Conversion of Handwritten Mathematical Expressions to LaTeX

Sharad Roy(14628), Gaurav Kumar(14240)

Indian Institute of Technology, Kanpur

Table of contents

1. Introduction
2. Dataset
3. Character Segmentation
4. Tree Generation
5. Classification
6. Results

Introduction

Motivation

- Digital documents have increased many fold in recent past.
- Difficult to render mathematical equations using plain text editor.
- Led to an increase in popularity of typesetting systems like LaTeX.
- Typing equation on LaTeX is cumbersome and time consuming.
- Important to perform recognition of mathematical equations.
- Handwritten digit recognition has been widely studied, but work on mathematical equations is still rare.

Existing Work

- Premilinary work done by LeCun et. al for handwritten digit recognition. Used for Zipcode reading in postal service
- Implemented a 7 layer Convolutional Neural Network for this.
- Awal et. et al tries to look at strokes inside text for segmentation and performs it simultaneously along with recognition and interpretation.
- Web app call detextify performs only symbol wise recognition of expressions and that too with many errors.
- ”Alvaro et. al - Offline Features for Classifying Handwritten Math Symbols with Recurrent Neural Networks” does this task to some extent, however fails in nested super/sub-script cases.

Dataset

Dataset

- Dataset is taken from kaggle and has 82 different classes. A snippet of the dataset is as follows

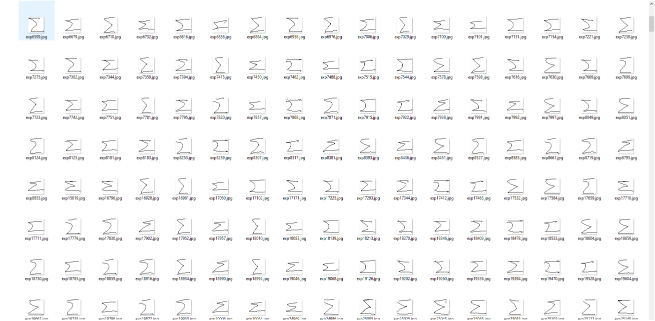


Figure 1: A snippet from Sigma class

Dataset Pre-processing

- Dataset is skewed with character '-' being the most frequent with 12328 data points and character '∃' the least with 11 data points
- Consider top 40 classes for our case to reduce class imbalance

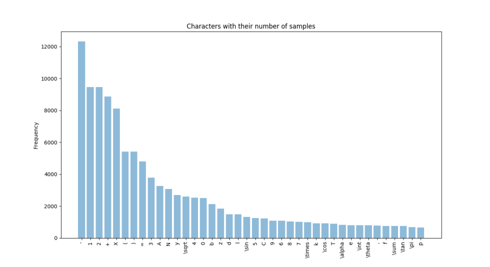
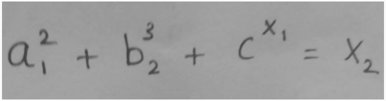


Figure 2: Histogram showing distribution of each character

Character Segmentation

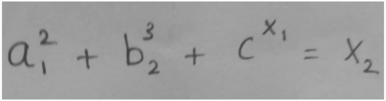
Preprocessing

- **Denoising:** Image is denoised using a median blur filter



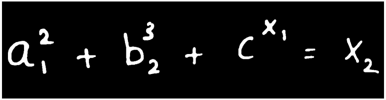
Preprocessing

- **Denoising:** Image is denoised using a median blur filter



$$a_1^2 + b_2^3 + c^{x_1} = x_2$$

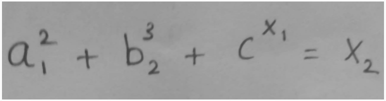
- **Binarization:** Used *minimum thresholding* to binarize the text.



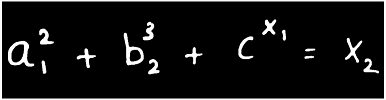
$$a_1^2 + b_2^3 + c^{x_1} = x_2$$

Preprocessing

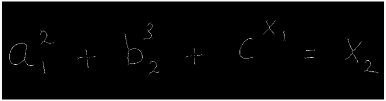
- **Denoising:** Image is denoised using a median blur filter



- **Binarization:** Used *minimum thresholding* to binarize the text.



- **Skeletonization:** The binary image is skeletonized to 1 pixel stroke thickness to match the dataset images.

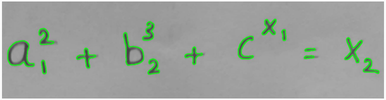


Contouring

- **Edge Detection:** Used *Canny Edge Detection* to draw the boundaries of characters.
- **Contouring:** Found the curve joining all the continuous points

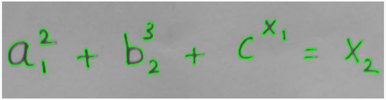
Contouring

- **Edge Detection:** Used *Canny Edge Detection* to draw the boundaries of characters.
- **Contouring:** Found the curve joining all the continuous points

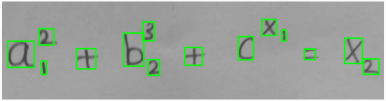


Contouring

- **Edge Detection:** Used *Canny Edge Detection* to draw the boundaries of characters.
- **Contouring:** Found the curve joining all the continuous points

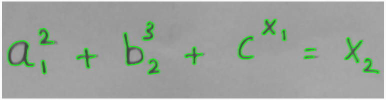


- **Segmentation:** Found the bounding box enclosing each individual outer contour.

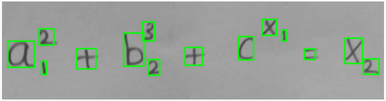


Contouring

- **Edge Detection:** Used *Canny Edge Detection* to draw the boundaries of characters.
- **Contouring:** Found the curve joining all the continuous points



- **Segmentation:** Found the bounding box enclosing each individual outer contour.



- **Deskewing:** Computed the skew of the character by using image moments and applied affine transformation.

Tree Generation

How tough are you?

In general, there are three ways a new character is related to the previous character:

- Super-script
- Sub-script
- Same level

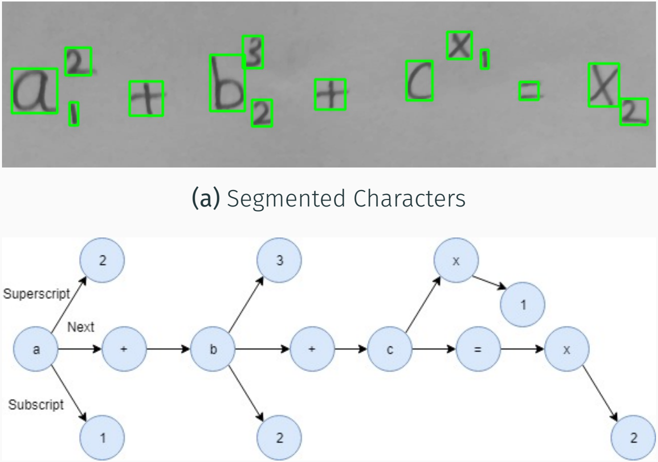
How tough are you?

In general, there are three ways a new character is related to the previous character:

- Super-script
- Sub-script
- Same level

Developed a recursion based algorithm to generate a spanning tree using the relative positions of the character boxes.

Example tree



(a) Segmented Characters

(b) Spanning Tree

Figure 3: Illustration of Super/Sub-script Tree

Classification

Feature Selection

Classification can be done using two approaches:

- Use handcrafted features like HOG, SIFT, SURF
- Use deep neural networks like CNN

After pre processing, image is binary and skeletonized to 1 pixel.

- Evident that SIFT and SURF features should not work very well for this task.
- Histogram of Oriented Gradient feature is used for each image.
- Stroke patterns of characters are important and hence HOG is a good descriptor for capturing this.

HOG Features

- Resize each image to 45×45 dimensions.
- Find hog descriptors for each image.
- Parameters for HOG Descriptors include the following :-
    - BlockSize =(10,10)
    - BlockStride= (5,5)
    - Cell size =(10,10)
    - nbins =9
- These parameters have been reached upon by tuning and careful inspection.

Results on different classifiers

- After feature extraction, different classification models are trained .
- Following results were reached upon:

Model	Accuracy on Test Data
SGD Classifier	89.5
Logistic regression	92.5
Naive Bayes Classifier	68.3
Decision Tree Classifier	64.5

So can we do better, yes we can !!!!

Time to move to CNN

- **Input Layer:** Raw pixel values of image normalized between 0 and 1
- **Convolutional layer**: 30 5×5 filters are applied followed by ReLU Activation followed by max pooling operation
- **Convolutional layer**: 15 3×3 filters are applied followed by ReLU Activation followed by max pooling operation
- Apply dropout layer next to it to make the system robust and reduce complexity.
- Flatten the layer and add two fully connected layer next to it.
- Apply softmax at the last layer to get class probabilities.

Hyperparameters Tuning

- **Loss function** used is categorical cross entropy
- **Optimizer**: Adam
- Adam optimization algorithm is an extension to stochastic gradient descent. Adam improves on the benefits of both AdaGrad and RMSProp.
- A lot of other parameters i.e, Batch Size and Number of Epochs were tuned to get the best results.
- **Batch Size**: 128
- **Epochs**: 20
- **Accuracy**: 98.59 %

Results

Sample Result 1

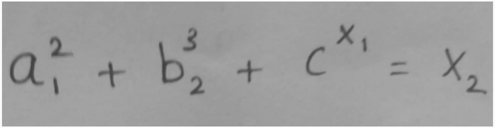


Figure 4: Sample Input 1

LaTeX Output:

$$a_1^2 + b_2^3 + c^{x_1} = x_2$$

Sample Result 2

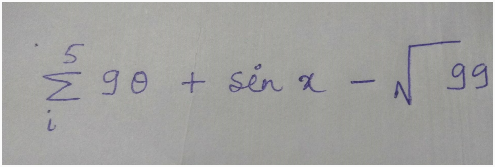


Figure 5: Sample Input 2

LaTeX Output:

$$\Sigma_b^5 9\theta + \sin x - \sqrt{9}9$$

Sample Result 3

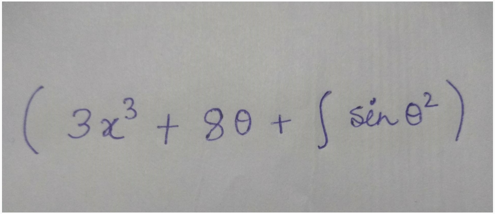
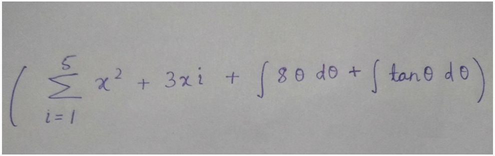


Figure 6: Sample Input 3

LaTeX Output:

$$(3x^3 + 8\theta + \int \sin\theta^2)$$

Sample Result 4



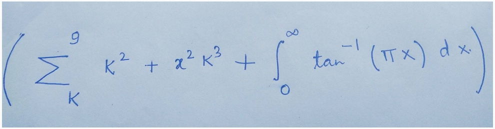
$$\left(\sum_{i=1}^{5} x^2 + 3xi + \int 8\theta \, d\theta + \int \tan\theta \, d\theta \right)$$

Figure 7: Sample Input 4

LaTeX Output:

$$({}_1\Sigma_1^5 x^2 + 3x1 + \int 8\theta d\theta + \int \tan\theta d\theta)$$

Sample Result 5



$$\left(\sum_{K}^{9} K^{2} + x^{2} K^{3} + \int_{0}^{\infty} tan^{-1} (\pi x) \, d x \right)$$

Figure 8: Sample Input 5

LaTeX Output:

$$(\Sigma_k^9 k^2 + 2^2 k^3 + \int_0^\infty \tan^{-1}(\pi x) dx)$$

Whats New?

- Contour based segmentation is faster than the popular sliding window segmentation method.

Whats New?

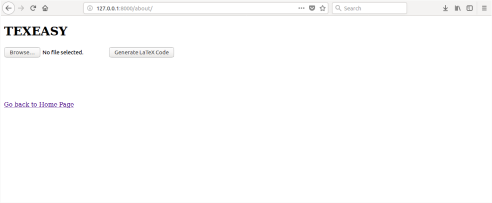
- Contour based segmentation is faster than the popular sliding window segmentation method.
- Recursion based tree algorithm takes care of virtually infinite nested super/sub-scripts.

Whats New?

- Contour based segmentation is faster than the popular sliding window segmentation method.
- Recursion based tree algorithm takes care of virtually infinite nested super/sub-scripts.
- Complete end-to-end conversion web application.

Further Work

- Addition of *frac* function for multi-line fractions.
- We are working on a web based application for this task.
- It takes as input an image and returns the latex code.



127.0.0.1:8000/about/

TEXEASY

Browse... No file selected.

Generate LaTeX Code

Go back to Home Page

Figure 9: Screenshot of Web based application

References

- ”Handwritten math symbols dataset” by Xai Nano on kaggle.
- Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner. ”Gradient-based learning applied to document recognition” Proceedings of the IEEE, Volume 86, Issue 11 (1998).
- Ahmad-Montaser Awal, Harold Mouchere, Christian Viard-Gaudin. ”Towards handwritten mathematical expression recognition.” 10th International Conference on Document Analysis and Recognition(ICDAR), 2009
- ”Detexify” by Kire Labs Link