Project Report
Recognition and Evaluation of Handwritten Mathematical Expressions

1. Introduction

1.1 Problem Statement

In this project, we will recognize the mathematical expressions, extract the digits, alphabets and mathematical operators that make up the expression as images for evaluation. To evaluate the handwritten Mathematical Expressions, we would be implementing Convolutional Neural Networks(CNN). The webcam captures handwritten expression as frames and pre-processing of the images is done through OpenCV. The Handwritten math symbols dataset from Kaggle which has over 100000 image samples is used for training purpose whereas testing is done on the handwritten expressions taken from webcam. The dataset had missing symbols like times (*) which we created artificially by taking 'X', rotate it to 45 degrees and bitwise & the original with the result. We then interpret algebraic equations from alphanumeric expressions and provide rules to solve systems of equations.

1.2 Motivation

There have been a lot of papers written on recognition of mathematical symbols, digits and characters, but very few research has been done to our best knowledge on evaluation of the handwritten mathematical expressions. Most of the tools or apps we have can evaluate the typed mathematical expressions but doesn't support handwritten expression evaluation. As more computer devices with better displays and input devices become available, demand will increase substantially for software systems to work with handwritten data. In our project, we have attempted to recognize and evaluate a large variety of handwritten mathematical expressions/operators via Convolution Neural networks and Sympy.

1.3 Review of other Researchers

We have referred the paper "Using Artificial Intelligence to evaluate Handwritten Mathematical Expressions" by Will Forfang where the Artificial Neural Network was used to read his handwriting from example images. The algorithm was fed with ten images containing handwritten digits (0-9), letters (x, y, m) and mathematical operators (+, -, /, %, * and =). The three-layer artificial neural network has the following topology:
Input Layer: 5625 neurons. The 75*75 pixels are flattened and sent to distinct 5625 neurons.
Hidden Layer: 50 neurons
Output Layer: 18 neurons (Total number of characters the algorithm has learned.).
As the training set is relatively small, a simple topology was chosen. However, to improve the performance experiments need to be performed on the topology.

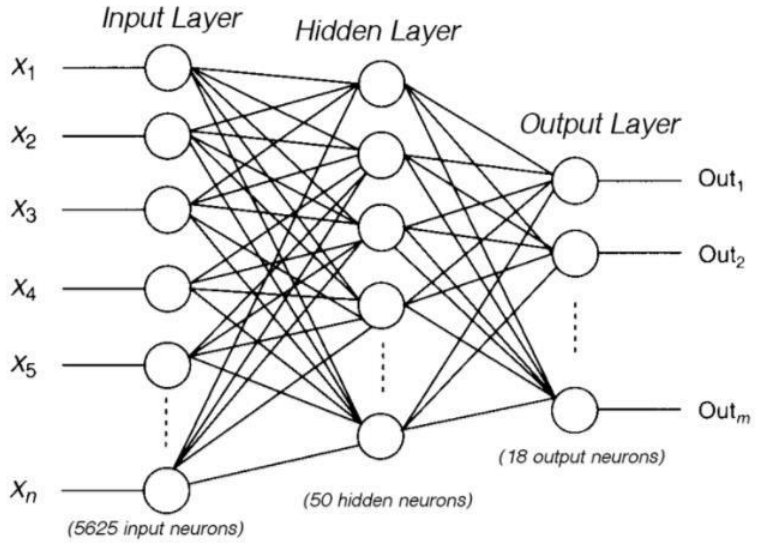


Fig 1: Artificial Neural Network

The characters where isolated using segmentation rules and each character was mapped to it's correct value using an "answer sheet". Ramer Douglas Peucker algorithm was used during segmentation. It generated representations of pen stroke from collection of points, thus converting a huge list of pixel coordinates into a small list of ploy lines. The ANN algorithm and Sympy was able to recognize and evaluate basic arithmetic operations, compound fractions and algebraic expressions.

The future scope includes:

- Improving topology to optimize the performance
- Expand the training dataset
- To recognize and evaluate subscripts and superscripts
- To solve differential equations

1.4 Open questions in the domain

Recognition and evaluation is one of the most challenging areas in the field of image processing, pattern recognition and mathematical computations. The foremost problem we face while dealing with mathematical expressions is data collection. Even though we have MNIST data set for numbers, the other handwritten mathematical symbols are difficult to gather.

Evaluation can be interpreted in different ways. For example: f(x+1) can be considered as f multiplied by (x+1) or function of (x+1).

1.5 Proposed Approach

The proposed approach involves two stages: recognition of mathematical symbols and evaluation of mathematical expressions. The dataset for training purpose was taken from Kaggle which consisted of handwritten images (100000). Since, asterisk symbol was missing we artificially created it by taking the image 'X' which was already present in the dataset, rotating the image 'X' to 45 degrees and then apply bitwise & operation to original image with the result. A connection is made with the webcam from which frames where collected and sent to the pre-processing function. For recognition of the mathematical symbols we are using Keras deep learning library to build Convolutional Neural Networks. The CNN allows us to attain better results as the image processing is computationally manageable through filtering the connections by proximity. Since the dataset is large the CNN model takes time to train. To overcome this, we are saving the model architecture in JSON format and model weights in HDF5 format. The

mathematical expressions are evaluated using Sympy. Sympy is a python library for symbolic mathematics. Although there are many computer algebra systems we have chosen Sympy for evaluation purpose as it uses Python and can be used as a library.

1.6 Reflection of Feedback

There have been many research papers on recognition of mathematical expressions/symbols using various algorithms such as Random Forest, Support Vector Machine, Multi-Layer Perceptron and Convolutional Neural Networks. When it comes to evaluation of mathematical expressions, the expressions which are already present are fed as images into the model. Some of the papers are:
https://github.com/JoseJoy249/Handwritten-Expression-Evaluation
http://par.cse.nsysu.edu.tw/~algo/paper/paper16/B2_1.pdf

We are trying to directly evaluate the mathematical expressions in real time by capturing it as frames through webcam. This approach is useful in real world computation. Also, for evaluation purpose usage of Sympy is another interesting aspect.

2. Background & Related Work

A lot of research papers have been published on handwritten recognition of digits with MNIST being the most popular dataset. Works have also been done on other datasets such as CROHME and USPS dataset. When it comes to symbol recognition most of the people have either used dataset form Kaggle which has arithmetic operators, set-symbols, comparison symbols, delimiters etc. The recognition of handwritten mathematical expression can be classified into on-line/off-line. In offline recognition the handwritten expressions are given in the form of images where as in the online recognition the handwritten expressions are given in a digital medium where recording is done with pen devices and the digital ink stores the data. In real-world systems offline handwritten recognition is used in applications such as credit card imprint and postal code, monetary values on bank cheques. However, less work has been done on handwritten mathematical expression recognition due to lack of a proper dataset.

In 1999, Nicholas E. Matsakis described an online recognition of handwritten MEs for converting a handwritten mathematical expression into an expression in a typesetting command language such as MathML and a feedback-oriented user interface which helps to make the errors more tolerable as they can be rectified quickly. The three primary components of the system are a method of classifying isolated handwritten symbols, an algorithm for partitioning an expression into symbols and an algorithm for converting a 2D arrangements of symbols into a typeset expression.

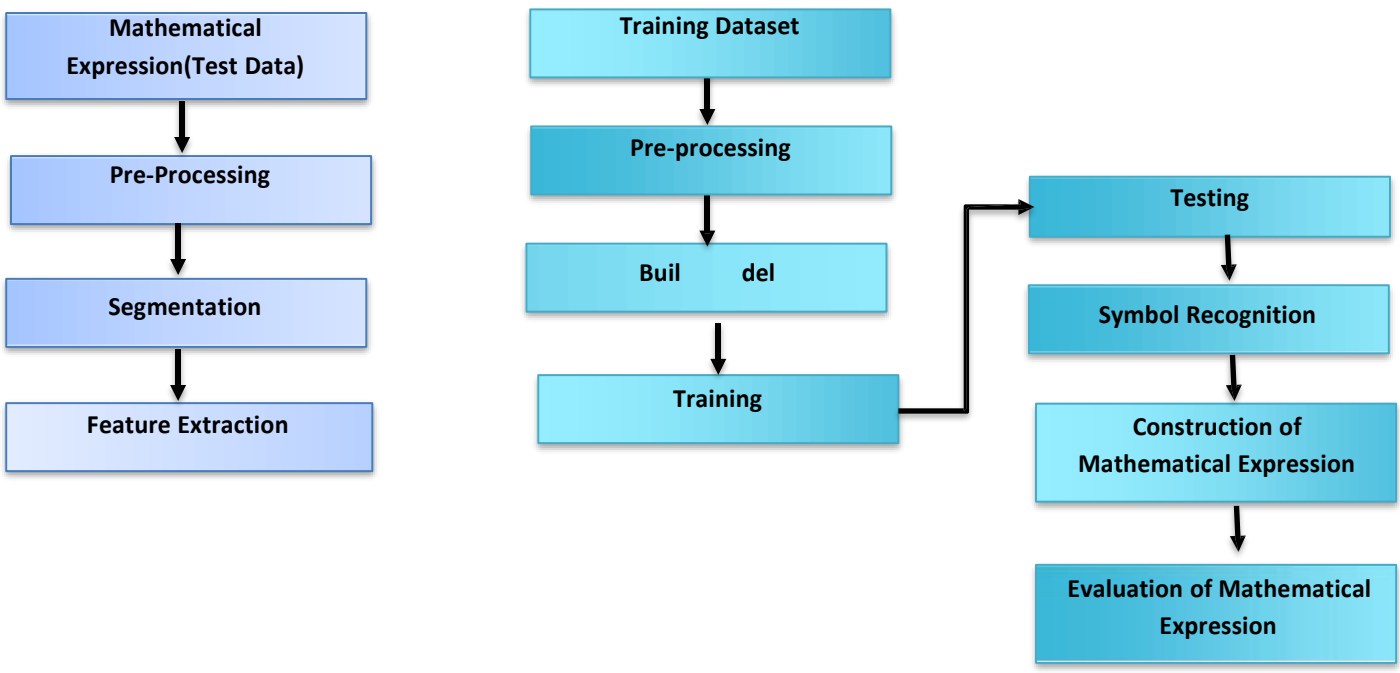
In 2010, Ahmad-Montaser Awal discussed about the problems of handwritten Mathematical Expression Recognition Evaluation. The first issue is to define how to ground truth a dataset of handwritten mathematical expressions and next is the issue of benchmarking systems. According to this paper the mathematical expressions should only be grounded by their content but foremost with their layout and displayed symbols.

In 2016, Will Forfang published the idea of one of his project “Using Artificial Intelligence to Evaluate Handwritten Mathematical Expressions” where he built an algorithm which contains Artificial neural network (ANN) that has learned to read his handwriting from example images. This algorithm was provided with some segmentation rules i.e. how to isolate individual characters for analysis and an “answer sheet” which mapped each written character to its correct value.

Paper	Online	Offline	Model	Dataset Size	Accuracy Claimed
[1]			Multilayer Perceptron Neural network	839 symbols: including digits, Roman letters, Greek Letters	87.5%
[2]			Artificial Neural Network	18 symbols: Including digits, letters, variety of operators	Not specified
[3]			Symbol Classification: Gaussian Classifier Parsing: Simple geometric grammar	Over 500 symbols were collected using a Wacom digitizing tablet and a custom collection program written in java	99%

3. Methods

Recognition and evaluation of mathematical expressions typically consists of 6 major stages: pre-processing, segmentation, feature extraction, symbol classification, construction of mathematical expression and evaluation of mathematical expression. The architecture of recognition and evaluation of mathematical expression for training and testing dataset in shown in figure 2.



3.1 Mathematical Expression

The mathematical expression is captured via webcam and OpenCV. The frames which are collected from webcam are in BGR format which is converted into gray scale image and sent for image pre-processing.

3.2 Pre-processing

The quality of the image can be improved through pre-processing**.** Gray scale image is converted into binary image, by selecting a binary image Otsu binarization is applied. Morphological transformations are applied to the images as it removes noise. We have applied both the morphological operators Erosion followed by Dilation as erosions removes the small white noises, detach connected objects. Since erosion shrinks are object we dilate it to increase the area of the object and join the broken parts of an object.

3.3 Segmentation

In segmentation, the input is segmented into individual symbols. The segmentation is now performed by threshold function. We chose adaptive thresholding as the image would have different lighting conditions in different areas. Adaptive thresholding algorithm calculates the threshold for small regions of the image. By doing so, we get different thresholds for different regions of the same image and thus yield better results for images having different lightning conditions.

3.4 Contour Extraction

Contours are used for object detection and recognition. In accordance with OpenCV, the object to be recognized is in white while the background is in black. The parameters given to contours are source image (In our case the segmented or threshold image), the hierarchy (Chose RETR_TREE as it retrieves the entire hierarchy of the image) and the approximation method of contour(CHAIN_APPROX_SIMPLE). We take into consideration the contour area greater than 50 pixel and a straight bounding rectangle where the height(length) is greater than 23 pixel. We wrote rules to fix the issue for symbols that contain two shapes like "=".

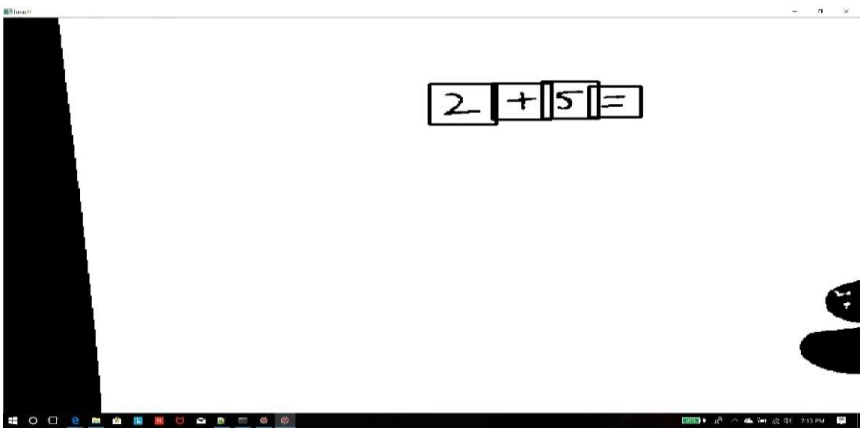


Fig 3: Result of Contour Extraction

3.5 Construction of Mathematical Expressions

To construct mathematical expressions, we have taken x and y coordinates of the expression and compared the single expression using x coordinates and multiple expressions using y coordinates.

3.6 Classification (Convolutional Neural Network)

The CNNs are used to recognize objects and performs object detection, segmentation. One of the main advantage is, it eliminates the need for manual feature extraction as it is directly handled by CNN. It arranges the neurons in three dimensions (width, height, depth). Every layer of CNN transforms the 3D input volume to 3D output volume of neuron activations. In our project, the project flattens the cropped image into vectors of length 2025 (45*45). We have used the Sequential model to initialize as it allows us to easily stack sequential layers of the network in order from input to output.

The three main layers which are used to build CNN are: Convolutional Layer, Pooling Layer and Fully-Connected Layer.

Convolution Layer:

This is the main operation in CNN and we have 8 convolution layers. It will compute the output of neurons that are connected to local regions in the input.

```
model.add(Conv2D(30, 5, 5, border_mode="same", input_shape=(height, width, depth)))
```

The first three parameters of the layer correspond to the following:

The number of convolution filters
The number of rows in each convolutional kernel
The number of columns in each convolutional kernel

We decreased the filter size as we moved to other layers (140,120,100,100,90,90,80) with kernel size of 5*5. This structure was chosen as it gave us the highest accuracy. We assigned the value "same" for border_mode as it does some padding around the input image, thus making the output image size same as that of input. The activation function we used for all the 8 layers is relu.

Max Pooling:

We used pooling layer for the first 5 convolutional layers. It basically reduces the spatial size i.e. height and width (not depth). The size of the filter maps is reduced as we apply max filter to the non-overlapping sub regions. The total number of filter maps is reduced by a factor of 4 by using 2*2 max filters (pooling size=2). We have given a stride of (2,2) which specifies how much we need to shift the kernel in each step to compute the next pixel in the result (stride size same as pool size by default). For the dim ordering parameter, we used tf since the depth is at index 3.

Fully Connected Layer:

To complete the model architecture, we flatten the output from the previous layers and enter a fully connected layer. The fully connected layer is declared using Dense() layer. We have taken 500 nodes each activated by relu function. Next, used SoftMax classification in the output layer which is the size of the total number of classes so that we could get one node per class label.

3.7 Evaluation of Mathematical Expressions

We evaluated handwritten mathematical expressions (arithmetic operators +, -, *) and univariant linear equations

As SymPy doesn't validate the mathematical expression but evaluates, some functions to check the validity of the expressions are developed. These functions help in removing the noisy contours detected by OpenCV and pass a valid expression to SymPy.

eolCheck():

This function checks for the balanced parentheses and if the expression is imbalanced in parentheses then eolCheck() returns false, otherwise returns true.

invalidTokenCheck():

SymPy cannot convert integers like “02” to “2” and even python raises an exception in doing so, which should not be the actual case i.e. when the expression is “02 + 03” it should be computed as 5. This function helps in converting such strings. So, the expression “02 + 03” after invalidTokenCheck() call becomes “2+3” which is a valid expression for SymPy.

checkSpecialChars():

Expressions like “2**3” should be treated as “2^3” instead of just discarding them, whereas “2***3” is not a valid expression. This function handles expressions like these and returns true if it is a valid expression. It also checks whether the expression ends with a valid character or not i.e. any valid arithmetic mathematical expression must end with a digit or a closed parenthesis. This function looks for such type of validity and returns true it is valid, otherwise returns false.

4. Experiments

Initially, when we built our model, it resulted in accuracy of 39%. To increase the accuracy, we increased the convolutional layers from 3 to 8 and used max pooling for the first 5 convolutional layers. By doing so, the accuracy increased to 80%. To further improve the accuracy of the model, we experimented with different structures. During the tensor flow assignment, we had an idea that as the number of neurons decreased in each layer it resulted in better performance. Based on this, we chose a structure where we decreased the number of filters in each layer (140,120,100,100,90,90,80) and this gave us the best accuracy 93%.

The CNN algorithm was wrongly recognizing the test data. The test data collected from webcam was pre-processed, but the training dataset was not fitting the test data. So, we pre-processed the training dataset. We applied thresholding and gaussian blur for pre-processing the training dataset.

Even though we used the training dataset from Kaggle, we added our handwritten images to the dataset for better results.

Model	CNN
Convolutional layer	3
Max Pooling layer	3
Model Structure	(20,50,100)
Worst Accuracy	35%

Model	CNN
Convolutional layer	8
Max Pooling layer	5
Model Structure	(140,120,100,100,90,90,80)
Average Accuracy	93%

5. Conclusion

We have performed the recognition and evaluation of handwritten mathematical expressions (arithmetic operators +, -, *) and univariant linear equations successfully. The most challenging part was pre-processing and evaluation of the expression. We worked a lot on pre-processing the training dataset as well as test dataset. There is still a lot more to be done for pre-processing of the image. For instance, the webcam takes other background images along with the equation which should be eliminated. Even though we had used Sympy for evaluation of mathematical expressions, we had to write a lot of rules like reading of the expression from left to right, checking if we have balanced parenthesis, considering integer like '02' as '2', checking for invalid tokens (For instance, multiple asterisk symbol between two digits). We had an idea regarding how to handle multiple mathematical expressions so that the system doesn't get confused but were not able to implement due to time constraint. To conclude, we gained a lot of knowledge during the entire project experience.

6. References:

[1] http://doi.ieeecomputersociety.org/10.1109/ICFHR.2010.106

[2] http://www.willforfang.com/computer-vision/2016/4/9/artificial-intelligence-for-handwritten-mathematical-expression-evaluation

[3] http://www.ai.mit.edu/projects/natural-log/papers/matsakis-MEng-99.pdf

[4] https://pdfs.semanticscholar.org/271c/41b2786f7e128afa0afdda4b32b88f08cb7f.pdf