Recognition of Handwritten Mathematical Expressions Using Convolutional Neural Networks

Mrudhula Murali
mrudhula@usf.edu

Rajitha
rajithap@usf.edu

Abstract

Document recognition and retrieval technologies complement one another, providing improved access to increasingly large document collections. While recognition and retrieval of textual information is fairly mature, with wide-spread availability of Optical Character Recognition (OCR) and text-based search engines, recognition and retrieval of graphics such as images, figures, tables, diagrams, and mathematical expressions are in comparatively early stages of research. This paper surveys the state of the art in recognition and retrieval of mathematical expressions, organized around four key problems in math retrieval (query construction, normalization, indexing, and relevance feedback), and four key problems in math recognition (detecting expressions, detecting and classifying symbols, analyzing symbol layout, and constructing a representation of meaning). Of special interest is the machine learning problem of jointly optimizing the component algorithms in a math recognition system, and developing effective indexing, retrieval and relevance feedback algorithms for math retrieval. Another important open problem is developing user interfaces that seamlessly integrate recognition and retrieval. Activity in these important research areas is increasing, in part because math notation provides an excellent domain for studying problems common to many document and graphics recognition and retrieval applications, and also because mature applications will likely provide substantial benefits for education, research, and mathematical literacy.

Introduction

The math domain provides an excellent vehicle for studying pattern recognition and retrieval problems, and for studying methods of integrating pattern recognition algorithms to improve performance. The four central pattern recognition problems – segmentation, classification, parsing, and machine learning (i.e. optimizing recognition model parameters) – all come into play when recognizing mathematics.

The input to a math recognition system can take three forms: vector graphics (such as PDF), strokes (such as pen strokes on a data tablet), or a document image. The processing that is needed to extract expressions and recognize characters depends greatly on the form of input.

Mathematical typesetting systems and editors such as $\LaTeX$ are widely used for formatting mathematical expressions. They produce well-formatted and professional output. However, they are slower to use than handwriting and may have a steep learning curve for new users. A system that recognizes handwritten mathematical expressions and turns them into a machine-readable format such as $\LaTeX$ would allow a user to benefit from the best of both approaches, but building such a system is difficult. There are many different mathematical symbols used, and there often are ambiguities in symbol location and layout in handwritten expressions .

The 2013 Competition on Recognition of Online Hand- written Mathematical Expressions (CROHME) was won by Vision Objects (now named MyScript), with 60% accuracy at the expression level; second place only achieved 23% accuracy. Vision Objects is a privately held company with proprietary technology [5], so while public-domain research has the potential to reach much higher accuracies, it is currently unclear how this can be achieved. The poor results in CROHME reflect the general state of handwritten expression recognition public research. Further, this paper relies on applying Convolutional

Neural Networks (CNNs) to the task of handwritten expression recognition.

Our project investigates the problem of recognizing handwritten mathematical expressions. Our primary contribution is in image classification for offline handwritten images & to create an end-to-end system using a well- trained CNN model to go from strokes to symbols to a $\LaTeX$ expression for online handwritten images.

Level wise implementation for Handwritten mathematical expressions

Level 1: Offline recognition

In this level the data is been collected from google images & then classified the given expressions to math & non math using CNN

Level 2 : Online Recognition

1.1. Handwriting Recognition

Much work has been published to date on recognizing handwritten numbers and English words. Character and digit recognition are very well-studied problems; the MNIST dataset is often used as a dataset to try out new machine learning models because it is so widely used. In the research community, online recognition is where the writing is done in a digital medium that records pen-tip movements such as a tablet. Naturally, online recognition accuracy is higher than offline recognition accuracy, since additional information is provided for online about how the writing is done. Nevertheless, offline handwriting recognition is used in large-scale real-world systems such as interpreting handwritten postal addresses or monetary values on bank checks .

Significantly less work has been done on handwritten mathematical expression recognition. Prior to CROHME, the relatively small number of math recognition research was done without benchmark data sets, standard encodings, or evaluation tools; this made progress slow and collaboration difficult for the community. The CROHME competition, organized since 2011, seeks to make it much easier to get started working on handwritten math recognition and meaningfully compare systems .

1.2. Convolutional Neural Networks

Our approach relies heavily on CNNs, which are widely used for a variety of vision recognition problems. Many papers document ways of achieving better results when training/evaluating CNNs, including using slant correction on images, elastic distortions to increase the size of the training data set and robustness of the model , and extracting additional features from images as inputs to the CNN .We will use some of these methods in our approach to the problem.

2. Approach

Our pipeline has five distinct phases: (1) image segmentation, (2) data extraction, (3) character-level classification

Data set

Before it is possible to explain our approach, it is necessary to talk about the data we took. Our data comes from the CROHME competition, which provides train and test datasets freely available for research purposes:

There are 75 different unique mathematical symbols present in the datasets. The data is given in Ink Markup Language (InkML) format, which is a specific XML type that is specifically used to describe digital writing [1]. Each expression is represented in a separate <ink> element. Within an expression, each stroke is then represented in a separate <trace> element; the contents of the <trace> element are X-Y coordinates representing the places at which the pen-tip is over time.

```
 <trace id="0">
 1969 1746, 1969 1746, 1965 1749, 1968 1763, 1978 1760, 1989 1749, 2021 1704, 2032
 1681, 2037 1666, 2037 1658, 2031 1699, 2030 1760, 2035 1808, 2039 1848, 2025 1857,
 2013 1859, 1996 1857, 1990 1853, 2065 1835, 2083 1834, 2083 1834
</trace>
```

A sample of InkML

2.1. Image Segmentation

Though we are given trace information from the dataset, it becomes necessary to determine which traces comprise a complete symbol for eventual symbol classification. Each trace in a trace group will be part of the same symbol. So we segmented our input images on the basis of the trace group. From the trace group, we get the trace Id details for a symbol, from which we collect the stroke details. We took stroke details for each trace group and segmented into a new symbol Inkml file.

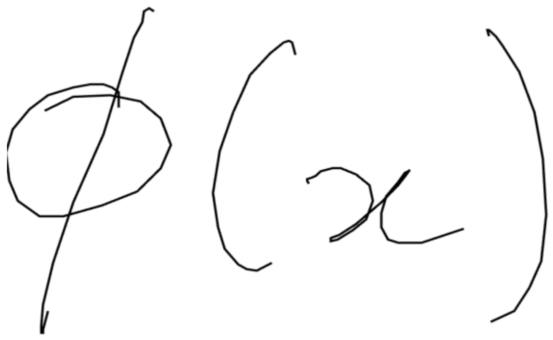


Figure 2. This expression would be perfectly segmented.

2.2. Data extraction

We transform the InkML data into a form that our models can handle. We change the InkML data comprising individual mathematical symbols into normalized images represented as pixel arrays. Then, we blur the pixels and include the number of strokes as an additional feature. The pixel arrays are normalized to size 32x32 pixels for the classifiers.

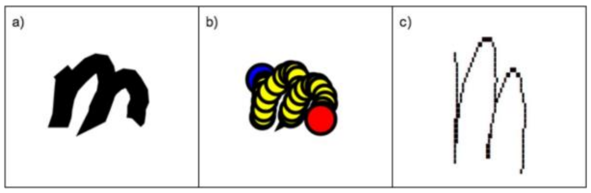


Figure 3. a) The symbol b) As InkML c) As pixel array

2.3. Character-Level classification

We train a classifier to output one of 75 symbol classes.

Finally, we spend most of our efforts in training good CNNs for the character classification task. This is done by training multiple models with different architectures and parameters in an effort to find the best trained CNN for this task.

the symbol to vary, we calculate

$$\arg\max_{s} p(A = s) * p(A = s|A \text{ is start tag}) * p(B|A = s)$$

where s denotes the symbol we are choosing. Note here that since A is chosen, C doesn't affect the probability as we are only using the probability given the preceding tag
i.e. binary potentials.

3. Experiments

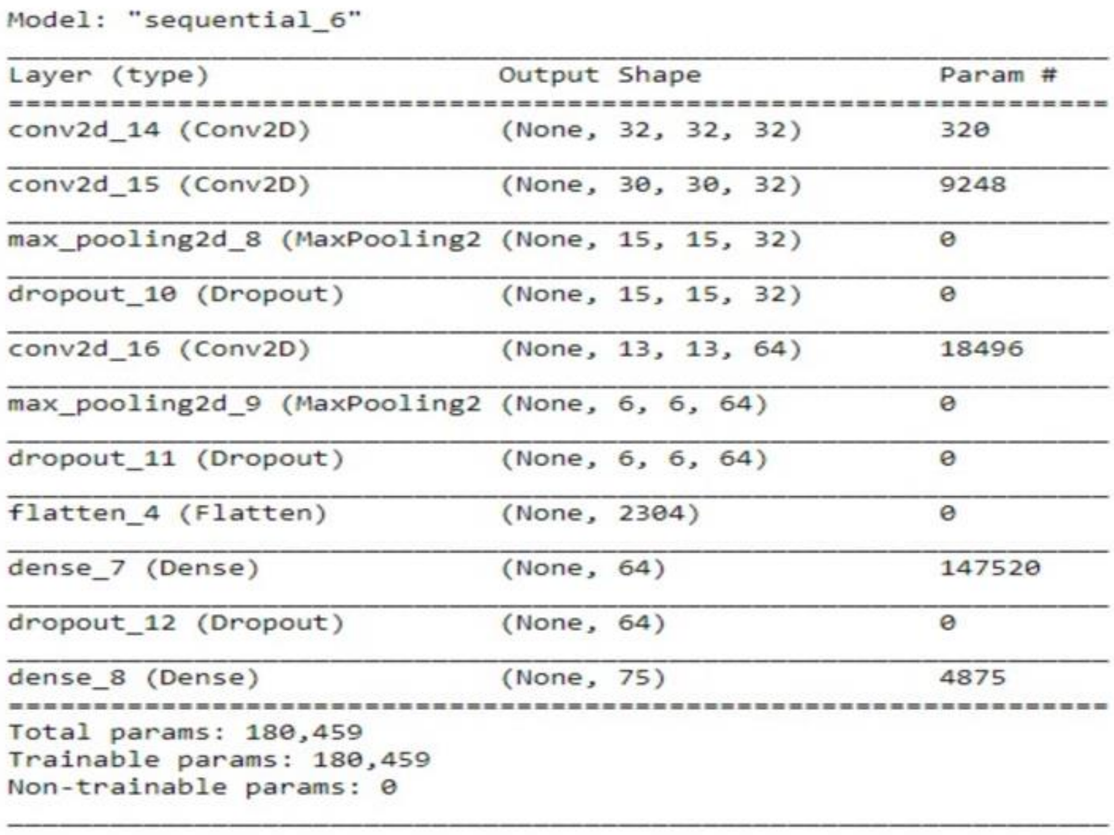
We will now report results of image segmentation and character-level classification.

3.1. Character-level classification

Our first step is to train a good classifier, which will be needed for segmentation

3.1.1 Convolutional Neural Network

Most of our time in character-level classification has been spent training and tuning CNNs. The architecture used for the CNN has 7 layers which has both dense & dropout layers. We are also using flatten to give its output to fully connected layer.various kernals & max pooling is done as shown in figure.



```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_14 (Conv2D)           (None, 32, 32, 32)        320
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 30, 30, 32)        9248
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
dropout_10 (Dropout)         (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
dropout_11 (Dropout)         (None, 6, 6, 64)          0
_________________________________________________________________
flatten_4 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_7 (Dense)              (None, 64)                147520
_________________________________________________________________
dropout_12 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_8 (Dense)              (None, 75)                4875
=================================================================
Total params: 180,459
Trainable params: 180,459
Non-trainable params: 0
_________________________________________________________________
```

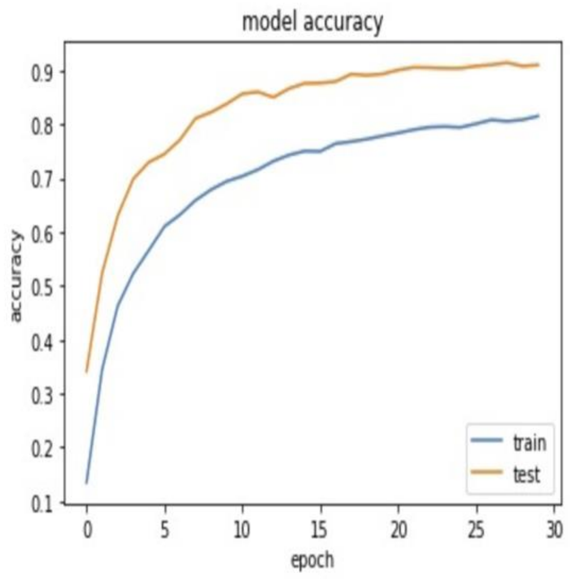
Metrics

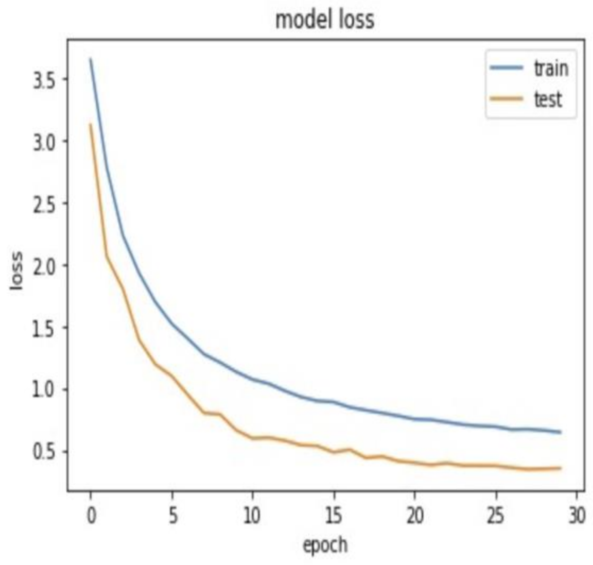
InkML Size	Symbols	UniqueSymbols	Loss(%)	Accuracy(%)
921	11224	56	33	90
1338	16970	75	34	91

Our best results for each input size & after parameter tuning are above.

3.1.2. Segmentation

Now, we perform segmentation as described previously. After performing segmentation we have applied to CNN model consisting of 7 layers with some dense,droupout layers & tweaked it to get best accuracy & less loss. Various plots are shown below to see models performance





Of course, in our segmentation approach we are making the big assumption that if strokes are overlapping, they be- long to the same symbol. Below is a simple example where our segmentation approach fails.

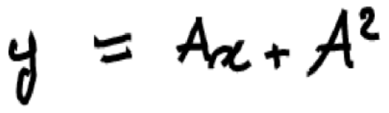


Figure 7. A case where segmentation fails. The symbols A and x intersect, so segmentation will mistakenly view them as one sym bol instead of two.

Conclusion & Future Work

From our research, we've learned that CNNs are a strong approach to solving handwritten expression recognition. With better computing resources and more time, it may be possible to achieve expression level classification & significantly increase the overall expression ac- curacy. Small percent increases in accuracy at the symbol level have drastic effects at the expression level, since the symbol accuracies are compounded at the expression-level. One potential way to increase symbol-level accuracy is to train deeper CNNs with smaller filter sizes. Another would be to do additional types of dataset distortions such as rota- tions and other affine transformations.

There are several limitations with our work. Our ap- proach to segmentation must be refined; the 88% segmen- tation accuracy cuts the overall expression accuracy in half. There are many other segmentation approaches to try; one is the approach that the second-place team, from Univ. Valen- cia, did for CROHME 2013. Their system contains a parser which builds multiple hypotheses for segmentation, sym- bol classification, and structural relation. The most likely hypothesis is then chosen. This is a natural extension to our work; we should include different segmentation hy- potheses as input into the HMM as well as part our expression level classification.

Another limitation is that we ignore spatial information. However, spatial information to recognize fractions, summations, etc. is necessary in mathematical handwriting recognition. Again, we could try the approach done by the Univ. Valencia team, which encodes spatial information while building their hypotheses .

A third limitation is that we ignore size information. This is evident by our misclassified symbols graph, which shows that the comma is the most over-represented misclassified symbol. By including a normalized size into the last affine layer of the CNN, this issue may be mitigated. All of these are promising potential areas of future work.

References

1. Ink markup language. http://www.w3.org/TR/InkML/.
2. CROHME: https://www.isical.ac.in/~crohme/
3. CROHME 2019: https://www.cs.rit.edu/~crohme2019/
4. Online Handwritten Recognition Stanford Paper: http://cs231n.stanford.edu/reports/2015/pdfs/mohan_lu_cs231n-project-final.pdf
5. Recognition of Online Handwritten Mathematical Expressions Stanford Paper: https://pdfs.semanticscholar.org/b024/eaab5a88a1a0019ad81a1507787f65cbd131.pdf
6. Github Link Image Classification: https://github.com/jungomi/math-formula-recognition
7. Chrome Python Resource: https://programtalk.com/vs2/?source=python/8033/hwrt/
8. Inkml Viewer: http://saskatoon.cs.rit.edu/inkml_viewer/
9. Github Link:https://github.com/anujshah1003/own_data_cnn_implementation_keras