Neural Networks for Recognizing Handwritten Mathematical Symbols

MLDN capstone project

Edward Zhou May 20th, 2017

I. Definition

Project Overview

Given the importance of mathematics in all branches of science (physics, engineering, medicine, economics, etc.), the recognition of handwritten mathematical expressions has become a very important area of scientific research. As a student from mathematics department, I understand how urge the problem is since many people preferred using pen instead of computer to present.

In this project, I will train my model with some most common mathematical handwritten symbols and then show it some mathematical symbols for it to predict.

Problem Statement

In this project, the mission for me is to let my trained model to recognize the single mathematical handwritten symbol. The possible solution to the problem could be SVM and CNN. This concerns a multi-class classification problem which is similar to the deep learning project.

Metrics

It is true that the dataset has an imbalanced nature, however, through some basic techniques, the dataset would be approximately uniform. Besides concerning the simplicity and straightforwardness,

Accuracy would be the main metrics for evaluation

$$Accuracy = \frac{correctly\ classified\ symbols}{total\ test\ size}$$

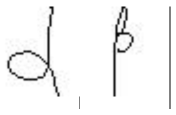
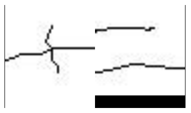
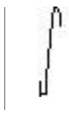
F1-score can also be a practical metrics

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

II. Analysis

Data Exploration

- The dataset can be easily obtained from kaggle(here).
- Dataset consists of (45x45) .jpg images.

- It includes basic Greek alphabet symbols like: α, β, γ, μ, σ, ϕ and θ ….
    - Sample: 
- All math operators ± = × ÷ …, set operators ∪ ∩ ∅ ….
    - Sample: 
- Basic pre-defined math functions like: log, lim, cos, sin, tan.
    - Sample:
- Math symbols like: $\sqrt{a}$ $\sum a$ $\int a$ and more.
    - Sample: 

- This dataset is the whole dataset I need for training and testing. The main task here is to classify the mathematical symbols. The frequency table below of a sample of the data shows the distribution of those symbols

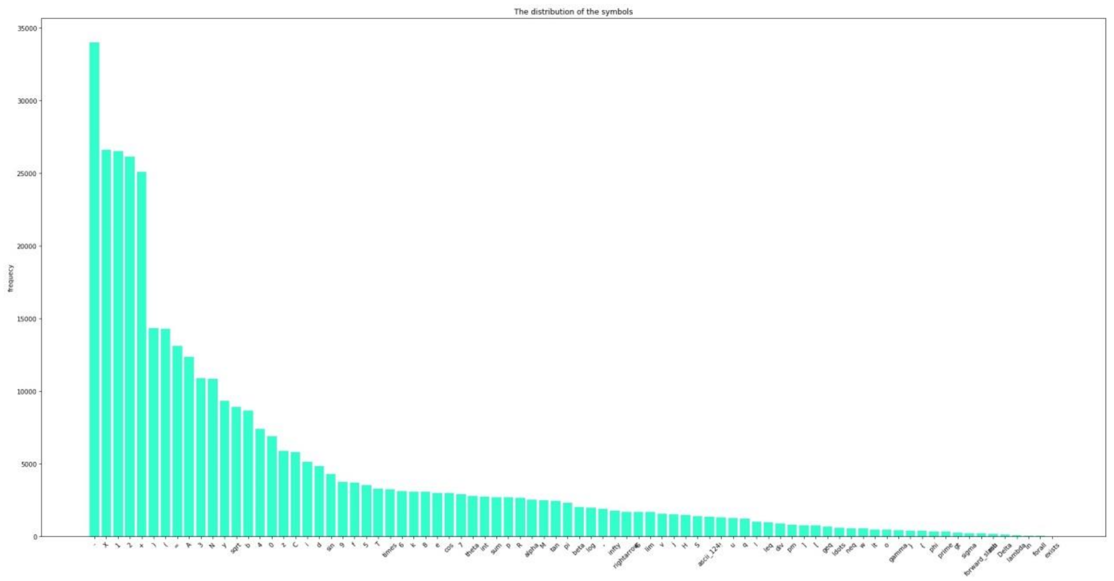
−	33997	!	1300	(	14294	)	14355	,	1906
[	778	]	780	{	376	}	377	+	25112
=	13104	0	6914	1	26520	2	26141	3	10909
4	7396	5	3545	6	3118	7	3765	8	3888
9	3737	A	12376	α	2546	b	8561	β	2025
cos	2986	lim	1675	√	8908	∀	45	≠	558

- The table above shows that the number of pictures varies much and some of symbols do not have enough pictures for the algorithm to capture its characteristic, which is a problem that need to be concerned.

- Generally speaking, the data set is cleaning and large enough for the model to train.

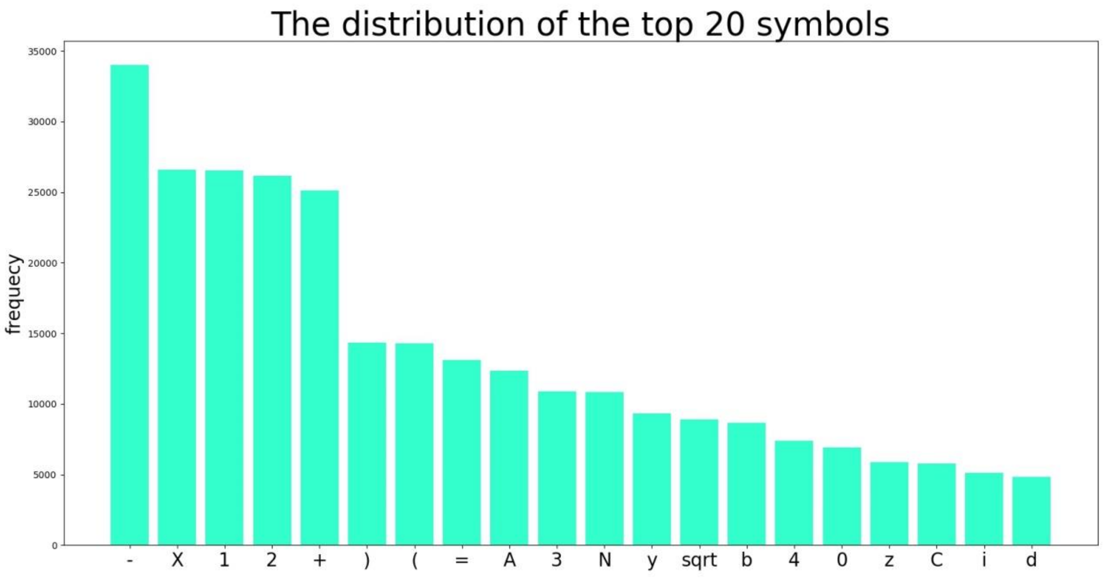
Exploratory Visualization

The plot below basically shows the distribution of the symbols that I collect:

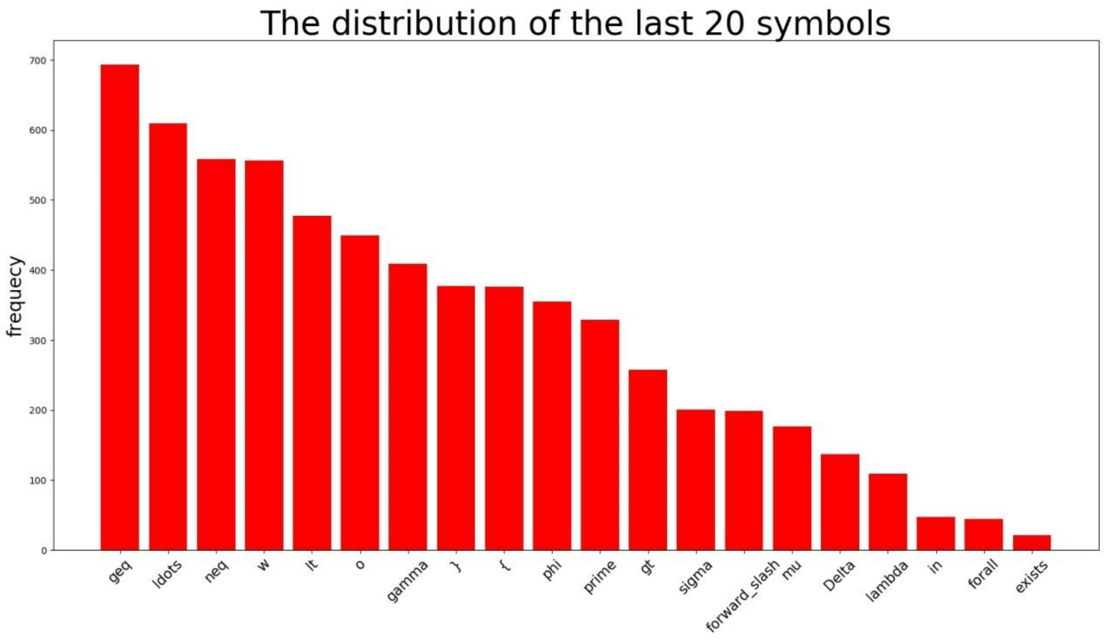


- With so many symbols to recognize, the challenge of the problem increase dramatically.
- The imbalanced nature of the dataset is obvious and it will cause problem for my model

The next two plots zoom in specific area to take a close look at the distribution.



- This is the plots of top 20 frequency symbols
- All of the frequencies are above 5000, which is enough for the model to train.



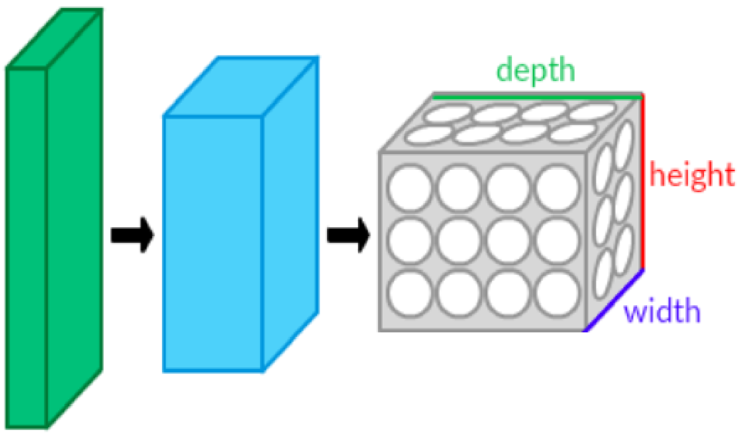
- These symbols with such a low frequency is not appropriate for training the model.

Generally speaking, the biased distribution of the dataset is a main problem for my training. Oversampling and undersampling could be ways out, which is discussed in the later section.

Algorithms and Techniques

The classifier is a Convolutional Neural Network, which is the state-of-art algorithm for most image processing tasks, including classification.

- Convolution and max pooling layer
    - Convolution layers have a lot of success with images. It uses filters to capture the features of the picture automatically.



    - Max pooling, which is a form of non-linear down-sampling. Max-pooling partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value.

- Flatten layer& Fully-connected layer
    - Flatten layer is a simple and smart way to reduce the dimensionality of the upper layer.
    - Full-connected layer with a activation function(ReLU here) is important for the whole neural network to work.

The architecture of the CNN plays a important role in the success of the whole project.

Benchmark

After CHROME(the Competition on Recognition of Online Handwritten Mathematical Expressions), papers and research flourish. However, Kaggle, which is the place that I fetch my data do not present a public kernel to the problem, so basically, I create a initial benchmark for the classifier. I use simple neuron network(a fully connected one) that is similar to the code in Tensorflow for Mnist to classify the data I got, which get a accuracy 50% .

III. Methodology

Data Preprocessing

The neural network needs to read the image data, one-hot encoded labels, and dropout keep probability. So basically, I have three steps to do here.(The preprocessing step is documented in "PrepareData" notebook)

1. Undersample the images data to make the images uniform (Approximately).I use a subset of the symbols and make all the symbols have 3000 to 5000 images.
2. The list of images is randomized.
3. The images are divided into a training set and validation set.

There are also some preprocessing steps which are done as the images get loaded into memory before training:

1. Flatten
   a) Each image is 45 pixels by 45 pixels. We can interpret this as a big array of numbers:

We can flatten this array into a vector of 45x45= 2025 numbers.

b) It doesn't matter how we flatten the array, as long as we're consistent between images.

2. Rescale

a) After I flatten the picture into array, I found the numbers in the array are integers from 0 to 255.

b) For consistency and efficiency, they are divided by 255.

2. one-hot encode the images

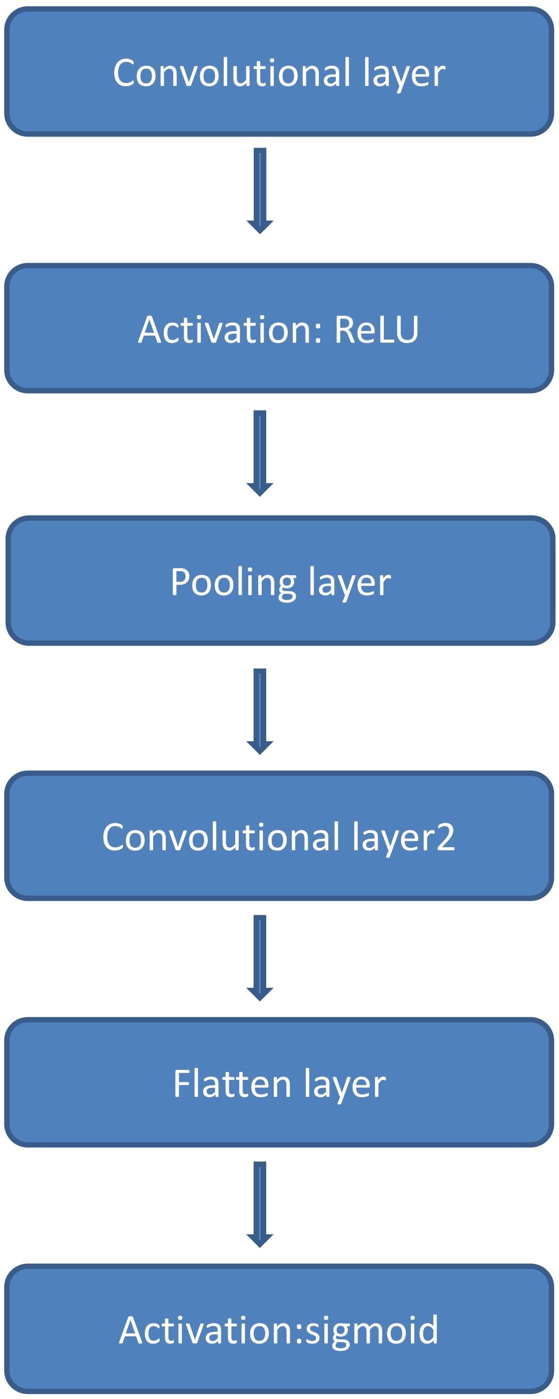
a) The labels are categorical.

b) Such representation can not be used directly with computing ,as many tools expect continuous input, and would interpret the categories as being ordered, which is often not desired

Implementation

The Implementation step is mainly concerned with the construction of the model which is documented in notebook ‘model’, this is the first architecture I tried and I think it works fine and then keep it.

The architecture of the my model is shown below:



Note:

1. The convolutional layer is defined by function conv2d which has a strides[1,1,1,1]
2. The pooling layer use max pooling method which has a ksize and strides both [1,5,5,1]
3. ReLU function : Two additional major benefits of ReLUs are sparsity and a reduced likelihood of vanishing gradient.
4. Sigmoid function is defined by :$1/(1+e^{-x})$

Refinement

As mentioned in the Benchmark section, the simple neuron network achieve a 50% accuracy.(This is not bad because the variety of those symbols) To get the initial result, the architecture above was ported to TensorFlow; the result was an accuracy around 72.5%.(the mean value of 50 mini batch).

This was improved upon by changing the following parameters:

- Change the strides of max_pool function to [1,2,2,1], in order to get a more detailed information of the images.
- Add drop out layer
- Add fully connected layer at the end of the network
- The final version of the classifier reached 80% in validation dataset and training accuracy are shown below:

```
step 0, training accuracy 0.03
step 50, training accuracy 0.21
step 100, training accuracy 0.41
step 150, training accuracy 0.48
step 200, training accuracy 0.67
step 250, training accuracy 0.66
step 300, training accuracy 0.63
step 350, training accuracy 0.74
step 400, training accuracy 0.75
step 450, training accuracy 0.75
step 500, training accuracy 0.81
step 550, training accuracy 0.81
step 600, training accuracy 0.73
step 650, training accuracy 0.74
step 700, training accuracy 0.8
step 750, training accuracy 0.84
```

IV. Results

Model Evaluation and Validation

During development, a validation set was used to evaluate the model.The final architecture and hyper parameters were chosen because they performed the best among the tried combinations.

For a complete description of the final model and the training process:

❖ The first convolutional layer learns 32 filters, the second learns 64 filters.

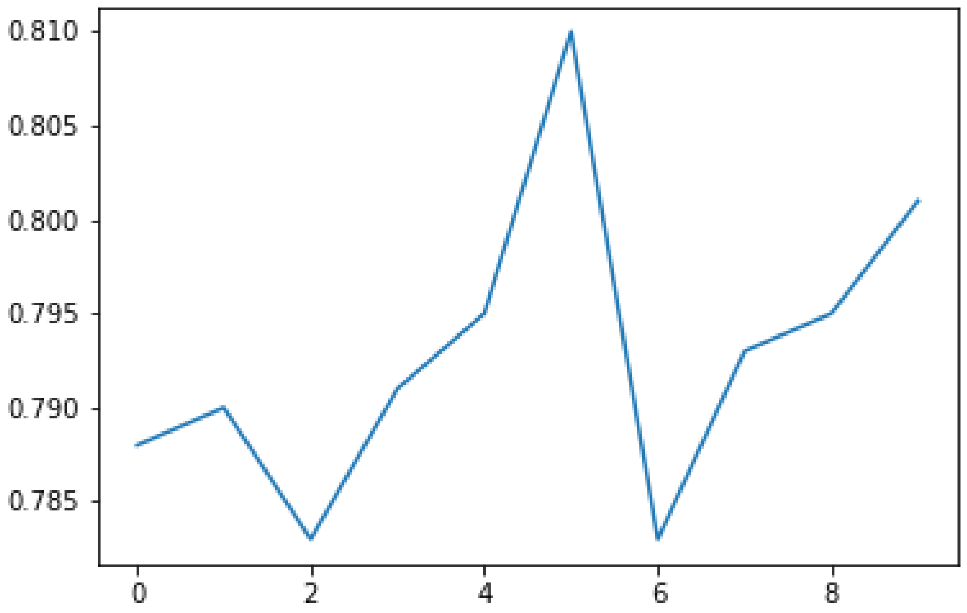
❖ The pooling layers have a stride of 2, so the resolution of the output matrices is half the resolution of the input matrices.

❖ The weights of the convolutional layers are initialized by sampling a normal distribution with a standard deviation of 0.1

❖ The training runs for 800 iterations.

To prove the robustness of the model, a shuffled validation dataset was used to test the model, there are 10 batches for the test, with 1000 images each batch:

Below is the plot of the f1_score from 10 batches:



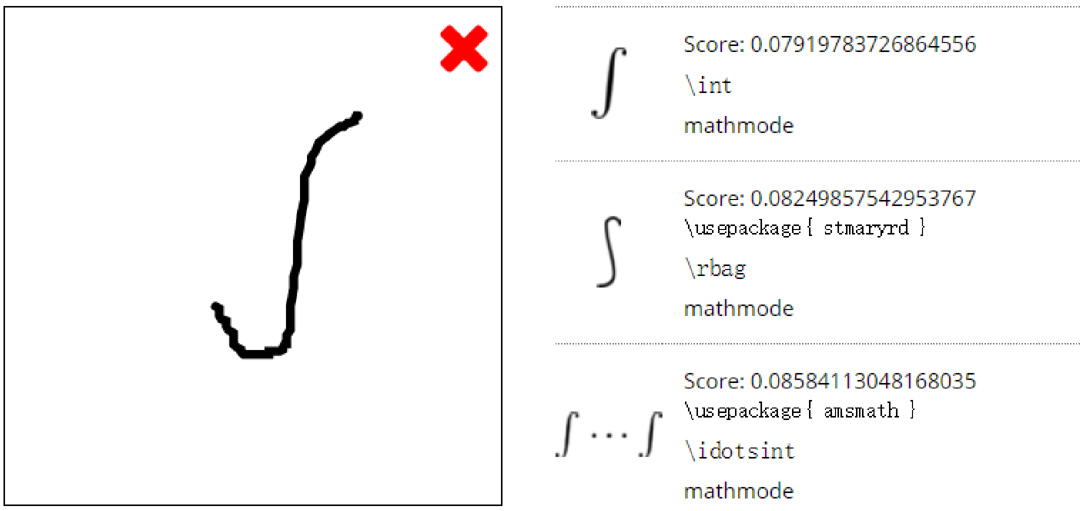
Justification

- *The final result has an accuracy 80% and f1_score 0.78,which is better than the benchmark result.*
- *I have to admit that solve the problem still needs long way to go, since I only use 30 symbols this time which is pretty small compare to all of the mathematical symbols, and 80% is not enough for practical usage.*

- *However, I understand the Neuron network and CNN deeper than before, which the most important part for the project.*

V. Conclusion

Free-Form Visualization

- This is some picture of mathematical symbols recognitions ,when successfully recognize those images, it can be easily turn to latex form:
- 

```
Score: 0.07919783726864556
\int
mathmode

Score: 0.08249857542953767
\usepackage{ stmaryrd }
\rbag
mathmode

Score: 0.08584113048168035
\usepackage{ amsmath }
\idotsint
mathmode
```

- Which is to say, we can digitalize those handwritten symbols to more robust form in many devices. It is very important for the communication among people who wants to express things with math.

Reflection

The process used for this project can be summarized using the following steps:

1. An initial problem and relevant, public datasets were found

2. The data was downloaded and preprocessed (segmented)

3. A benchmark was created for the classifier

4. The classifier was trained using the data (multiple times, until a good set of parameters were

found)

5. Evaluate the classifier with the dataset.

I found step 2 and 4 is especially difficult for me since I have never deal with raw data problem and all the tutorials on the websites use clean, uniform and well-prepared data. I have to learn the I/O methods in python first. Besides, I have to build the architecture step by step which is pretty difficult to me since I am new to Tensorflow.

Improvement

- To increase the accuracy, basically we need more data. Besides, a good computer is also needed since my machine broke down several times during training.
- We could also try to use those pre-trained model, like VGG-16(In the paper, the VGG-16 model is denoted as configuration D. It achieves 7.5% top-5 error on ILSVRC-2012-val, 7.4% top-5 error on ILSVRC-2012-test.)