Colin Short

CSCI 575

Final Project Report

Problem statement:

When doing scientific or engineering work, it is often a hassle to convert handwritten math to LaTeX. To address this, for my project I attempted to build a system that accepts images of handwritten math as input and returned usable LaTeX output.

Solution:

My solution to this problem was to use a Convolutional Neural Network to classify handwritten math symbols. The inputs are images of handwritten math. These inputs are then processed using computer vision methods to separate the image into images of the individual symbols. These individual images are then classified and the aggregate of the classifications is the output.

Assumptions, Constraints, and Implications:

For this project, I am assuming that the input images have high-contrast and clear backgrounds. Additionally, I am assuming that no symbols are used that are not present within the training dataset. Besides these, no other assumptions are required.

There are some major constraints that arose during this project. First, the computer vision processing of the input images is very limited due to my inexperience in computer vision. Because of this, the handwriting in images must be well-formed. To identify individual symbols, my project uses connected components. Some symbols like the letter ‘i’ are composed of two different connected components. My solution goes through the list of connected components and combines those that are close in proximity. The limitation that arises from this is the connected components in a symbol must be close enough together to be combined, but far enough away from other symbols. This results in some symbols with multiple connected components not being properly combined in the cases where the components of the symbol were not written close enough to each other. Another constraint is that the line width of the input image must not be too large. The images in the training dataset have very thin lines, so inputs with thicker lines are often misclassified. Additionally, the line darkness of input images is very important. Light or patchy-looking lines might confuse the connected component classifier or the neural network and result in the symbol being misclassified. Finally, the neural network is not trained to classify equation structures like subscripts or exponents. Therefore, it cannot represent these structures in the outputted LaTeX.

There are not many important implications of this project. One might be an increase in interest in collecting handwriting data. If this problem or similar problems grow large enough the demand for this type of data might lead to possible copyright issues with handwritten art or

manuscripts. Demand for any particular type of data may lead to privacy infringements, which is possible if there is a high demand for handwritten data.

Solution Architecture:

My solution consists of 4 primary components:

1. Data: The dataset I used is this Kaggle dataset. It consists of over 100,000 handwritten images of basic Greek letters, English alphanumeric symbols, math, and set operators, predefined math functions (e.g., sin, lim, log), and other math symbols. Each sample image is a 45x45 pixel jpg file. They are labeled by placing images into a folder whose name is their label. For example, all images that are labeled '\pi' are placed in the 'pi' folder.
2. Data processing: In this step, all of the data is read in and categorized into a data frame where each row is a sample and each column is a pixel value. The last column of each row is a numeric label. This data frame is then loaded into a pickle file. Additionally, a .csv file is produced that stores the numeric labels and their corresponding true, text value.
3. Convolutional Neural Network: Samples are classified using a convolutional neural network built using TensorFlow. Its structure is as follows:
   Input -> 2D Convolution Layer -> 2D Max Pooling Layer -> 2D Convolution Layer -> 2D Max Pooling Layer -> Fla7ening Layer -> Dense Layer (Output)
   The first 2D Convolution Layer has 16 channels and the second has 32. Both use 3x3 convolutions padded with 0s evenly on all sides and have ReLU activation functions. Both Max Pooling Layers use the TensorFlow default parameters.
   Here is the model summary from TensorFlow:
   Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 45, 45, 16)	160
max_pooling2d (MaxPooling2D)	(None, 22, 22, 16)	0
conv2d_1 (Conv2D)	(None, 22, 22, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 11, 11, 32)	0
flatten (Flatten)	(None, 3872)	0
dense (Dense)	(None, 82)	317586

Total params: 322,386
Trainable params: 322,386
Non-trainable params: 0

The model was trained over 5 epochs:
Epoch 1/5
9400/9400 - 279s 30ms/step - loss: 0.8680 - accuracy: 0.8206 - val_loss: 0.3813 - val_accuracy: 0.8929

Epoch 2/5
9400/9400 - 266s 28ms/step - loss: 0.2684 - accuracy: 0.9236 - val_loss: 0.2185 - val_accuracy: 0.9365

Epoch 3/5
9400/9400 - 261s 28ms/step - loss: 0.1597 - accuracy: 0.9524 - val_loss: 0.1658 - val_accuracy: 0.9546

Epoch 4/5
9400/9400 - 270s 29ms/step - loss: 0.1227 - accuracy: 0.9634 - val_loss: 0.1385 - val_accuracy: 0.9627

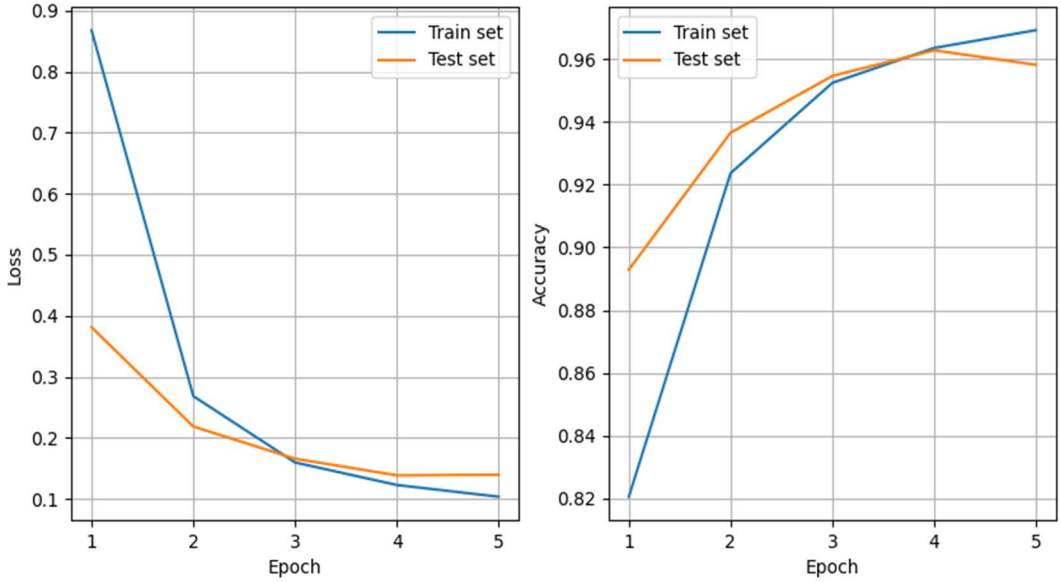
Epoch 5/5
9400/9400 - 262s 28ms/step - loss: 0.1035 - accuracy: 0.9691 - val_loss: 0.1395 - val_accuracy: 0.9581

The accuracy on the final epoch of training was 96.91%

Here are the evaluation stats:
2350/2350 - 21s 9ms/step - loss: 0.1395 - accuracy: 0.9581

The model achieved an accuracy of 95.81% on the test data.

The plots below visually represent the training and testing loss and accuracy over the 5 training epochs.



4. Computer Vision: The input images for this project require a lot of processing to be classified by the above convolutional neural network. This process is as follows:
    a. The image is binarized so that it is a grayscale image.
    b. A 3x3 Gaussian blur is applied over the whole image to soften any edges and imperfect lines.
    c. A binary threshold is applied to the image so that it only consists of black and white pixels.
    d. All connected components are identified and really small and really large ones are then filtered out. Each connected component is stored as a separate image
    e. Connected components that are really close to each other either horizontally or vertically are identified and combined. This resolves the issue of symbols that consist of more than one connected component.
    f. The connected component images are cropped down to the size of the bounding box around the connected component.
    g. The connected components images are padded with white pixels to make each image square. Then, they are resized to be 45x45 pixels.

Summary:

Unfortunately, this solution is not very effective on new handwritten images, as shown in the demo. The classification of the symbols in the image is often incorrect. Because of the high

training and testing accuracy scores for the neural network, I believe that issue is with the preparation of the input images. It seems that the line widths and darkness differing from the line widths and darkness of the training images makes a really big difference. Due to my lack of computer vision experience, I was not able to rectify this problem. This problem demonstrates that it is important to have really robust data preparation and training data so that irrelevant discrepancies like these do not cause a system to fail. Additionally, there are some datasets like this CROHME dataset that include math structure data, that could allow a system to recognize and classify mathematical structures. In general, this project demonstrates the need for robust data preparation and robust datasets to deal with problems that use extremely varied data like images of handwritten math.