ExpViz: An Automated System for Mathematical Expression Visualization

Estey, William
wpe262@rit.edu

Hurlbutt, Michael
mxh5594@rit.edu

Porillo, Nicholas
nsp6459@rit.edu

Searns, Andrew
abs2157@rit.edu

April 27, 2019

Abstract

Processing and visualizing handwritten mathematical expressions is desirable for a potential augmented reality system integrated into modern classrooms. This paper describes the development and implementation of a image processing pipeline that processes handwritten equations and visualizes them in 2D and 3D. A convolutional neural network architecture which uses of state of the art regularization techniques is presented. Our goal was to introduce a novel application of the most recent research in the areas of natural language processing, computer vision, and machine learning.

Contents

1 Introduction

Our goal was to use image processing and classification techniques to extract and parse mathematical formulas, equations, and other expressions from input image frames. Using the extracted information from the input frame, we wish to overlay the resultant visualization.

2 Image Processing Pipeline

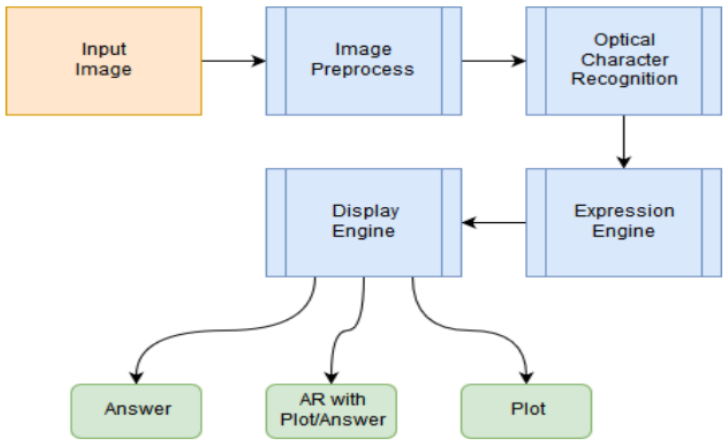

Figure 1: Overview of the Image Processing Pipeline

2.1 Preprocessing Steps

2.2 Symbol Classification

2.3 Expression Parsing

2.4 Visualization

3 CNN Architecture

We designed and implemented a convolutional neural network for symbol classification. We experimented and found the ReLU [1] activation function was optimal for fast convergence which reduced the overall training time without negatively impacting classification accuracy.

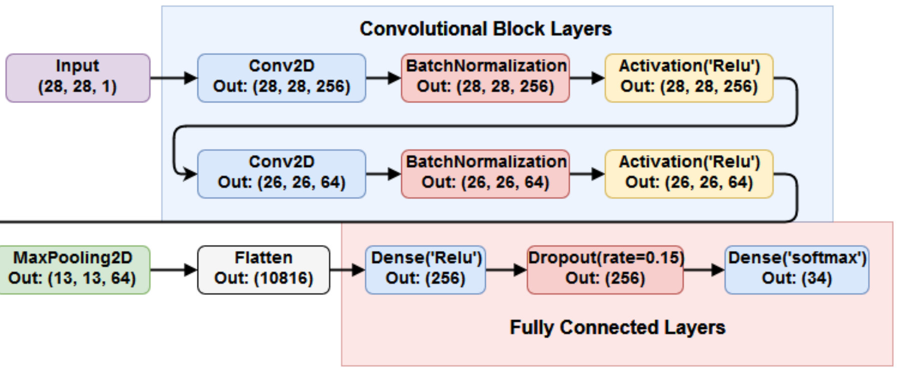
Convolutional Block Layers
Input (28, 28, 1)
Conv2D Out: (28, 28, 256)
BatchNormalization Out: (28, 28, 256)
Activation('Relu') Out: (28, 28, 256)
Conv2D Out: (26, 26, 64)
BatchNormalization Out: (26, 26, 64)
Activation('Relu') Out: (26, 26, 64)
MaxPooling2D Out: (13, 13, 64)
Flatten Out: (10816)
Dense('Relu') Out: (256)
Dropout(rate=0.15) Out: (256)
Dense('softmax') Out: (34)
Fully Connected Layers

Figure 2: Overview of the Convolutional Neural Network

3.1 Image Data Augmentation

We utilized the Keras image data generator API to dynamically generate additional training images. The goal was to improve the classifiers robustness to minor perturbations that result from our upstream connected components bounding box detection. We defined the data augmentation to be shifts in width, height, and zoom with a magnitude of 5%.

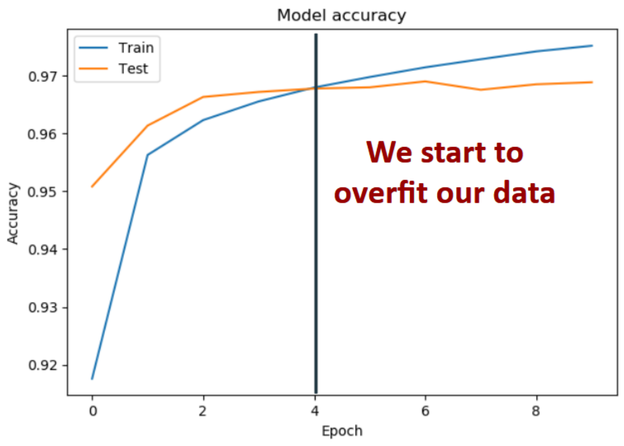

Figure 3: Our results without Image Data Augmentation

Data Augmentation turned out to be instrumental to achieving good classifier accuracy while reducing overfitting.

3.2 Batch Normalization

Batch Normalization [2] was introduced in 2015 as a way to improve both the training speed and results of neural networks. Per the recommendation in the paper, we utilized Batch Normalization after our convolution layers and before non-linearity (activation function).

3.3 Dropout

Dropout [3] was introduced in 2014 as a simple way to prevent overfitting in neural networks. We used Dropout in between the large fully connected layer and the final fully connected layer. While Dropout was helpful, we only needed to use a 15% dropout rate thanks to Batch Normalization.

4 Training

We selected both the EMNIST [4] and HASYv2 [5] datasets to train our convolutional neural network with. Our implementation allowed us to manually select the exact symbol classes we wanted from both datasets, and merged the images from both datasets into a train and test matrix. Since our labels were represented as integers that were mapped to the interval [0, num_classes], our training program produces a mapping dictionary used to obtain the actual symbol.

4.1 Datasets

4.1.1 EMNIST

Because we desired the capability to produce visualizations, we needed to support classification of digits and letters. The EMNIST dataset is an extension of the original MNIST dataset that includes a large sample of letters.

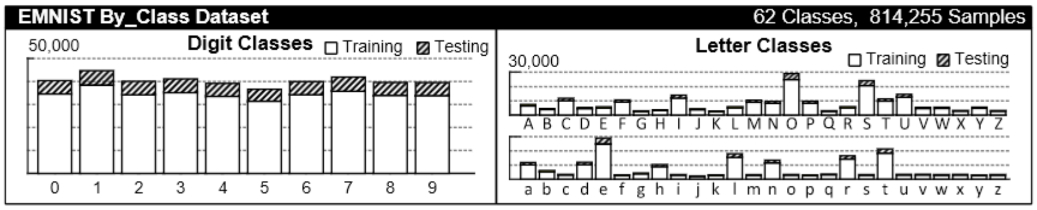

Figure 4: EMNIST Class Symbol Histogram [4]

4.1.2 HASYv2

The HASYv2 dataset was chosen because it already contains mathematical symbols we wished to include in our classifier. Originally we intended to include a large portion of the HASYv2 symbol classes in our training process but found after experimentation they negatively impacted classification accuracy.

4.2 Issues

To obtain the greatest number of training and test samples, we manually selected the classes from the *emnist-byclass.mat* file.

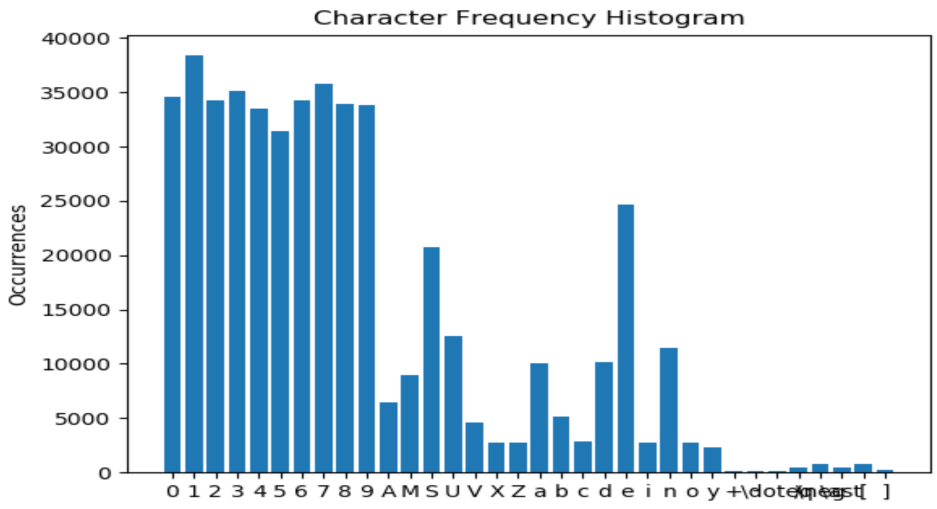

Figure 5: Symbol Class Distribution

Notice that we do not have a healthy representation of math symbols from the HASYv2 dataset. While the imbalance was partially improved by image data augmentation, we found that our results could have been better if we simply had more training samples of these math symbols. We explored other datasets but found the HASYv2 project provided the most streamlined integration with our project.

4.3 Results

We trained for 10 epochs because the accuracy stopped improving significantly as we continued. The training took roughly 20 minutes using a **NVIDIA GTX 1070ti** GPU.

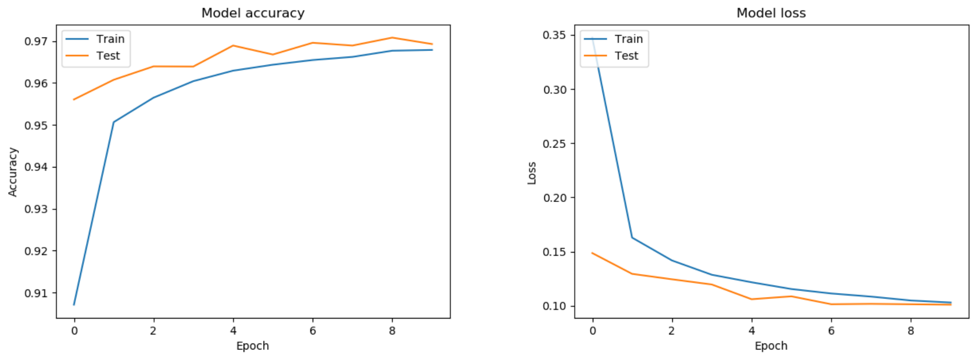

Figure 6: Training Results

Our accuracy and loss plots exhibit good learning rates and minimal overfitting. This is due to the aforementioned constraints on the number of symbol classes we desired to train with.

5 Results

6 Discussion

To the best of our knowledge, even the most sophisticated CNN architectures are capable of roughly 90% training accuracy on the EMNIST dataset. Because we constrained the number of classes used to train with, we obtained better accuracy. However this significantly limits the capabilities of our proposed system. In practice, support for all possible symbol classes would be necessary and improved preprocessing techniques would reduce misclassification.

Our research and implementation provides an excellent basis for future work. Our results include many limitations due to time constraints, but we believe the idea can be improved upon and eventually become a fully functioning system in classrooms around the world.

7 Conclusion

The latest advancements in image processing and machine learning research alongside hardware improvements allowed us to produce a proof of concept system for automatic mathematical expression processing and visualization. While now it is just an idea, eventually the state of the art will be sufficient for a system as proposed in this paper to be robust enough for real world use. Our inspiration is rooted in the idea that visualizations of mathematical concepts help students gain valuable intuition while recognizing that the instructors who teach the concepts typically don't have the resources or time to produce usable visualizations.

References

[1] Raman Arora et al. "Understanding Deep Neural Networks with Rectified Linear Units". In: *CoRR* abs/1611.01491 (2016). arXiv: `1611.01491`. URL: `http://arxiv.org/abs/1611.01491`.

[2] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015). arXiv: `1502.03167`. URL: `http://arxiv.org/abs/1502.03167`.

[3] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958.

[4] Gregory Cohen et al. “EMNIST: an extension of MNIST to handwritten letters”. In: *CoRR* abs/1702.05373 (2017). arXiv: `1702.05373`. URL: `http://arxiv.org/abs/1702.05373`.

[5] Martin Thoma. “The HASYv2 dataset”. In: *CoRR* abs/1701.08380 (2017). arXiv: `1701.08380`. URL: `http://arxiv.org/abs/1701.08380`.