Survey on Handwritten Name Recognition Features Using ANNs

Dr. Seema S, Shruti M Chavan

Dept of CSE, Ramaiah Institute of Technology, Bangalore

ABSTRACT: *The Handwritten Name Recognition proposed in this paper gives the application of names recognition using ANNs (Artificial Neural Networks). The classifiers that are selected for the classification tasks area pipeline of a RBM (Restricted Boltzmann Machine) to extract the features and MLP (Multi-Layer Perceptron) to classify, MLP using HOG (Histogram of Oriented Gradients) features, MLP using PCA (Principal Component Analysis) and only MLP. We have implemented the classification process using the scikit-learn library. We have learned the classifiers using the training data and computing various metrics in the test data. By the testing results, we got the best results produced by the MLP and MLP with PCA features.*

Keywords: Restricted Boltzmann Machine, Multi-Layer Perceptron, Histogram of Oriented Gradients, Principal Component Analysis

I. INTRODUCTION

The recognition of handwritten names from the scanned images has been a problem that received much attention in the fields of pattern recognition, image processing, and the artificial intelligence. The classifiers that are used for classification in this proposed paper are: RBM (Restricted Boltzmann Machine) are excellent feature extractors, working just like auto encoders. They easily outperform PCA (principal component analysis) and LDA (Linear Discriminant Analysis) for dimensionality reduction techniques. The features that are extracted from images are fed to neural networks or machine learning algorithms such as SVM (Support Vector Machine) or Logistic regression classifiers. RBMs only work on inputs between 0 and 1 and even works very well on binary inputs. MLP (Multi-layer perceptron's) are a popular form of feedforward artificial neural networks with many successful applications in data classification. The MLP used for character recognition provides excellent level of accuracy and it will take minimum time for training the network.

1) Problem Description

The goal of the project is to solve the task of name recognition from the handwritten images implementing a NN (Neural Network) approach and using a database with a large number of images of handwritten names, the dataset includes over 125,000 images of handwritten names along with human contributor's transcription of these written names.

2) The dataset has the following characteristics:

- 1 unique identifier per entry on the dataset.
- 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 the last name per entry
- The images don't contain the names only and it may contain more data that needs to be cut or deleted, the transcriptions of the names sometimes are missing, are not correct or are repeated in multiple lines.

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

II. DESCRIPTION OF OUR APPROACH

We organized the implementation of the project according to the following tasks:

1. Pre-processing of the dataset
2. Pre-processing of the images
3. Feature extraction
4. Classifiers
5. Inference
6. Validation

A. Pre-processing 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 name is written next to the first name or last name word and others which the name is written under that words. In the second type of images the transcribed name is duplicated in many lines, there are also labels that are duplicated only in the line below that belongs to the first type of images.

We solved this with a small program that deletes the duplicated names and modifies the last label to indicate if the name is positioned below the first name or last name

word or next to it. In case the name is located on the right the last label will have the value ”first” or ”last” and in case that the name is located below it will have the value ”first b” or ”last b”.

B. Pre-processing of the images

We used the panda’s library to import the dataset from the CSV file. In the images apart from the name that we want to recognise we have a lot of noise that we want to erase, for example, the word first name or last name or some split characters are part of other names. First name or last name 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 these words from one image of the database. We used a template matching algorithm from the OpenCV library [2] (Histograms and Matching) [3] using this reference image that locates the first name or last name 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.

After this what we have is an image that contains only the name that we want to recognize. The next step of the procedure is the character extraction. This is done with a clustering algorithm. First, we get the binaries of the image, then using clustering we calculate the different connected component of the binarized image [4] [5]. When this is done we calculate the coordinates of every component to extract the characters. The extracted characters are then rescaled to a 28x28 image; these are the images that we will use to train our Neural Networks. Due to the noise, split characters or by the bad label of the name present in the dataset we may get extra characters which are not present in the recognised name in the label.

Nevertheless we are able to obtain the same number of characters that the name label has with approximately 94% of success rate. The remaining 6% of the names are not added to the trained or test data, because those data might be wrong and we also know that if we try to test the result with it, it would be labelled as failure. These inconsistent names are added to another list so that after having trained the neural networks we can try to extract some valuable information from them. 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 enough amount of properly extracted characters to successfully train our Neural Networks. The data is split into two different sets: training and testing, for validation. We use the same training set to learn all the classifiers, and the same test set for evaluating their accuracy.

C. Feature extraction

We have implemented different types of feature extraction to test which of them gives the best results [6].

1) Restricted Boltzmann Machine

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 learnt by the RBM:

2) Histogram of Oriented Gradients

This technique uses the distribution of directions of gradients as features [7]. Gradients of an image are useful because the magnitude of gradient is large around edges and corners which give us a lot of information about objects shape. The problem with this method is that the characters image are of size 28x28. The (Figure 1) shows an example of feature extraction using HOG which is one of the characters in the dataset.

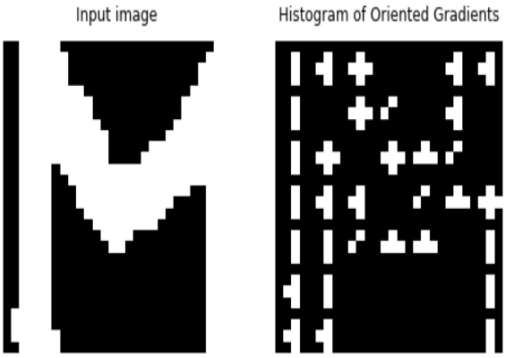

Figure 1: Original character and the result of HOG

3) Principal Component Analysis

PCA is a linear transformation algorithm that seeks to project the original features of our data onto a smaller set of features while retraining most of the information. This algorithm tries to find the most appropriate direction or angles that maximize the variance in the new subspace [8] [9]. The number of PCA features used in this paper is 100. First 2 features of the PCA are shown in the below (Figure 2).

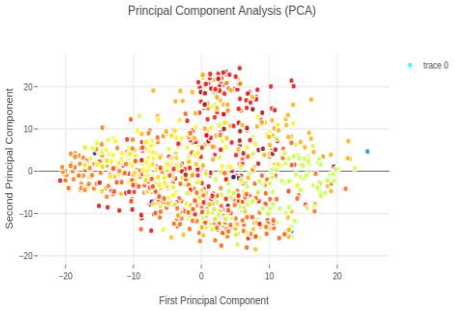

Figure 2: PCA of the first two features

4) No feature extraction

For the purpose of evaluating the previous methods we also trained our neural networks without any type of feature extraction using the binarized image of the characters (Figure 3) shown below.

Figure 3: A binarized character without feature extraction

The(Figure 4) represents the process that we follow to train our network.

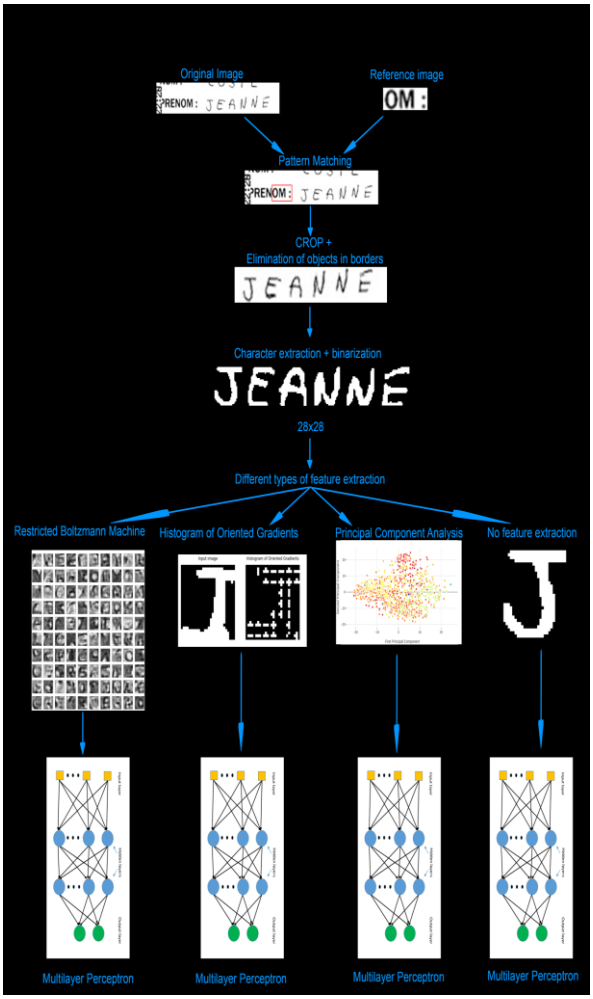

Figure 4: Process diagram

D. Classifiers

We use four different classifiers with the following parameters:

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

- RBM:
  - n components = 300
  - learning rate = 0.01
  - n iterations = 45
- MLP:
  - layers = (300, 400, 150)
  - activation function = 'relu'(Rectified Linear Unit)
  - max iterations = 5000
  - tol(Total Outside Liability) = 0.0001

2. MLP classifier with HOG (Histogram of Oriented Gradients) features:

- layers = (300, 400, 150)
- activation function = 'relu'(Rectified Linear Unit)
- max iterations = 5000
- tol(Total Outside Liability) = 0.0001

3. MLP classifier with PCA (Principal Component Analysis) features:

- PCA:
  - n components = 100
- MLP:
  - layers = (300, 400, 150)
  - activation function = 'relu'(Rectified Linear Unit)
  - max iterations = 5000
  - tol (Total Outside Liability)= 0.0001

4. MLP classifier with character images directly:

- layers = (300, 400, 150)
- activation function = 'relu'(Rectified Linear Unit)
- max iterations = 5000
- tol(Total Outside Liability) = 0.0001

The learning algorithm for the RBM is Stochastic Maximum Likelihood and learning algorithm for all the MLPs is default in Scikit-Learn: Adam optimizer.

1) Rationale behind the conception of NN

The multilayer perceptron is the most popular network for this task and that it commonly produces good results, which is why we chose this type of Neural Networks. There are other more advanced methods used for handwritten text recognition such as convolutional neural networks, but the objective of this work was to solve the problem using an NN approach excluding Deep Learning.

E. Inference

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 i.e., we need to extract each individual character on the image. Then, apply the feature extraction that we want to use, for example calculate the Histogram of Oriented Gradients or calculate the state of the RBM for this image. Then the multilayer perceptron predicts a letter for each individual character. The final step is to put the values predicted by the Neural Network together to form the name.

F. 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% as training and 20% as testing 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 for each classifier show in the (Figure 5, 6, 7, 8) below.

MLP classification using RBM features:

	precision	recall	f1-score	support
-	0.75	0.25	0.38	24
A	0.91	0.94	0.92	1686
B	0.78	0.65	0.71	98
C	0.91	0.87	0.89	305
D	0.79	0.83	0.81	183
E	0.96	0.93	0.94	1471
F	0.69	0.70	0.70	61
G	0.84	0.90	0.87	110
H	0.87	0.67	0.76	327
I	0.89	0.91	0.90	979
J	0.81	0.85	0.83	109
K	0.80	0.73	0.76	48
L	0.94	0.95	0.95	1000
M	0.80	0.87	0.84	571
N	0.91	0.92	0.91	1060
O	0.93	0.89	0.91	598
P	0.74	0.76	0.75	93
Q	0.62	0.57	0.59	14
R	0.90	0.88	0.89	537
S	0.94	0.94	0.94	402
T	0.88	0.96	0.92	518
U	0.89	0.89	0.89	374
V	0.89	0.74	0.81	105
W	0.84	0.71	0.77	38
X	0.95	0.86	0.90	102
Y	0.89	0.92	0.90	163
Z	0.79	0.70	0.74	37
avg / total	0.90	0.90	0.90	11013

Figure 5: MLP classification using RBM features

HOG + MLP classification:

	precision	recall	f1-score	support
-	1.00	0.25	0.40	24
A	0.91	0.89	0.90	1686
B	0.59	0.73	0.65	98
C	0.84	0.93	0.88	305
D	0.68	0.71	0.69	183
E	0.90	0.92	0.91	1471
F	0.71	0.44	0.55	61
G	0.72	0.66	0.69	110
H	0.81	0.74	0.77	327
I	0.81	0.92	0.87	979
J	0.73	0.73	0.73	109
K	0.70	0.58	0.64	48
L	0.93	0.94	0.93	1000
M	0.75	0.85	0.80	571
N	0.90	0.86	0.88	1060
O	0.89	0.85	0.87	598
P	0.73	0.71	0.72	93
Q	0.62	0.36	0.45	14
R	0.88	0.76	0.82	537
S	0.94	0.90	0.92	402
T	0.93	0.93	0.93	518
U	0.84	0.86	0.85	374
V	0.75	0.71	0.73	105
W	0.73	0.79	0.76	38
X	0.86	0.74	0.79	102
Y	0.88	0.83	0.86	163
Z	0.83	0.65	0.73	37
avg / total	0.87	0.87	0.87	11013

Figure 6: HOG+MLP classification

PCA + MLP classification:

	precision	recall	f1-score	support
-	0.78	0.29	0.42	24
A	0.94	0.93	0.94	1686
B	0.91	0.72	0.81	98
C	0.85	0.94	0.89	305
D	0.74	0.84	0.79	183
E	0.96	0.95	0.95	1471
F	0.71	0.61	0.65	61
G	0.87	0.82	0.84	110
H	0.83	0.78	0.80	327
I	0.92	0.92	0.92	979
J	0.78	0.88	0.83	109
K	0.88	0.77	0.82	48
L	0.97	0.94	0.95	1000
M	0.83	0.89	0.86	571
N	0.94	0.92	0.93	1060
O	0.93	0.91	0.92	598
P	0.76	0.75	0.76	93
Q	0.89	0.57	0.70	14
R	0.88	0.92	0.90	537
S	0.94	0.95	0.94	402
T	0.92	0.95	0.93	518
U	0.90	0.88	0.89	374
V	0.78	0.80	0.79	105
W	0.78	0.74	0.76	38
X	0.90	0.90	0.90	102
Y	0.92	0.87	0.89	163
Z	0.76	0.86	0.81	37
avg / total	0.91	0.91	0.91	11013

Figure 7: PCA+MLP classification

MLP only classification:

	precision	recall	f1-score	support
-	1.00	0.12	0.22	24
A	0.93	0.95	0.94	1686
B	0.97	0.72	0.83	98
C	0.84	0.96	0.90	305
D	0.82	0.81	0.82	183
E	0.96	0.93	0.94	1471
F	0.66	0.74	0.70	61
G	0.89	0.89	0.89	110
H	0.79	0.83	0.81	327
I	0.87	0.93	0.90	979
J	0.88	0.77	0.82	109
K	0.61	0.85	0.71	48
L	0.95	0.96	0.95	1000
M	0.89	0.84	0.87	571
N	0.95	0.92	0.93	1060
O	0.91	0.91	0.91	598
P	0.78	0.81	0.79	93
Q	0.79	0.79	0.79	14
R	0.94	0.89	0.91	537
S	0.94	0.95	0.94	402
T	0.95	0.96	0.96	518
U	0.91	0.88	0.89	374
V	0.82	0.76	0.79	105
W	0.63	0.76	0.69	38
X	0.94	0.89	0.91	102
Y	0.91	0.89	0.90	163
Z	0.86	0.65	0.74	37
avg / total	0.91	0.91	0.91	11013

Figure 8: MLP only classification

III. IMPLEMENTATION

All the project steps were implemented in Python. 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 pre-processing the dataset, matplotlib and plotly for plotting different data and scikit-learn for the classification tasks. We illustrate how the implementation works in the Python notebook.

1) Usage

Downloading all the database and training the Neural Networks takes a long time, so in the GitHub repository of the project [2] we added a database with 10,000 downloaded images and pre-trained the 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. Dataset load method: If the value is 'load' it will load the database provided. If the value is 'download' it will download the images.
2. Save database: If the value is true it will save to a file the generated dataset with the downloaded images.
3. Load classifiers: If the value is true it will load the pre-trained classifiers provided. If the value is false it will train the classifiers.
4. Save classifiers: If the value is true it will save to a file the classifiers once they have been trained.
5. Save results: If the value is true it will save the results of the classification test to a file. It's useful because when dealing with a lot of data the RAM fills up and could cause the screen to freeze.
6. Enable error output: If the value is true it will print some information useful for debugging and for improving the program such as the images where the character extraction has failed.

IV. RESULTS

We ran our testing for the first 30000 names in the dataset because testing with more data was not possible with our machines (we don't have enough RAM), the included prebuilt dataset and classifiers from our repository were from this testing.

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.

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**

Figure 9:Individual character classification results
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 results of the computation of the Scikit metrics for the RBM-MLP Pipeline, MLP with HOG features, and MLP with PCA features and MLP classifiers are respectively shown in (Figure 9, 10)

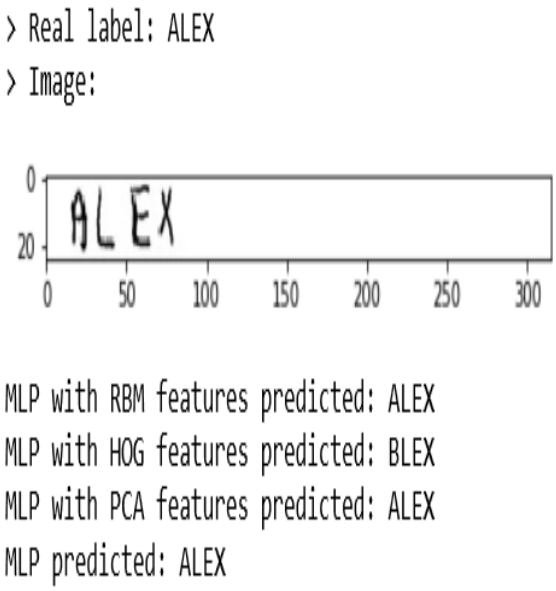

Figure 10: Recognized of name with all 4 features

We can also see the full name correct ratios and correlation ratios in the (Figure 11) below.

Full name test results:

Classifier	Correct percentage	Correlation ratio
MLP with RBM features	0.6017039403620873	0.9051507367279156
MLP with HOG features	0.5005324813631523	0.8717804066533881
MLP with PCA features	0.6437699680511182	0.9158415696972759
MLP only	0.64270500532481 37	0.915788133570558

Figure 11:Full name test results

As we can see, the percentage of correctly classified names is not too high, but if we calculate

The correlation of the predicted names with the real names the number is really high, that means that 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.

V. 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 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] Crowd Flower. Data for everyone library: Transcriptions of names from handwriting.

[2] Gary Bradski and Adrian Kaehler. Learning OpenCV: Computer vision with the OpenCVLibrary.

[3] OpenCV documentation: Template matching.

[4] Scipy Lectures. Labelling connected components using scikit-image.

[5] Scikit-Image. Scikit-image documentation: skimage.measure.label.

[6] 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.

[7] Satya Mallick. Learn opencv: Histogram of oriented gradients.

[8] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis.

[9] Anisotropic. Interactive intro to dimensionality reduction.