HANDWRITTEN TEXT RECOGNITION

A Project Report submitted in Partial Fulfillment of the Requirement for the Degree of

Bachelor of Technology

in

Information Technology

2016-2020

Submitted By:

Sneha Kedia

Roll No. [redacted]

Reg. No. [redacted]

Aayushi Shrivastava

Roll No. [redacted]

Reg. No. [redacted]

Under the Guidance of:

Prosenjit Mukherjee

Assistant Professor (Department of Information Technology)

(Affiliated to MAKAUT (Formerly WBUT))

June, 2020

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them.

We are highly indebted to our guide Professor ***Prosenjit Mukherjee*** for his guidance and constant supervisions well as for providing necessary information regarding the project and also for his support in completing the project. We express our thanks to our Principal ***Dr. Aloke Ghosh*** and our Head of the Department Professor ***Dr. Jagannibas Paul Choudhury*** for extending their support. We would also thank our Institution and the faculty members without whom this project would have been a distant reality.

Lastly, we thank our friends for their constant encouragement without which this assignment would not have been possible.

CERTIFICATE

I hereby certify that the work which is being presented in the project report entitled **'HANDWRITTEN TEXT RECOGNITION',** in partial fulfillment of the requirements for the award of the **Bachelor of Technology in Information Technology (IT)** and submitted to the Department of Information Technology of Future Institute of Engineering and Management affiliated to MAKAUT, West Bengal is an authentic record of my own work carried out during a period from July, 2019 to June, 2020 under the supervision of ***PROSENJIT MUKHERJEE*****, Supervisor, Assistant Professor, Department of IT, FIEM.**

The matter presented in this thesis has not been submitted by me for the award of any other degree elsewhere.

Sneha Kedia

Roll Number: [redacted]

CERTIFICATE

I hereby certify that the work which is being presented in the project report entitled **'HANDWRITTEN TEXT RECOGNITION',** in partial fulfillment of the requirements for the award of the **Bachelor of Technology in Information Technology (IT)** and submitted to the Department of Information Technology of Future Institute of Engineering and Management affiliated to MAKAUT, West Bengal is an authentic record of my own work carried out during a period from July, 2019 to June, 2020 under the supervision of ***PROSENJIT MUKHERJEE*****, Supervisor, Assistant Professor, Department of IT, FIEM.**

The matter presented in this thesis has not been submitted by me for the award of any other degree elsewhere.

Aayushi Shrivastava

Roll Number: [redacted]

TABLE OF CONTENTS

ABSTRACT

In todays' world advancement in sophisticated scientific techniques is pushing further the limits of human outreach in various fields of technology. One such field is the field of Handwritten Text Recognition (HTR).

Handwritten Text recognition has been one of the active and challenging research areas in the field of image processing and pattern recognition. Handwritten Text Recognition is defined as the task of transforming a language represented in its spatial form of graphical marks into its symbolic representation. The purpose is to categorize or classify data or object of one of the classes or categories.

This project is about developing an algorithm for recognition of Handwritten Text, also known as HTR (Handwritten Text Recognition) using Artificial Neural Networks (ANNs) including the schemes of feature extraction of the text.

In this project we developed an algorithm for the application of handwritten text identification using ANNs. The classifiers that we have selected for the classification tasks were: a pipeline of an RBM to extract the features and Multi-layer Perceptron (MLP) to classify, an MLP using HOG (Histogram of Oriented Gradients) features, an MLP using PCA (Principal Component Analysis) and an MLP. We have implemented the classification process using the scikit-learn library. We have learned the classifiers using the train data and computing various metrics in the test data. From our testing the best results were produced by the MLP with PCA features and the MLP.

INTRODUCTION

In this fast paced world there is an immense urge for the digitalization of printed documents and documentation of information directly in digital form. And there is still some gap in this area even today. Handwritten Text Recognition techniques and their continuous improvisation from time to time is trying to fill this gap.

Handwritten Text system is commonly used system in various applications, and it is a technology that is a mandatory need in this world as of now. Before the correct implementation of this technology we have dependent on writing texts with our own hands that result in errors. It's difficult to store, access physical data and process the data in efficient manner. Manually it is needed to update, and labour is required in order to maintain proper organization of the data. Since for long time we have encountered a severe loss of data because of the traditional method of storing data.

Modern day technology is boon, and this technology is making people to store the data over machines, where the storage, organization and accessing of data is easier. Adoption of the Handwritten Text Recognition software is a practical idea and, it is easier to store and access data that was traditionally stored. Furthermore, it provides more security to the data.

This project, 'Handwritten Text Recognition' is a software algorithm project to recognize any hand written character efficiently on computer with an input image that has handwritten text. It is about devising an algorithm for recognition of hand written characters leaving aside types of OCR that deals with recognition of computer or typewriter printed characters.

A novel technique is proposed for recognition of English language characters using Artificial Neural Network including the schemes of feature extraction of the characters and implemented.

OBJECTIVE

The goal of the project is to solve the task of text transcription from handwriting images implementing a NN approach and using a database with a large number of images of handwritten text.

The dataset includes over 400,000 images of handwritten text along with human contributors' transcription of these written text.

Since we have the transcription of the names this is a ***supervised learning*** problem.

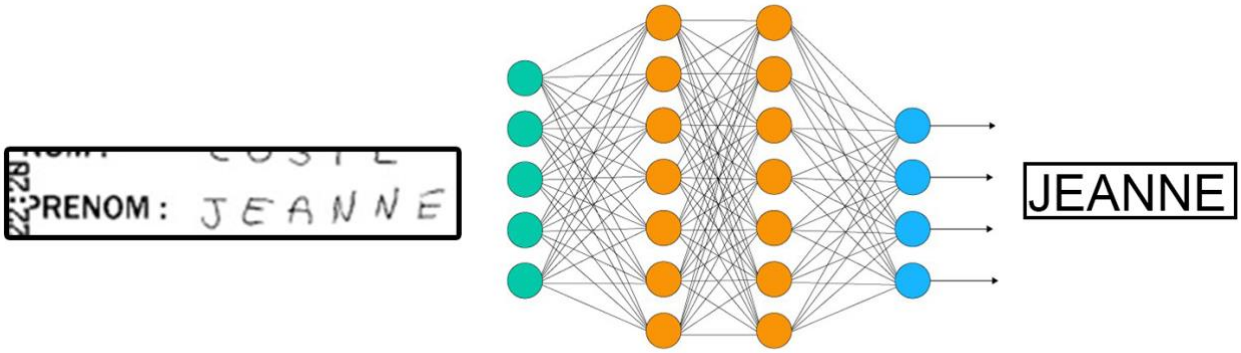

Fig: Idea of the project, Digital Text from Handwritten Text by applying NN

PROGRAMMING LIBRARIES

All the project steps were implemented in Python 3. We used the following python libraries to achieve our goals:

- numpy
- matplotlib
- joblib
- urllib
- plotly
- scipy
- scikit-learn
- scikit-image
- pandas
- cv2 (OpenCV)

We used pandas and urllib for reading the original database and getting the images from the servers, scikit-image, cv2 (python3-opencv) and scipy for preprocessing the dataset, matplotlib and plotly for plotting different data and scikit-learn for the classification tasks.

IMPORTING THE DATASET

We import the "Transcriptions of handwritten text" database obtained from https://appen.com/datasets/handwritten-name-transcription-from-an-image/. This dataset consists of more than four hundred thousand handwritten names collected through charity projects to support disadvantaged children around the world.

The database has the following characteristics:

- 1 unique identifier per entry on the database.
- 1 URL to the corresponding image per entry.
- 1 transcription of the name per entry.
- 1 label indicating if it's a first name or a last name per entry.
- The images don't contain only the text and sometimes contain more data that needs to be cut or deleted, the transcriptions of the text sometimes are missing, are not correct or are repeated in multiple lines.

As the csv file in the website has some errors, we fixed those errors using a small program written in Java (included in java file: database_fix.java). The data set, when imported into the Jupyter Notebook, looks as follows:

	0	1	2	3
0	_unit_id	image_url	transcription	first_or_last
1	952459271	http://crl.checkbacksoon.nl/dls/dss/d2m/15/fir...	HIMELIN	first
2	952459272	http://crl.checkbacksoon.nl/dls/dss/d2m/15/fir...	ROBIN	first
3	952459273	http://crl.checkbacksoon.nl/dls/dss/d2m/15/fir...	YOANN	first
4	952459274	http://crl.checkbacksoon.nl/dls/dss/d2m/15/fir...	MARTIN	first

Fig: Dataset in Jupyter Notebook

Each input image looks as follows:

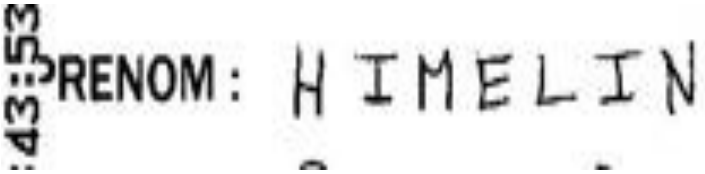

Fig: Sample Input Image from the Dataset

APPROACH

To solve the defined handwritten character recognition problem of classification, we are going to use various Artificial Intelligence Tools like Machine Learning, ANN, along with Deep Learning. The general idea is to divide the entire task into multiple subdivisions and carry them out step by step.

- Preprocessing of the dataset
- Preprocessing of the images
- Feature extraction
- Classifiers
- Inference
- Validation

PREPROCESSING OF THE DATASET

The original dataset has many format problems, so the first step was to solve this. We have two types of images, some in which the text is written next to the NOM or PRENOM word and others which the text is written under that words.

Fig: Two examples of the different type of images in the dataset

In the second type of images the transcribed text is duplicated in many lines, there are also labels that are duplicated only in the line below that belong to the first type of images.

```
952482140,"http://crl.checkbacksoon.nl/dls/dss/d2m/15/first_names/D2M_15_0460046A_0079_22687_first_name.jpg","ARCHIE","first"
952485740,"http://crl.checkbacksoon.nl/dls/dss/cf/test_questions/first_names/DM_3_0010008D_0021_20_first_name.jpg","LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE
LORIE","first"
```

Fig: One of the duplicated entries in the dataset

We solved this with a small program made in Java that deletes the duplicated text and modifies the last label to indicate if the name is positioned below the NOM/PRENOM word or next to it. In case the text is located on the right the last label will have the value "first" or "last" and in case that the text is located below it will have the value "first_b" or "last_b". The dataset is with the name first_and_last_names_fix.csv.

PREPROCESSING OF THE IMAGES

The pre-processing is a series of operations performed on the scanned input image. We used the pandas library to import the dataset from the CSV file. The dataset does not contain the images itself so we need to download them.

❖ **Noise Reduction:**

In the images apart from the text that we want to transcribe we have a lot of noise that we want to erase, for example the code of the image, the word NOM: or PRENOM: or some split characters that are part of other text. NOM or PRENOM words appear in all the images and because they are written in a machine and then printed, they have always the same exact form, so we extracted as reference the last two characters of this words from one image of the database. We used a

template matching algorithm from the OpenCV library using this reference image that locates the NOM or PRENOM word. After this is done, we use the information about the location of the word to crop the image and extract the name. However, with this procedure apart from the name we can also crop some noise like parts of other names written above or below. Because the images are taken with the name centered, it would never touch the corners so to remove the noise we find any figure that is touching the lower border of the image and we delete it.

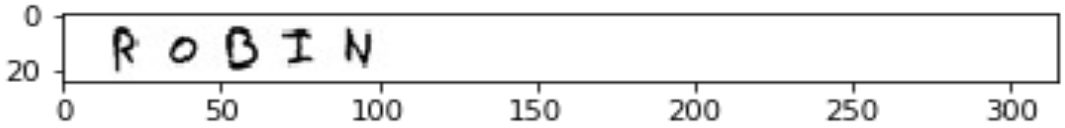

Fig: Scaled image obtained after cropping and elimination of objects

❖ Binarization:

Now we have the image that contains only the text that we want to recognize. The next step of the procedure if the character extraction. This is done with a clustering algorithm. First we binarize the image, then using clustering we calculate the different connected components of the binarized image When this is done we calculate the coordinates of every components to extract the characters, but the problem is that not every character must be a unique component, for example the point of an “i”, of one of the lines of an capital ”E”, so for calculating this coordinates we take in account that if a component is above other component both of them belong to the same character. The characters then are extracted and rescaled to a 28x28 image; this are the images that we will use to train our Neural Networks.

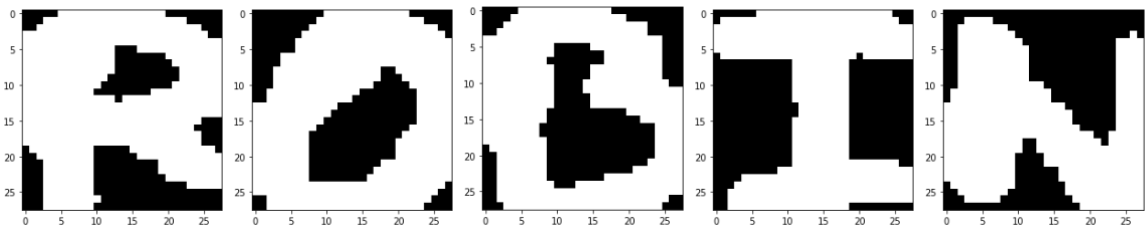

Fig: Image results after character extraction and binarization

This procedure does not always work correctly, sometimes we obtain a different number of characters that the number of letters that the text has in the label, this can happen due to noise, characters written split, a bad label of the name. Nevertheless, we are able to obtain the same number of characters that the text label has with approximately an 94% of success rate. The remaining 6% of the text are not added to the train or test data, because we know that we will train with data that is wrong and we also know that if we try to test the result with it, it would be a labeled as failure. This inconsistent texts are added to another list so after having trained the neural networks we can try to extract some valuable information from them, for example try to test with of the images extracted are characters and which are noise so we can add to the database the images that we recognize as characters.

We have no way of checking if the character extraction has been done properly in the data that we will use for training and testing, but based on the results of the project we think that we have an enough amount of properly extracted characters to successfully train our Neural Networks.

The data is split into two different sets: train and test, for validation. We use the same train set to learn all the classifiers, and the same test set for evaluating their accuracy.

FEATURE EXTRACTION

In this project we have used various types of feature extraction to test which of them gives the best results.

- **Restricted Boltzmann Machine:**

Restricted Boltzmann Machines are a two-layered artificial neural network with generative capabilities. RBMs are a special class of Boltzmann Machine and they are restricted in terms of the connections between the visible and the hidden units. As stated earlier, they are a two-layered neural network (one being the visible layer and the other one being the hidden layer) and these two layers are connected by a fully bipartite graph. This means that every node in the visible layer is connected to every node in the hidden layer but no two nodes in the same group

are connected to each other. This restriction allows for more efficient training algorithms than what is available for the general class of Boltzmann machines, in particular, the gradient-based contrastive divergence algorithm. The Restricted Boltzmann Machines contains a set of latent variables that represent higher order patterns present in the data. They are commonly used in the character recognition task. This is an example of the features learned by our RBM:

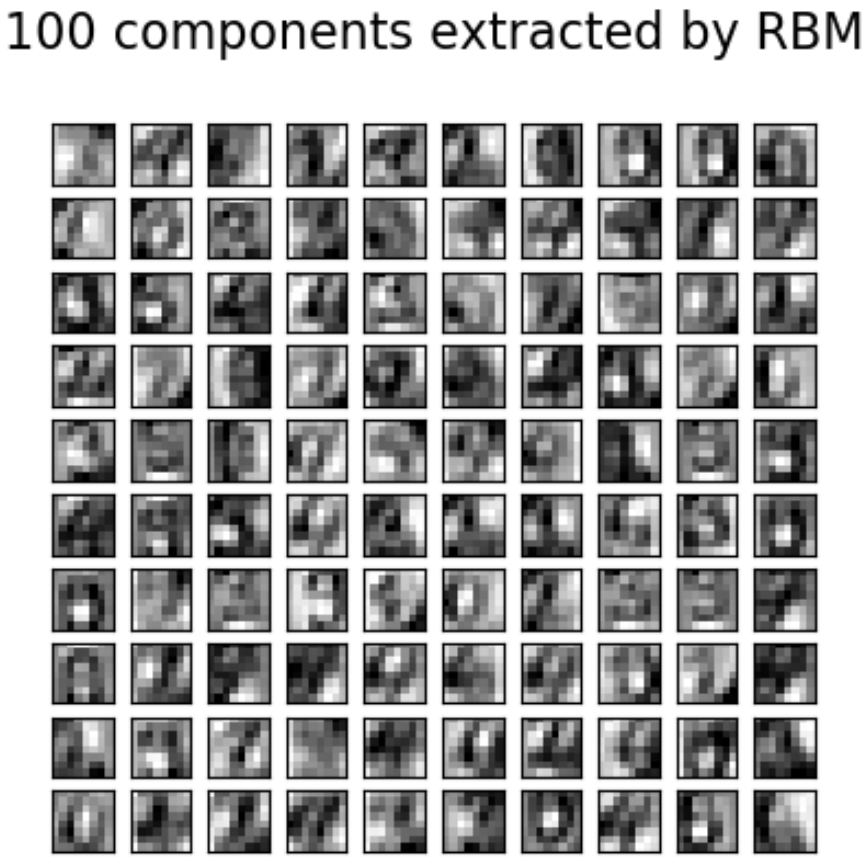

Fig: Features learned by the RBM

- **Histograms of Oriented Gradients:**

This technique uses the distribution of directions of gradients as features. Gradients of a image are useful because the magnitude of gradient is large around edges and corners which gives us a lot of information about objects shape. The Histogram of Oriented Gradients (HOG) is an efficient way to extract features out of the pixel colors for building an object recognition classifier. With the knowledge of image gradient vectors, it is not hard to understand how HOG works.

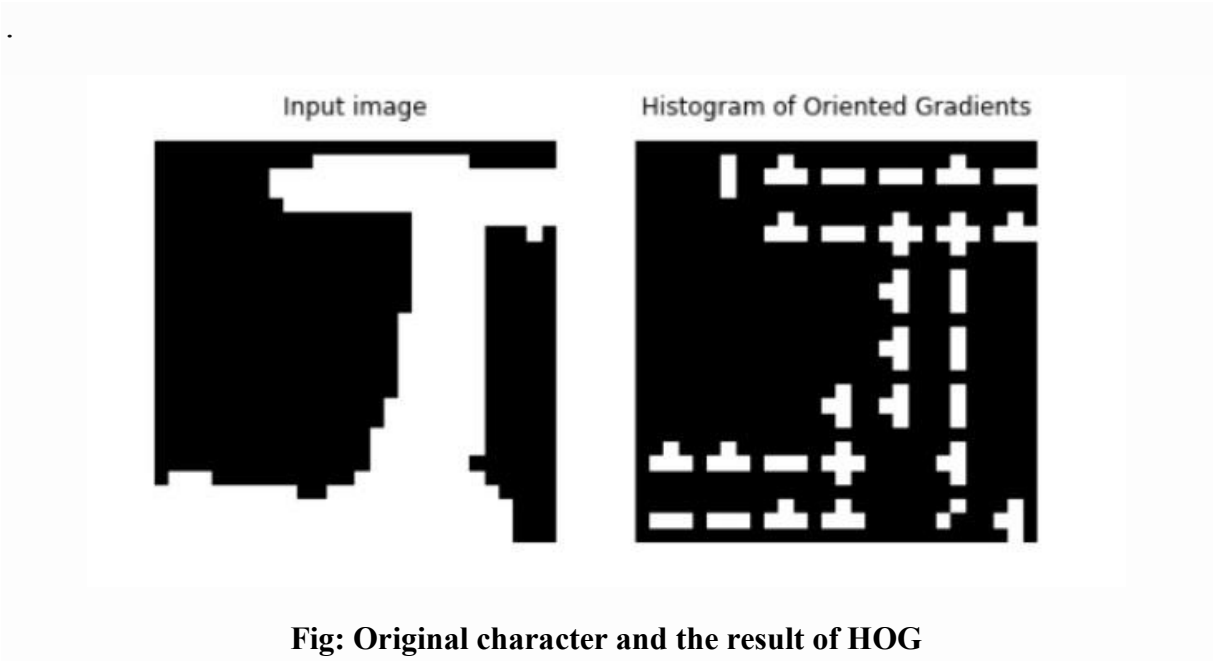

Fig: Original character and the result of HOG

❖ Principal Component Analysis:

Principal component analysis (PCA) simplifies the complexity in high-dimensional data while retaining trends and patterns. It does this by transforming the data into fewer dimensions, which act as summaries of features. High-dimensional data are very common in biology and arise when multiple features, such as expression of many genes, are measured for each sample.

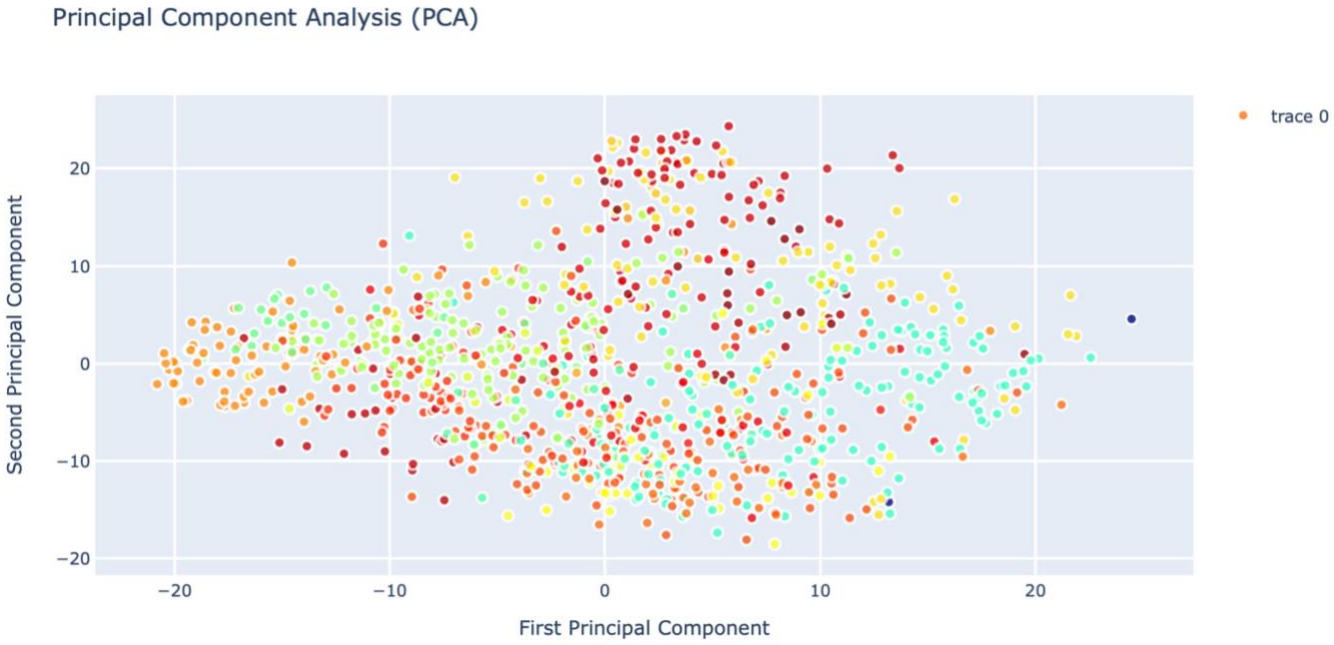

Fig: PCA of the first two features

This type of data presents several challenges that PCA mitigates: computational expense and an increased error rate due to multiple test correction when testing each feature for association with an outcome. PCA is an unsupervised learning method and is similar to clustering—it finds patterns without reference to prior knowledge about whether the samples come from different treatment groups or have phenotypic differences.

❖ No Feature Extraction:

For the purpose of evaluating the previous methods we also trained our neural networks without any type of feature extraction, that is, using the binarized image of the characters.

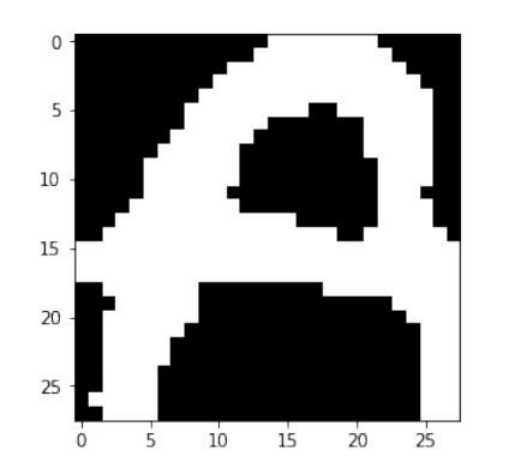

Fig: A binarized character without feature extraction

CLASSIFIERS

A classifier is a hypothesis which are often learned with Machine Learning Algorithms. It is also a discrete valued function that is used to assign class labels to particular data points. The learning algorithm for the RBM is the Stochastic Maximum Likelihood and learning algorithm for all the MLPs is the default in Scikit-Learn: Adam optimizer.

We use four different classifiers with the following parameters:

1. Pipeline of RMB for feature extraction and MLP for classification:

- RMB:
  - n components = 300
  - learning rate = 0.01
  - n iterations = 45
- MLP:
  - layers = (300, 400, 150)
  - activation function = ‘relu’
  - max iterations = 5000
  - tol = 0.0001

2. MLP classifier with HOG (Histogram of Oriented Gradients) features:
   - layers = (300, 400, 150)
   - activation function = ’relu’
   - max iterations = 5000
   - tol = 0.0001

3. MLP classifier with PCA (Principal Component Analysis) features:
   - PCA:
     - n components = 100
   - MLP:
     - layers = (300, 400, 150)
     - activation function = ’relu’
     - max iterations = 5000
     - tol = 0.0001

4. MLP classifier with character images directly:
    - ❖ layers = (300, 400, 150)
    - ❖ activation function = 'relu'
    - ❖ max iterations = 5000
    - ❖ tol = 0.0001

INFERENCE

After some research we found that MLP was the most popular network for this task and that it commonly produces good results, which is why we chose this type of NNs. There are other more advanced methods used for HTR such as CNNs, but the objective of this project was solving the problem using an NN approach excluding Deep Learning.

Once the Neural Networks are trained the process to predict new names is very similar to the process to train it. The new image must follow the same process that the images used for training, that means, we need to extract each individual character on the image. Then, apply the feature extraction that we want to use, for example calculate the HOG or calculate the state of the RBM for this image. Then the MLP predicts a letter for each individual character. The final step is to put the values predicted by the NN together to form the name.

VALIDATION

To validate our results, we compute the scikit classifier metrics and our own full name prediction metrics in the test data. Another possibility was to compute the cross-validation in the complete dataset but we used the split between train and test because it was simpler. We divided the dataset in 80% train 20% test batches. We computed the classification report of Scikit metrics (which gives the precision, recall and f1- score for each classifier) for individual character prediction. We also tested full name recognition and we output the full correct name ratio and the correlation ratio (how similar all predicted names are to the original label) for each classifier.

IMPLEMENTATION

The image below represents the process that we follow to train our network.

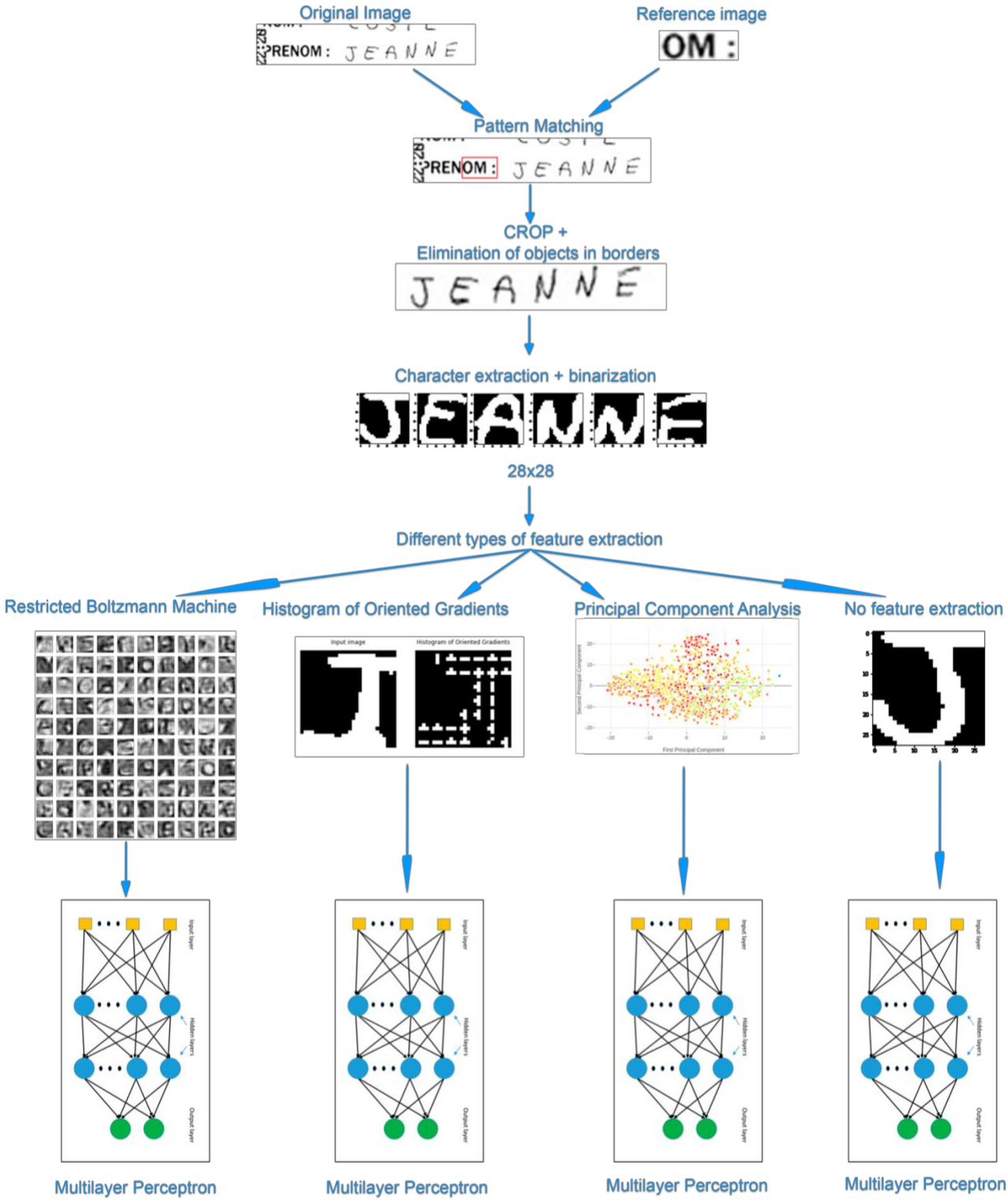

Fig: Process Diagram

Downloading all the database and training the Neural Networks takes a long time, so in the Jupyter Notebook of the project, we added a database with 10,000 images downloaded and pre-trained classifiers with that database. At the beginning of the notebook there are some global variables that allow the user to choose how to run the program.

1. delborders- The *delborders* function eliminates the bottom border noise in case of "first_b" and "last_b" types in the dataset.
2. getcrop- The *getcrop* function obtains an image from an entry of the database and returns the image in grayscale and cropped to show just the name part.
3. gen_dataset- The gen_dataset function, this function creates a binarized image list "*data*" and a label list "*labels*" from the first _n_ entries of the database (if not specified uses all the entries of the database) using the crops of the *getcrop* function and its corresponding label on that index.
4. get_labels- This function labels the connected components in an image by binarizing it and running a clustering method, it returns the labels and the number of labels it detects.
5. get_bboxes- This function gets the bounding boxes to cut each character correctly given the labels obtained from get_labels. It returns a list of each character's bounding boxes (2 2D points).
6. crop_characters- Given an image and character bounding boxes this function crops each character in an image and returns each character's corresponding binarized image in a list.
7. labelsep- Separates a full name label into a character list. Useful for the training part to have the labels of each character.
8. get_characters- Given an image from the dataset and its label this function splits each character into one image and a label. The img_only variant doesn't return the labels (useful when testing with full names to save some memory and time).

RESULTS

We ran our testing for the first 30000 images in the dataset. The precision produced by the RBM-MLP Pipeline, MLP with HOG features, MLP with PCA features and MLP classifiers in individual character recognition were, respectively: 0.92, 0.89, 0.92 and 0.93. The full name correct ratio produced by the RBM-MLP Pipeline, MLP with HOG features, MLP with PCA features and MLP classifiers were, respectively: 0.664, 0.579, 0.688 and 0.709. The full name correlation ratios (similarity of name produced to original name) produced by the RBM-MLP Pipeline, MLP with HOG features, MLP with PCA features and MLP classifiers were, respectively: 0.923, 0.899, 0.929 and 0.934. Therefore, the best classifiers were the MLP with PCA features and the MLP. The prediction results are as follows:

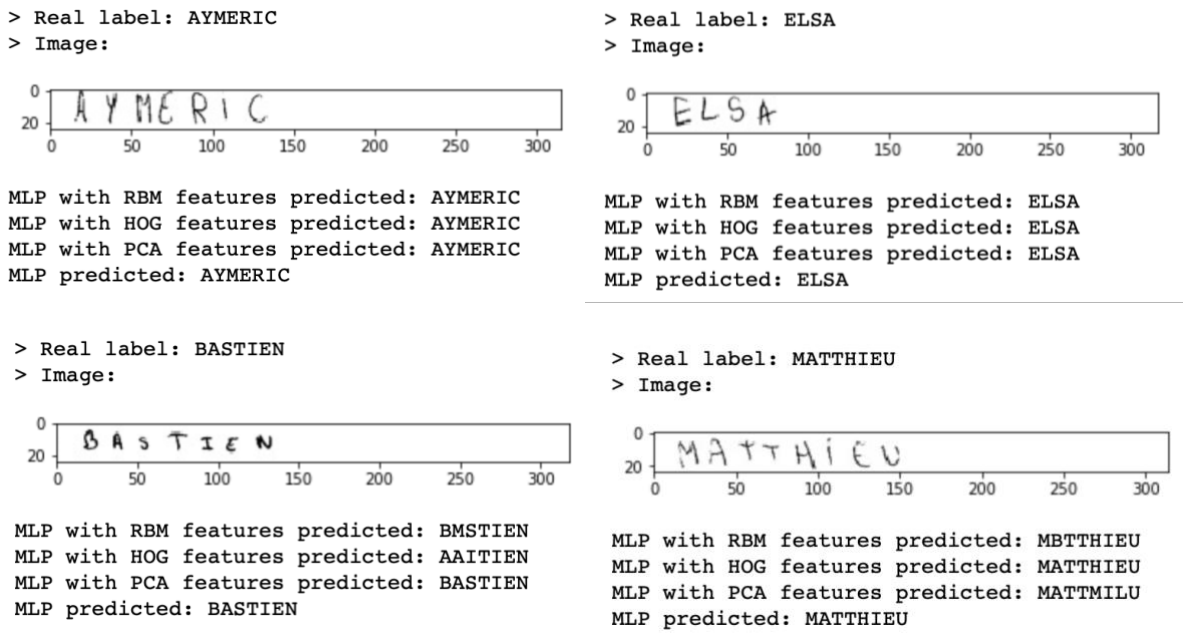

Fig: Prediction Results

The results of the computation of the Scikit metrics for the RBM-MLP Pipeline, MLP with HOG features, MLP with PCA features and MLP classifiers are respectively shown in Tables below.

Train	Names:	23,945	Characters:	131,759
Test		5,987		33,211
Precision	RBM + MLP	HOG + MLP	PCA + MLP	MLP
-	0.50	0.70	**0.74**	0.70
A	**0.94**	0.91	0.93	**0.94**
B	0.85	0.76	**0.86**	0.78
C	0.92	0.90	**0.93**	0.92
D	0.86	0.80	**0.88**	0.86
E	0.94	0.93	0.95	**0.96**
F	**0.70**	0.66	0.65	0.64
G	0.86	0.83	**0.91**	0.90
H	0.84	0.81	**0.88**	0.87
I	0.91	0.90	0.91	**0.93**
J	0.88	0.89	**0.90**	0.89
K	0.85	0.75	**0.90**	0.88
L	**0.96**	**0.96**	**0.96**	**0.96**
M	0.87	0.84	0.86	**0.89**
N	0.92	0.90	**0.95**	0.94
O	0.91	0.90	0.91	**0.94**
P	0.81	0.72	**0.85**	0.78
Q	**0.86**	0.62	0.84	0.81
R	0.89	0.83	**0.92**	0.91
S	0.93	0.92	**0.95**	**0.95**
T	0.93	0.91	0.92	**0.94**
U	0.92	0.91	0.93	**0.95**
V	0.88	**0.89**	0.82	0.88
W	**0.90**	0.68	0.81	0.83
X	**0.94**	0.82	0.91	**0.94**
Y	0.93	0.86	**0.94**	0.93
Z	0.82	0.83	**0.93**	0.87
TOTAL	0.92	0.89	0.92	0.93

Fig: Individual character classification results

The full name correct ratios and correlation ratios are shown in the table below.

Classifier	Correct percentage	Correlation ratio
MLP with RBM features	0.664	0.923
MLP with HOG features	0.579	0.899
MLP with PCA features	0.688	0.929
MLP only	**0.709**	**0.934**

Fig: Full name prediction results

The percentage of correctly classified names is not too high, but the correlation of the predicted names with the real names the number is really high, that means what the classifier predicts is very similar to the real name, so the names that are not correctly classified are probably wrong in very few characters.

CONCLUSION

In our project we applied a multilayer perceptron combined with different types of feature extraction to the “Transcriptions of names from handwriting” dataset. We have computed the accuracy of this Neural Networks and we observed that surprisingly the simplest implementation, the MLP without feature extraction produces the highest accuracy. Finally, as future upgrades in this implementation we did not do anything with names where the character extraction gives us inconsistent results. Even so the program stores in a list all these names to make possible to work with them in the future. Due to computational limitations we were not able to train with the full database, training with more names could help to improve the results. Also, fine tuning of the parameters of the neural networks could improve the results but computational power is again a limitation.

REFERENCES

1. Appen: Open Source Datasets - Handwriting Recognition (Transcriptions of 400,000 handwritten names), 2020

   [https://appen.com/datasets/handwritten-name-transcription-from-an-image/]

2. Dataset Fix Reference from Cornell University Publications, I. García [https://arxiv.org/search/cs?searchtype=author&query=Garc%C3%ADa%2C+I]

3. Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.

4. OpenCV. OpenCV documentation: Template matching [https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html].

5. Scipy Lectures. Labelling connected components using scikit-image: [http://www.scipy-lectures.org/packages/scikit-image/auto_examples/plot_labels.html].

6. Scikit-Image. Scikit-image documentation:

   [http://scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.label].

7. Dewi Nasien Muhammad 'Arif Mohamad, Haswadi Hassan and Habibollah Haron. A review on feature extraction and feature selection for handwritten character recognition. *International Journal of Advanced Computer Science and Applications(ijacsa)*, 6(2), 2015.

8. Anisotropic. Interactive intro to dimensionality reduction, 2017. [https://www.kaggle.com/arthurtok/interactive-intro-to-dimensionality-reduction/notebook]

9. Satya Mallick. Learn OpenCV: Histogram of oriented gradients [https://www.learnopencv.com/histogram-of-oriented-gradients/].

10. Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.