Handwriting Analysis using Graphology & Machine Learning

Jesús Alberto Martínez Mendoza

Adviser: Juan Carlos Cuevas Tello

I. INTRODUCTION

Graphology is the study of human personality through writing. It is a **projective personality test**, that let us know diverse personality traits. Our behavioral habits and reactions acquired through life experiences. Nowadays this science is used for a wide variety of purposes like criminology, recruitment, pedagogy, historical research, etc.

To make a diagnosis, several factors are used, such as text size, shape, direction, pressure on the paper, among others. A text can be analyzed as a set of these general qualities or by each character individually.

Each character has a meaning or portray something about the person, the goal of this project is to take these fixed and proven rules to give a general diagnosis.

The approach proposed is taking a photograph of a handwritten text, then using an photo OCR reader to extract a single character to be analyzed, obtain the best matches of it and apply these rules to make the analysis.

The project has so many challenges, the objective is to take the first steps in this field of human psychology with Machine Learning, the objective is that eventually this analyzer becomes as accurate as a professional psychologist.

In the example below we have a set of a handwritten character, each one has a different interpretation according to its features. The main approach of this project is to use a Machine Learning algorithm to correctly classify each character and give a correct valuation.

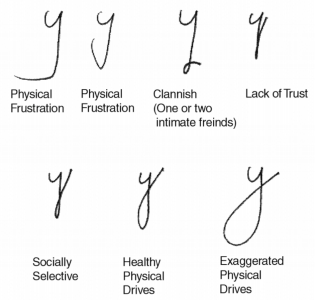

Fig. 1. Different interpretation according to its features

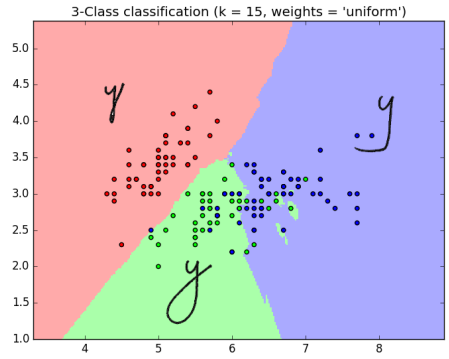

Fig. 2. Emulated classification of letter Y

II. MULTICLASS CLASSIFICATION

To attempt classification, one method is to use linear regression and map all predictions greater than 0.5 as a 1 and all less than 0.5 as a 0. However, this method does not work well because classification is not actually a linear function.

The classification problem is just like the regression problem, except that the values we now want to predict take on only a small number of discrete values.

Another possible approach is to use an One vs All logistic regression algorithm, basically choosing one class and then lumping all the others into a single second class. We do this repeatedly, applying binary logistic regression to each case, and then use the hypothesis that returned the highest value as the prediction. This approach works but it has a big drawback, it is too slow to compute and if the project wants to scale, a better technique is needed.

So the best approach for the classification problem is to use a Neural Network since computationally cheaper to learn a more complicated model.

Before using high level libraries like Tensorflow or Keras to build a neural network it is recommended to take into consideration two factors:

1. Become familiar with the data set; and
2. Understand how a neural network works.

III. DATA

Currently there is no classified data set for graphology, however the EMNIST data set is a perfect one as it contains about 145,600 letters in 26 balanced classes.

The tough task is to sub-classify the letters manually and create a model to identify the patterns of each letter. This work of re-classification must be done by a human.

A GNU Octave script was developed to convert the ubyyte data to PNG files. Having the images facilitates the task of sub-classification for a human.

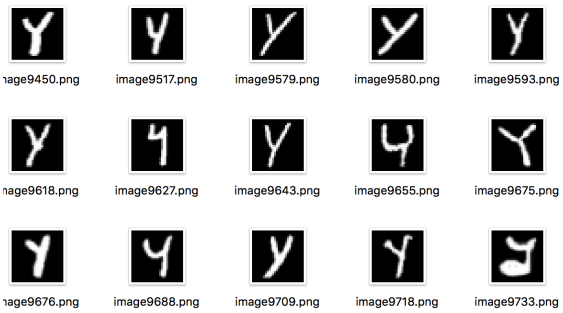

Fig. 3. Extract from EMNIST data set

IV. NEURAL NETWORK

First of all, I would like to explain how a Neural Network is shaped. According to Christos Stergiou and Dimitrios Sigano in their Neural Network paper *an artificial neuron is a device with many inputs and one output. The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not.*

So a program was developed to understand what is happening internally in those high level libraries, it consist in training a neural network with a simple architecture of one single hidden layer of N units.

The program made with GNU Octave 4.4.1 feed-forwards the neural network and return the cost, then calculates theta values using Back-propagation.

Results		
	EMNIST	MNIST
Training examples	4000	5000
Image size (pixels)	28x28	20x20
Iterations	50	50
Hidden units	60	25
Time to train (min)	16.8	5.4
Test accuracy	75.8%	97.5%

The neural networks were trained in a system with this specifications:

- Model: MacBookPro 12,1
- Processor: Intel Core i5, 2.7 GHz
- Core number: 2
- Caché level 2 (per core): 256 KB
- Caché level 3: 3 MB
- RAM: 8 GB

V. PHOTO OCR

A web system was developed for the very first prototype of this project. It consists of a module to upload the photograph of the handwritten text, then each symbol is extracted from the text using TesseractJS.

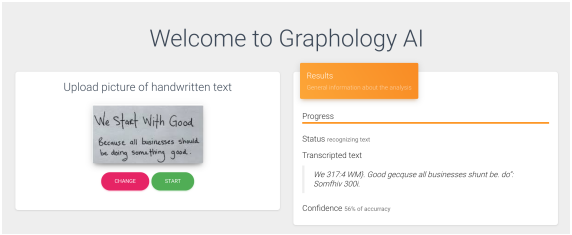

Fig. 4. Preview of web platform

This library supports over 60 languages, automatic text orientation and script detection, a simple interface for reading paragraph, word, and character bounding boxes. This library was chosen because unlike other Google Cloud OCR services, TesseractJS has a confidence variable for each symbol.

For a next iteration of the project it is planned to use Google OCR since its accuracy is about 95% as opposed to 65% of Tesseract.

Now that we have each symbol, we will take those with the greatest confidence of recognition, and pass them as input of our neural network to proceed with the graphological analysis.

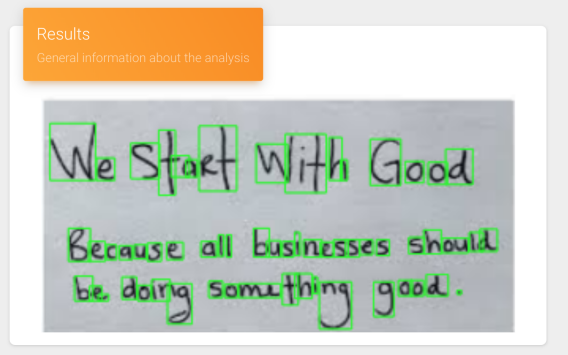

Fig. 5. Bounding boxes using TesseractJS

VI. SYMBOL RECOGNITION USING A CNN

Once we have the most accurate symbols, those are used as input for a LeNet-5 architecture.

The Convolutional Neural Network was built on top of Keras since it is very friendly to use. The figure 6 show the loss/accuracy results for 100 epochs, as we can see it has a lot of noise and does not properly converge. The main reason for the problem is the lack of data since we have only 40 samples per class. Other problem is the minimal variation between the classes, for this Neural Network the main characteristic that we try to classify is the second arc of the letter M. According to the graphology if the second arc is higher than the first one then the person who wrote this letter is more social worried about what others think about him.

Fig. 6. The left M represent a person not worried about their social circle

A. DATA AUGMENTATION

To solve the issue about lack of data, Keras provides an easy API to expand the batch of images using data augmentation. The augmentation consisted on zoom the images, rotate them, stretch them a little and add Gaussian noise over the whole image. For this particular case it is not recommendable to use a horizontal flip as it would toggle the positive class into our negative class.

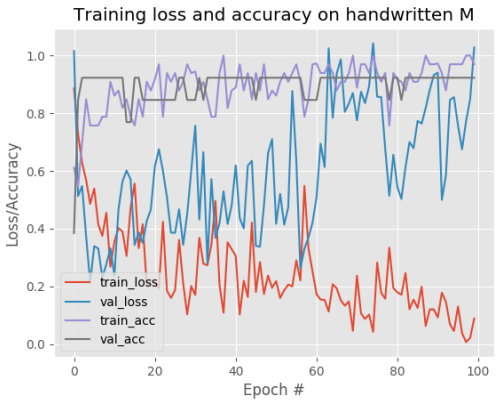

Fig. 7. Training loss and accuracy

VII. SYMBOL RECOGNITION USING A SIMPLER NEURAL NETWORK

Because the data set is very limited for a CNN, the next approach is to use a back-propagation algorithm with a simpler neural network.

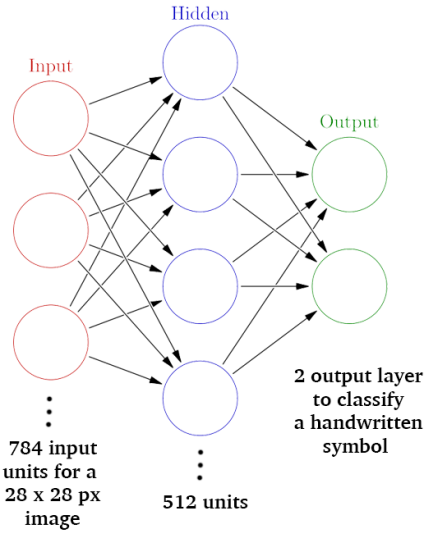

Fig. 8. Proposed Neural Network

The hypothesis is that the neural network will converge better with smaller data sets. Using a single hidden layer also helps to converge faster. The neural network will be built with Keras.

VIII. RESULT

Effectively, the neural network with only one hidden layer performed better than the LeNet CNN.
Once the desired results were obtained in Keras the next step was to migrate the model to TensorFlowJS and start with the originally proposed web platform.

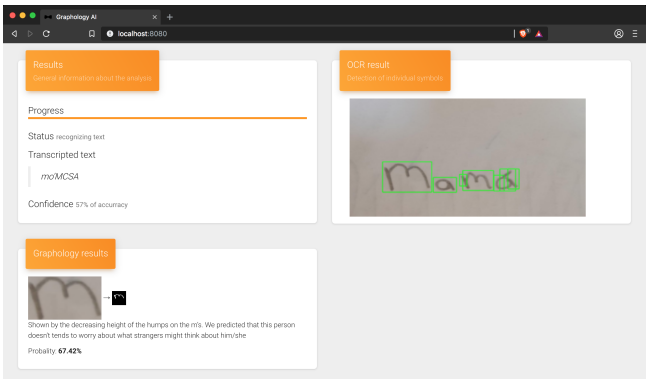

Fig. 9. Web page to analyze handwritten m

The platform is developed using regular web technologies, HTML, Javascript and CSS. In order to make the app very user-friendly the user just have to upload a handwritten word and the system automatically will split the symbols using OCR, apply several filters over the selected symbol and use it as input for the neural network. Finally the system will show the interpretation and the probability of accuracy.

IX. CHALLENGES AND SOLUTIONS

During the development of the project several challenges appeared. Here is a list with the most significant ones.

A. KERAS MODEL TO TENSORFLOWJS

The main platform runs in a web browser, it uses TensorFlowJS for the prediction, however the neural network was trained with Keras and the export is a *h5* file. This *h5* file can't be read by TensorFlow so it was necessary to use a converter to make it compatible with it.

B. APPLYING FILTERS

One disadvantage of changing from a convolutional neural network to a simple network was the missing filters of the CNN. To this particular project the filters were applied manually.
This part was one the most challenging during the development of the project. Mainly because the web browser doesn't provide a nice library like OpenCV so the filters and image processing were done using Canvas API.

X. FINAL THOUGHTS

Although the system is still somewhat basic and is currently only able to recognize and analyze the letter M, the objective was to prove that the proposed workflow (OCR, filters and analyzing) will work.

The project is open source and can be taken up by anyone interested in the topic of graphology with machine learning, it can be found in the following **repository**.

The works is divided in two, the python code is used to train the neural network and the code in Javascript is related to user interface.

A. IMPROVEMENTS

Right now the data set is very limited, it only has 26 samples per class. It is the best is to have more than 500 samples per class and thus be able to better detect the characteristics of the letter m.

For now the system only uses one single character to make the graphological analysis, but in the future is proposed to create a neural network for each symbol and thus make the analysis richer and closer to reality, increasingly closer to one given by a professional.

REFERENCES

[1] Bart A. Baggett, Handwriting Analysis Quick Reference Guide for Beginners, U.S., 2004
[2] Mauricio Xandró, Grafología elemental, Spain, 1994
[3] Chris Crawford. (2017). EMNIST (Extended MNIST). 2018, from Kaggle website: https://www.kaggle.com/crawford/emnist/version/1emnist-byclass-mapping.txt
[4] TesseractJS. (2018). Pure Javascript Multilingual OCR. 2018, from Tesseract website: http://tesseract.projectnaptha.com/
[5] TensorFlow. (2018). Importing a Keras model into TensorFlow.js. 2018, from Google website: https://js.tensorflow.org/tutorials/import-keras.html
[6] Adrian Rosebrock. (2017). Image classification with Keras and deep learning. 2018, from PyImage Search websote: https://www.pyimagesearch.com/2017/12/11/image-classification-with-keras-and-deep-learning/
[7] Lecun. (-). LeNet-5, convolutional neural networks. 2018, from LeCun website: http://yann.lecun.com/exdb/lenet/
[8] Ulises Ibarguren. (2016). Handwriting Analysis: Letter M and Interpersonal Relationships. 2018, from Handwriting and Graphology website: http://www.handwriting-graphology.com/handwriting-analysis-letter-m/