfitting. It is my belief that this diminished classification accuracy is the result of poor normalization and standardization techniques when combining the CROHME and MNIST datasets. Each of our classifiers saw a substantially increased performance when trained on the MNIST or CROHME datasets by themselves. The ResNet18 architecture was able to achieve slight improvements to classification for the full dataset when it was first trained exclusively on the MNIST dataset. These weights were then loaded into the model and it was trained again with a on the full dataset with a substantially decreased learning rate of .0001.

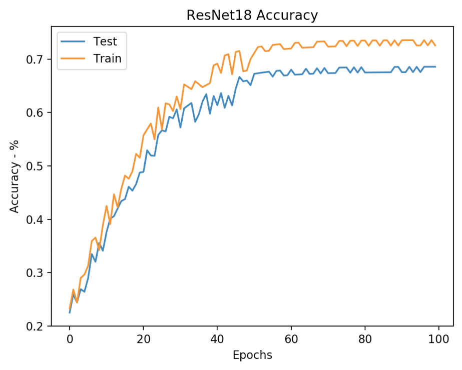

Figure 7. ResNet18 Accuracy

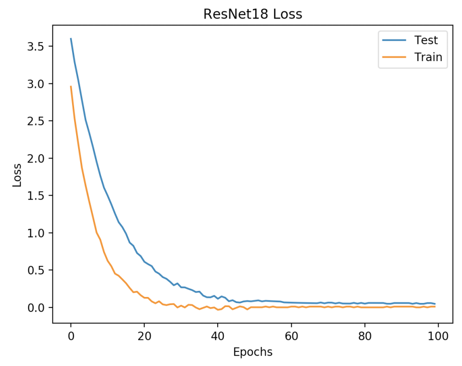

Figure 8. ResNet18 Loss

5. Conclusion

In this paper we introduce an approach for reading offline handwritten mathematical equations and converting them to LaTeX. It is capable of properly segmenting the input image into individual characters and classifies these characters at with an accuracy of 70.2%. The rule based structural analysis is able to rectify errors made by the classifier and output a properly formatted LaTeX equation.

In future work, we plan to improve the system by performing further testing of classification models and tuning the hyperparameters of our existing ResNet18 model. We will also work to better standardize the combination of MNIST and CROHME datasets so that there is less of a drop in performance when our classifier is trained on both of them. Lastly, we will increase the number of classes the model is able to classify.

References

[1] The latex project. http://https://www.latex-project.org/. Accessed: 2019-11-28. 1

[2] N. A. Hamid and N. N. A. Sjarif. Handwritten recognition using svm, knn and neural network. *ArXiv*, abs/1702.00723, 2017. 1, 4

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. 3

[4] M. S. Kamel and A. Zhao. Binary character/graphics image extraction: a new technique and six evaluation aspects. *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol. III. Conference C: Image, Speech and Signal Analysis,*, pages 113–116, 1992. 1, 2

[5] H. Mouchère. Icfhr 2016 competition on recognition of on-line handwritten mathematical expressions. http://tc11.cvc.uab.es/datasets/ICFHR-CROHME-2016_1, 2016. 1

[6] A. Nazemi, N. Tavakolian, D. Fitzpatrick, C. a Fernando, and C. Y. Suen. Offline handwritten mathematical symbol recognition utilising deep learning. *ArXiv*, abs/1910.07395, 2019. 1, 3, 4

[7] J. Ranger, F. Wang, and H. Cheng. Optical character recognition of printed mathematical symbols using a hierarchical classifier, 2012. Copyright - Copyright The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp) 2012; Last updated - 2014-01-10. 2

[8] S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30:32–46, 1985. 2

[9] V. Wu, R. Manmatha, and E. M. Riseman. Finding text in images, 1997. 1, 2, 4

[10] W. Wu, F. Li, J. Kong, L. Hou, and B. Zhu. A bottom-up ocr system for mathematical formulas recognition. In D.-S. Huang, K. Li, and G. W. Irwin, editors, *Intelligent Computing*, pages 274–279, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 1

[11] J. Zhang, J. Du, S. Zhang, D. Liu, Y. Hu, J.-S. Hu, S. Wei, and L.-R. Dai. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition*, 71:196–206, 2017. 1, 4