Character to Text Conversion

Mr. Sumedh Walujkar (1001560040)

May 11, 2018

Abstract

This project helps to convert handwritten data to its subsequent digital form. In order to achieve this, we perform character segmentation on individual word to divide it into individual characters. Then we use convolution neural network to train a model and thereby accurately classify those characters(by passing segmented characters to the model). Once accurate predictions have been generated we reconstruct the entire word and display the classification result.

# Scope

Using the following, research an optical character recognition(OCR) system can be developed by considering multiple font styles. This research can also be respectively altered for character recognition of many other languages(for example Hangul, Hanja, mandarin, Navajo, Sanskrit, Basque etc.) Once a complete OCR has been developed for two or more languages then a converter can be implemented to convert sentences from one language to another through a transliteration and translation. Text to speech converter through OCR can also be achieved with the help of this following research.

# Introduction

The main inspiration for the project is to convert handwritten notes to digital formats automatically for future conservation. Lot of historic and prehistoric handwritten data can be converted into text which would ensure its prolonged existence and would thereby preserve knowledge. Also, a lot of automated mathematical operations could be performed on handwritten mathematical problems once they are converted to text.

* Input: Handwritten data
* Train the model using input data
* Output: Digital conversion

Initially, our goal will be to convert single character to text, Afterwards a single word to text.

# Method

In order to successfully achieve Character recognition, it was very necessary to work with a well established dataset which could be used to train our model. Thus we used “EMNIST: an extension of MNIST to handwritten letters” The EMNIST dataset is a set of handwritten character digits derived from the NIST Special Database 19 and converted to a 28x28 pixel image format and dataset structure that directly matches the MNIST dataset.

There are six different splits provided in this dataset. A short summary of the dataset is provided below:

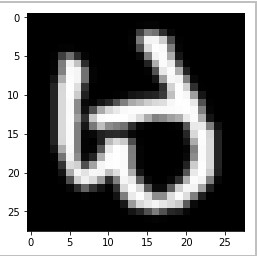


Figure 1:

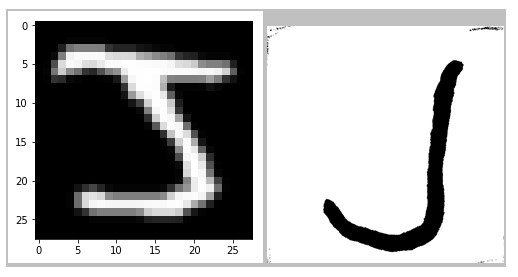


Figure 2:

EMNIST ByClass: 814,255 characters. 62 unbalanced classes.

EMNIST ByMerge: 814,255 characters. 47 unbalanced classes.

EMNIST Balanced: 131,600 characters. 47 balanced classes.

EMNIST Letters: 145,600 characters. 26 balanced classes.

EMNIST Digits: 280,000 characters. 10 balanced classes.

EMNIST MNIST: 70,000 characters. 10 balanced classes.

From these given datasets we used EMNIST letters to train our model. This dataset consists a total of 88800 data items ranging over 26 classes(Both uppercase and lowercase). We appended a NULL dataset(numpy.zeroes() of shape (784,)), which could be used to classify blank space(<spacebar>), which could be classified as class number “0”.

The following program picks up an image from real world. This image though, not be directly used for classification and thus must be preprocessed. Hence, we compressed the image to required size before using it. For experimental purposes we considered the newheight of each line to be 28(pixels). This is because each training instance is of dimension (784,1) that can be reshaped to (28,28) .

Once the desirable image is derived from resizing original image we segment it into subsequent subparts in a way that each segment consist of an individual character. We used Keras to train the model and all the training elements are reshaped of the form (88800,28,28,1). Same procedure is followed for the test data and the respective predictions are made. These predictions are grouped together at the end and displayed to the user.

# Experiment

When individual data item from emnist-letters-train.csv was displayed (by reshaping (784,1) to

(28,28)), the image formed was the transpose of the actual image which can be seen in the Figure 1 (above)

Thus in order to make accurate predictions we need to take the complements of the characters(which is derived from real world image). We can see that the dataset and the image data are color complements of each other. For example see the Figure 2 (above)

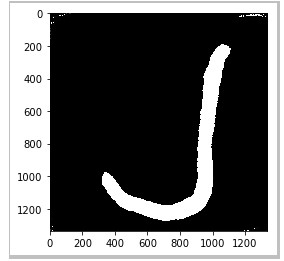


Figure 3:



Figure 4:

Therefore we solve this problem by changing the real world image by using the following formula Image = 255- img[ : ][ : ] See Figure 3.

Once the image is transformed into an appropriate color format we need to resize it to according to dimension of training elements. For this we consider the new-height as 28(pixels). Based on the formula in Figure 4.

We round of to get an integer so that we can divide the actual width into parts. We can segment the entire word into individual characters. Once all of this is done, we display all the individual predictions together thereby showing the final output to the user.

While shortlisting models we initially tried out naïve bayes algorithm which was not able to deliver well ,then we tried out K nearest neighbors which though outperformed naïve bayes but was still inefficient in terms of accuracy. On the contrary use of logistic regression and Multilayer Perceptron(MLP) performed much better then the other two, it took a lot of time to train.

Also, it was very sensitive to noise. Thus, we tried using Keras(convolution neural network(CNN)). This outperformed all other models with respect to both computational time and accuracy. The model was to generalize much better and was impervious to noise as well. Also the computation was done online and hence we decided to go with it.

Note: An attachment consisting of the SVM code we tried running has also been given. We were not able to use it because the training data size was too big and the algorithm along with the local system limitation was not able to use it.

# Conclusion

In this research we proposed to convert handwritten data to its subsequent digital form by performing character segmentation on individual words to divide it into corresponding characters.

We evaluated the performance of multiple machine learning classification algorithms like naive Bayes, K nearest neighbors, centroid, logistic regression, Multilayer Perceptron, convolution neural network. We thus found that CNN was much better in computational time and accuracy. Thus we decided to work on it.

# References

<https://www.nist.gov/itl/iad/image-group/emnist-dataset><http://cs231n.stanford.edu/reports/2017/pdfs/810.pdf><https://github.com/rohit-vg/SVM-KNN-Face-Recognition/blob/master/faceRec.py><https://github.com/siddhx/Deep-learning-python/blob/master/VGGnet_in_keras.ipynb><https://www.google.com/patents/US5420403><http://www.kurzweiltech.com/kcp.html><https://github.com/tesseractocr/tesseract/wiki/4.0-with-LSTM>