Performance Analysis of Character case-sensitive and case-insensitive Classification in Handwritten Character Recognition

Gaye Ediboğlu Bartos*, Éva Hajnal **, Yasar Hoşcan *
* Anadolu University/Department of Computer Engineering, Eskişehir, Turkey
**Óbuda University Alba Regia Technical Faculty, Székesfehérvár, Hungary
gayeediboglu@anadolu.edu.tr, hajnal.eva@amk.uni-obuda.hu, hoscan@anadolu.edu.tr

Abstract— **In character recognition classes are letters, numbers and punctuations. Therefore, the number of classes depends on the number of characters in the language. However many classes (such as upper case "C" and lower case "c") have very similar characteristics therefore merging such classes is also an option for a more successful recognition. In this study, we aim at evaluating the effects of abovementioned phenomenon for handwritten character recognition by performing the recognition in three different set of classes namely case-sensitive (52 classes), case-insensitive (26 classes) and similarity based (38 classes) using Deep Feedforward Networks. Looking at the results, as expected case-insensitive classification outperformed case-sensitive classification. Surprisingly, similarity based classification having a greater number of classes resulted in better accuracy rate compare to case- insensitive classification.**

I. INTRODUCTION

Handwriting recognition is heavily researched both by researchers and technology makers since recognition of handwritings has become a feature of mobile devices such as smart phones and tablets. However, the success is still limited to discretely written characters. When it comes to cursive handwritings or characters with special accents, the success of the recognition starts to decrease [1]. The challenge with such recognition mostly derives from individuals having various types of handwritings and certain characters having similar shapes such as "o" and "O". These bring about the complications and ambiguities for the recognition process. Additionally, having a large set of class label also has a negative effect on the success of the classification. In order to tackle such errors, it is possible to put the similar letters into the same class and perform classification with a smaller number of classes. Having more samples in each class and fewer class labels is believed to increase the accuracy rate in most cases. In this study, we aim at evaluating the performance of three different classification strategies using a cursive handwritten character database which are namely case-sensitive (52 classes), case-insensitive (26 classes) and similarity based (38 classes) classes two layer feed-forward networks

II. RELATED WORKS

In 2003, Koerich published a study in which he evaluated the performance of different classification strategies on NIST handwritten database [2], [3] using Multilayer Perception Classifier (MLP). The results indicated that 26 letter meta class classifier outperformed the other two strategies which were combination of two 26-class classifier and 52-class classifier. In 2012 Sulistiyo et. al. adopted another classification strategy for an alphanumerical dataset including letters a-z, A-Z and numbers 0-9[4]. However, they used a two level classification method using Nested MLP. For the first level classification, instead of having 62 class labels; they divided the entire set into 15 similar groups (classes) using a Fuzzy C-Means (FCM) clustering method. After completion of the group level classification, each group was classified for individual characters. The results indicated that, accuracy for classifying into correct groups was 88.5 % and into correct character classes was 64.6%. Additionally, recognition of the upper case characters yield the highest accuracy by 84.38% followed by digits with 78.92% and lower case characters with 76.43%. Another handwritten character dataset EMNIST was published in 2017 by Cohen et. al. as an extension of its previous versions NIST and MNIST [3], [5]–[7]. The EMNIST dataset contains 814255 samples of letters and digits. In addition to NIST and MNIST, EMNIST not only provides two class hierarchies namely By Class (every character into a different class with a different label) and By Merge (similar characters into the same class with the same label) but also provide four more options namely: balanced dataset which is easy to apply due to its balanced subset of all the By Merge classes; letters dataset generated to increase the number of errors occurring from case confusion by merging all the uppercase and lowercase classes to form a balanced 26-class classification task; digits dataset being a balanced subset of the digits dataset containing 28,000 samples of each digit and a copy of MNIST dataset. In their study they used a three layer Extreme Learning Machine (ELM) network and Online Pseudo-Inverse Update Method (OPIUM). For recognition purposes, increasing the size of hidden layers in the network gave higher accuracy naturally requiring higher memory. The results presented that the Digits dataset gave higher accuracy rate compare to other sets including letters. Amongst the ones including letters, Balanced dataset gave the highest accuracy followed by By Merge and By Class classes.

III. THE DATA SET

In this work, we adopted C-Cube (Cursive Character Challenge) cursive character database [8]. The C-Cube dataset includes 57293 characters including 26 upper and 26 lower case versions of each Latin letter. In order to collect the characters, several postal Plants in the United States were used. This way the images used in the data set two different type of resolution 212 DPI and 300DPI depending on the source plant. Additionally for the same reason, the dataset is not a balanced dataset which means the number of each character are not the same. Some letters can be found in a larger quantity than other characters depending on how frequently they were used in the original papers. In addition to characters, several features for each character are also provided by the dataset. However, applying a deep learning algorithm (Convolutional Neural Networks) in the study we will not be using the features provided. Finally, the dataset is represented into training and test set containing respectively 38160 and 19133 characters.

A. Modification of the Data Set

The dataset is available in ".chr" file format. In the file, five integer numbers corresponding to the five features namely width, height, distance baseline-upperline, position upper extreme and position lower extreme can be found for each character [8]. After the five above integer numbers, the bitmap representation of the characters can be found ("0" referring to white pixels and "1" referring to black). And finally after each bitmap representation, class label is written in the next row. A sample character with its features and class label can be seen in Fig. 1.

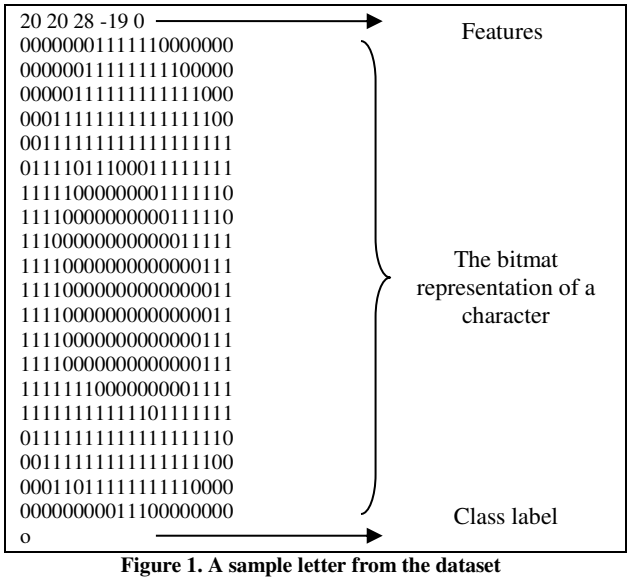

Figure 1. A sample letter from the dataset

The characters are written one after another without blank spaces between one character and the following one therefore it is not easy to use the dataset in the given format. The characters, binary numbers and integers are saved into the same file. Another challenge is that the characters are represented in different sizes, in other words they are not normalized into a standard size such as 28x28 pixels each. In order to be able to use the characters as an input to CNNs, we manipulated the characters given in the dataset. The steps followed are described below. Firstly, we copy the ".chr" file format into the MATLAB environment. It imports the data into a cell array and saves every row into one cell. Therefore we start the process with splitting the separate characters into separate cells. Since the file not only contains binary characters but also the class label and 5 features represented with numbers, the splitting process is more complex. Splitting the features is straightforward since they are separated with empty space. Secondly, we split the cells containing binary values representing each character in binary. For such cells, we split bit by bit, having one single bit in every cell. Lastly, we split the class label into a cell as a character. Having all the information in one cell array is not practical for classification therefore later we split the cell array into two separate matrices. The first matrix (1xN, N= number of characters) consists of the class label and the second one (Nx784) consists of the character representation for every character normalized into a 28x28 pixels and saved as 1x784 row for each character.

IV. EXPERIMENTS

Having all the letters saved into a matrix, we created three versions of the set with different class labels which are case-sensitive (52 classes), case-insensitive (26 classes) and similarity based (38 classes) classes as can be seen from Table 1. Merging upper and lower case of the characters based on their similarity was manually performed by looking at the shapes of the characters. However in the future, we plan to adopt a clustering method for the detection of similar classes. As for classification a two-later feed-forward network adopted. The experiments were carried out using MATLAB 9.3 Environment and Deep Learning Toolbox provided by MATLAB [9]. MATLAB is a programming platform which can be used in many fields such as control systems, image and video processing, medicine and finance[10]–[12].

Table 1 Similarity Based Merged Classes

	Merged Classes		**Merged Classes**
1	c- C	8	s-S
2	i-I	9	u-U
3	j-J	10	v-V
4	k-K	11	w-W
5	m-M	12	x-X
6	o-O	13	y-Y
7	p-P	14	z-Z

The unbalanced nature of C-Cube dataset was a disadvantage for the experiment since some letters have several samples and others only have a few.

V. RESULTS

The dataset was divided into sets namely train (70%), test (15%) and validation (15%) sets. As for the hidden layers 20 layers gave the highest accuracy for the dataset. The percentages for accuracies for 10 and 20 hidden layers are shown in the Table 2 below.

Table 2 Results

	No of Classes	Number of Hidden Layers	
		10	20
Case-sensitive	52	79,2 %	80,1%
Case-insensitive	26	82,5%	82,9%
Similarity Based	38	82,5%	83,1%

Looking at the results, it can be said that case insensitive classification gives better results than character sensitive classification for every single character. The main reason for such performance can be related with a smaller number of class labels and greater number of samples in each class. Having fewer number of classes from the case– sensitive and more number of classes from case–insensitive classification; similarity based classification outperforms other classifications as expected.
More detailed recognition results for case-sensitive and case-insensitive classification for 20 hidden neurons can be found in Table 3 below.

Table 3 Detailed Results for case-sensitive and case-insensitive classification

Case-sensitive				Case– insensitive	
Class	**Recognition Rate (%)**	**Class**	**Recognition Rate (%)**	**Class**	**Recognition Rate (%)**
a	79,1	A	79,5	A-a	80,1
b	78,8	B	80,9	B-b	80,2
c	69,3	C	72,0	C-c	75,0
d	78,1	D	80,2	D-d	81,5
e	85,0	E	84,5	E-e	85,5
f	71,5	F	72,9	F-f	80,0
g	77,5	G	80,0	G-g	82,0
h	77,3	H	77,5	H-h	80,0
i	71,5	I	70,0	I-I	72,5
j	75,0	J	74,3	J-j	77,5
k	77,5	K	75,0	K-k	80,0
l	70,1	L	76,6	L-l	78,4
m	84,5	M	85,5	M-m	87,5
n	77,8	N	80,1	N-n	82,5
o	71,5	O	74,6	O-o	75,2
p	75,0	P	80,0	P-p	87,5
q	75,0	Q	75,0	Q-q	80,0
r	75,0	R	77,5	R-r	80,0
s	82,3	S	85,5	S-s	87,5
t	80,3	T	81,3	T-t	83,0
u	78,1	U	72,3	U-u	81,3
v	70,0	V	77,8	V-v	81,5
w	83,2	W	85,0	W-w	87,5
x	85,0	X	86,1	X-x	88,0
y	75,5	Y	71,1	Y-y	81,1
z	82,5	Z	80,0	Z-z	85,3

VI. CONCLUSION

In this study, different classification strategies were applied on C-Cube handwritten character dataset in order to investigate the effects of grouping the classes. Putting upper case and lower case version of each character into one group provided higher accuracy rate compare to keeping case sensitive class labels. However, having a higher number of class labels, similarity based strategy outperformed the others by a small difference. We believe the results would have been more distinctive if the number of samples in each class was more balanced. The number of instances in each class in the dataset was highly unbalanced therefore; the results may differ in more balanced datasets.

VII. FUTURE WORK

A more sophisticated method such as clustering for finding metaclass labels rather than putting similar characters into same classes manually could improve the performance of the merging. The next step of the study is going to be application of such automated grouping for similar classes. Additionally, application of the same strategies on EMNIST dataset is planned to be performed.

REFERENCES

[1] G. E. Bartos, É. Hajnal, and Y. Hoşcan, "Comparison of Feature Extraction Techniques for Handwriting Recognition," in *IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI 2018)*, Temesvár: IEEE Hungary Section; IEEE Romania Section, 2018, pp. 405-410 PG–6.

[2] A. L. Koerich, "Unconstrained Handwritten Character Recognition Using Different Classification Strategies," 2003.

[3] P. J. Grohter, "NIST Special Database 19 Handprinted Forms and Characters Database," 1995.

[4] M. D. Sulistiyo, D. Saepudin, and Adiwijaya, "Optical character recognition using modified direction feature and nested multi layer perceptrons network," *Proceeding - 2012 IEEE Int. Conf. Comput. Intell. Cybern. Cybern. 2012*, no. October 2014, pp. 30–34, 2012.

[5] G. Cohen, S. Afshar, J. Tapson, and van A. Schalik, "EMNIST : an extension of MNIST to handwritten letters," 2017.

[6] P. J. Grother and K. K. Hanaoka, "NIST Special Database 19," pp. 1–30, 2016.

[7] M. Thoma, "The HASYv2 dataset," pp. 1–8, 2017.

[8] F. Camastra, M. Spinetti, and A. Vinciarelli, "Offline Cursive Character Challenge : a New Benchmark for Machine Learning and Pattern Recognition Algorithms .," *18th Int. Conf. Pattern Recognit.*, vol. 2, pp. 913–916, 2006.

[9] "MATLAB and Statistics Toolbox Release 2017b." The MathWorks, Inc., Natick, Massachusetts, United States.

[10] M. Kalechman, *Practical MATLAB Applications for Engineers*. Boca Raton: CRC Press, 2009.

[11] O. Demirkaya, M. Asyali, and P. Sahoo, *Image Processing with MATLAB*. Boca Raton: CRC Press, 2009.

[12] K. Széll and G. Manhertz, "Development of a web-based training module in robotics," in *2016 International Symposium on Small-scale Intelligent Manufacturing Systems (SIMS)*, 2016.