International Journal for Research in Engineering Application & Management (IJREAM)
ISSN : 2454-9150 Vol-08, Issue-02, MAY 2022

Handwritten Text Recognition using CRNN

Avishkar Sudharma Dalvi[1], Riddhi Jagdish Narkar[2], Soumyojyoti Jyotirmoy Dutta[3], Vedang Nilesh Gore[4], Deepali Kayande[5]

Department of Computer Engineering, A.P. Shah Institute of Technology, Thane, India[1,2,3,4,5]

20202002@apsit.edu.in[1], *19102003@apsit.edu.in[2], 19102014@apsit.edu.in[3], 19102065@apsit.edu.in [4], ddkayande@apsit.edu.in [5]*

Abstract Handwritten Text Recognition (HTR) is a very important field of research in the domain of Machine Learning, Image Processing, and Artificial Intelligence. This is because it often involves decoding human written letters and words in a given language. This project involves the recognition of handwritten words. This project uses ANN (artificial neural network) as it mimics the biological functioning of the brain, and hence is very efficient according to research for this problem statement. A Kaggle dataset with 4 lakh transcripted images of names written in capital letters was used. This works aims to perform a comparative study by using 4 different activation functions and by keeping the model, algorithm and dataset fixed.

Keywords — comparative study, CRNN, ELU, Handwritten text recognition, HTR, OCR, ReLU, SELU, Sigmoid.

I. INTRODUCTION

Handwritten text recognition (HTR) involves an artificially intelligent system to interpret human written words into a machine-readable format, for example, Unicode, or editable text. There are two versions of this technology, the first is real-time recognition, and the second is recognition based on a scanned image of handwritten text. The real-time recognition involves an input device, such as a touchscreen tablet or a digital drawing and writing tablet and the recognition runs live when the user writes something using a digital device like a digital pen or a stylus on the tablet. This focuses a lot more on the type of curves and shapes of the material being written on the device to interpret it. Some of the first industry examples of such technologies was the handwritten notes to Unicode characters conversion feature of Apple's Newton which was launched in 1992 [8]. Although it was discontinued after some time, today, a lot of devices and software have been launched which try to detect and convert handwritten text in real time with significantly more accuracy.

In 1993, Goldberg and Richardson from Xerox PARC published a paper about *Unistrokes* (Goldberg & Richardson, 1993) [9]. Goldberg and Richardson wanted to design a character set that could be entered in an eyes-free manner. Hence, they came up with a very simplified version of all 26 English alphabets and called it Unistrokes. Unistrokes was built to optimize the recognition accuracy and text entry speed. Here, each character is drawn using a single stroke. But to put it to practice, one needs to learn this alphabet set. It is illustrated in Figure 1. Hence, this was not possible to be used in systems where raw data, in the form of normal English writing, could be processed. This would require users to change their normal style of writing if it were to be used in handwritten text recognition software.

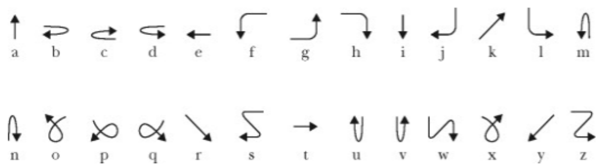

Figure 1 - Unistrokes character set by Goldberg and Richardson

Handwritten text recognition and OCR, or Optical Character Recognition are very similar. One of them is a special case of the other, and the other can be used as an umbrella term. So, OCR is an umbrella term which involves recognizing any component of human language in the written form. Here human language suggests any language which humans use to communicate, like for example English. Handwritten text recognition involves Optical Character Recognition, or is a 'special' case under OCR where we only deal with handwritten language components.

Handwritten text recognition (HTR) is of two types, offline and online [3]. It could also be classified into many different levels. The most basic level is where this project deals with the recognition of individual characters of a language, in which sense, we call handwritten character recognition or HCR. For example, recognizing A, B, g, m, etc. (lowercase and uppercase both included). The next level consists of individual words. For example, tree, computer, Rahul, etc.

Further down the line comes whole sentences. For example, *Let's play chess, I like to drink milk at night,* etc. This level is the most complex and also the most rewarding, in the sense that a software which is able to recognize at this level can have myriad use cases in our day to day lives.

 DOI : 10.35291/2454-9150.2022.0161

ISSN : 2454-9150 Vol-08, Issue-02, MAY 2022

This research project involves work on handwritten text recognition of the English language up to the level of recognizing words. The dataset used was sourced from Kaggle and contains transcriptions of 400,000 handwritten names. It is a 1GB data set consisting of images. It has data distributed into train, test and valid sets and contains 3 csv files for labelled data.

II. PROBLEM STATEMENT

Writing has been an ancient form of art as well as the main medium for written communication throughout history. It is still prevalent in society and is considered indigenous to a lot of practices modern humans do. With the advent of digital media, however, it seems challenging to keep up with writing instead of switching to typing. Typing, although considered more convenient, there is still a lot of data which is present in hard copies and written form. Not only this, but a lot of sectors involve a mixture of digital and handwritten media to do their job. For example, students and working professionals often need to switch between the digital medium and the traditional writing medium.

A lot of people often face slight inconveniences when they need a text format for a piece of writing. The only possible way out here is to manually type out the information on a digital medium to use it in the digital form. The work presented tries to research the methods which does this efficiently.

For this project, the aim is to research which activation function when used with CRNN for this project gives the best accuracy. The project uses ‘Sigmoid’, ‘SELU’, ‘ELU’, and ‘ReLU’ activation functions. For training, a dataset from Kaggle is used, which includes data in the form of words in capital letters and names and surnames of French students. Due to this, the work could be working in the 2nd level of handwritten text recognition, that is, recognizing words.

III. OBJECTIVE

The objective of this research project is to keep the ML model and algorithm and dataset the same and change activation functions to get a comparison between these different activation functions. The activation functions considered for this research project are ‘Sigmoid’, ‘SELU’, ‘ELU’, and ‘ReLU’. There were several more to this list, but the accuracies they showed had insignificant differences between them and any one of the considered activation functions, so they were not considered for this study. The implementation section contains a brief introduction to all these 4 activation functions.

IV. ALGORITHM

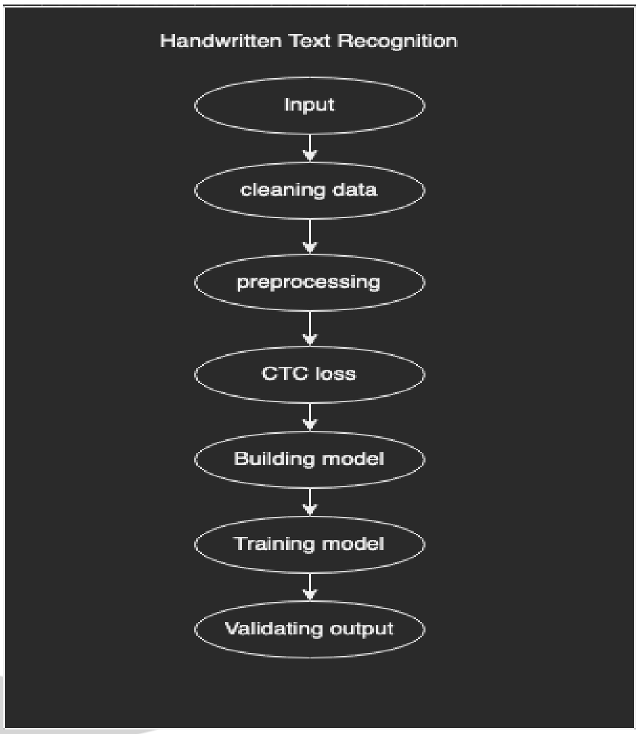

Figure 2 – Flow of modules

A. Input

The input is taken in the form of images from the user.

B. Cleaning Data

A lot of these images may be null, or unreadable. In this stage, such images are removed.

C. Preprocessing

It is needed to make all the data uniform so that the model can train on it. So in order to achieve consistent dimensions, the image has been cropped if it is larger, or padded with white pixels if it is smaller. [4], [6]

D. CTC loss

This project also involves determining the CTC loss, or Connectionist Temporal Classification Loss here. Many times, when scanning the input, a letter can occur in 2 different alignments, in which case, it may get considered twice, whereas it actually was just one. CTC loss takes care of such scenarios.

E. Building the model

Here is where the layers are added and the is built. This project is developed by building a CRNN model. (Convolution Recurrent Neural Network.)

F. Training the model

The model is trained on the training set. In the dataset this project uses, the data was already divided into three groups namely, training, testing and validation.

G. Predictions

Based on the model, the results are predicted.

Expected input:

DOI : 10.35291/2454-9150.2022.0161

ISSN : 2454-9150 Vol-08, Issue-02, MAY 2022

Figure 3 – Sample data

Expected output: SIMON

V. IMPLEMENTATION

The model we used in this project is CRNN, which is the Convolutional Recurrent Neural Network model, as several researchers recommend this model for handwritten text recognition. There has been a myriad of research papers comparing different ANNs, or Artificial Neural Networks to see which type of model gives the best accuracy under which specific conditions. CRNN was a model which proved to be the best for HTR. [2], [4], [5], [7]

```
input_data = Input(shape=(256, 64, 1), name='input')

inner = Conv2D(32, (3, 3), padding='same', name='conv1', kernel_initializer='he_normal')(input_data)
inner = BatchNormalization()(inner)
inner = Activation('relu')(inner)
inner = MaxPooling2D(pool_size=(2, 2), name='max1')(inner)

inner = Conv2D(64, (3, 3), padding='same', name='conv2', kernel_initializer='he_normal')(inner)
inner = BatchNormalization()(inner)
inner = Activation('relu')(inner)
inner = MaxPooling2D(pool_size=(2, 2), name='max2')(inner)
inner = Dropout(0.3)(inner)

inner = Conv2D(128, (3, 3), padding='same', name='conv3', kernel_initializer='he_normal')(inner)
inner = BatchNormalization()(inner)
inner = Activation('relu')(inner)
inner = MaxPooling2D(pool_size=(1, 2), name='max3')(inner)
inner = Dropout(0.3)(inner)
```

Figure 4 – Layers of the model

Figure 4, has the demonstration of the 3 layers, with the activation function ReLU. [1].

Here's a brief introduction to all the 4 activation functions considered in this study:

1) Sigmoid

It is a non-linear, continuous, differentiable and monotonic function which outputs between the range of 0 and 1. It accepts real numbers as inputs. It's a safe function to use, as its output is defined in a bounded interval.

$$f(x) = \frac{1}{1 + e^{-x}}$$

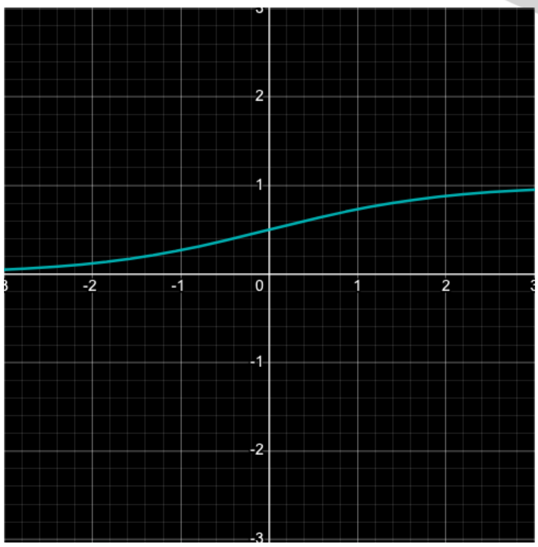

Figure 5 – Sigmoid Graph

2) SELU

It stands for scaled exponential linear units. They can self-normalize themselves. The constants used here, lambda and alpha have positive values defined for them. Here we are just assuming the values upto 4th place after the decimal.

$$f(x) = \begin{cases} \lambda x, & x > 0 \\ \lambda(\alpha e^{x} - \alpha), & x \leq 0 \end{cases}$$
$$where\ \alpha \approx 1.6733, and\ \lambda \approx 1.0507$$

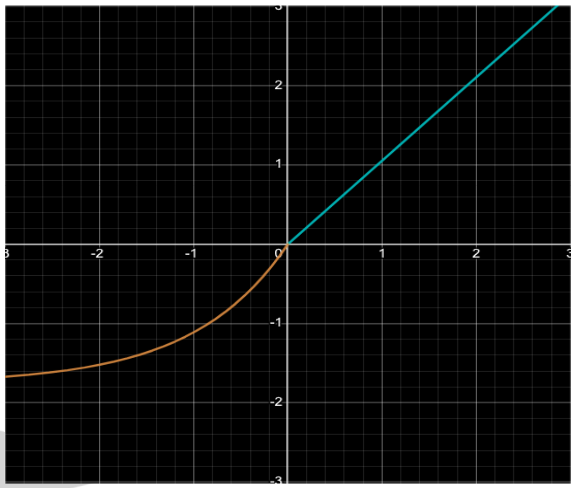

Figure 6 – SELU Graph

3) ELU

ELU stands for Exponential Linear Unit. It is similar to ReLU, except for when the input is negative. For this, the value of the constant alpha, generally a value between 0.1 or 0.3 is considered.

$$f(x) = \begin{cases} x, & x > 0 \\ \alpha(e^{x} - 1), & x < 0 \end{cases}$$

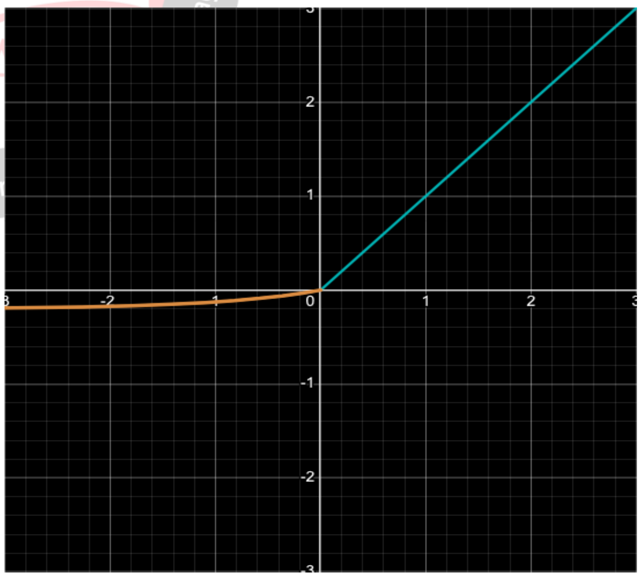

Figure 7 – ELU Graph with alpha = 0.2

4) ReLU

ReLU stands for Rectified Linear Units. It is non-linear, are gives a similar, but a better performance than Sigmoid.

$$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

DOI : 10.35291/2454-9150.2022.0161

International Journal for Research in Engineering Application & Management (IJREAM)
ISSN : 2454-9150 Vol-08, Issue-02, MAY 2022

Figure 8 – ReLU Graph

```
# CNN to RNN
inner = Reshape(target_shape=((64, 1024)), name='reshape')(inner)
inner = Dense(64, activation='relu', kernel_initializer='he_normal', name='dense1')(inner)

## RNN
inner = Bidirectional(LSTM(256, return_sequences=True), name = 'lstm1')(inner)
inner = Bidirectional(LSTM(256, return_sequences=True), name = 'lstm2')(inner)

## OUTPUT
inner = Dense(num_of_characters, kernel_initializer='he_normal',name='dense2')(inner)
y_pred = Activation('softmax', name='softmax')(inner)

model = Model(inputs=input_data, outputs=y_pred)
model.summary()
```

Figure 9 – CRNN model code

Layer (type)	Output Shape	Param #
input (InputLayer)	[(None, 256, 64, 1)]	0
conv1 (Conv2D)	(None, 256, 64, 32)	320
batch_normalization (BatchNormalization)	(None, 256, 64, 32)	128
activation (Activation)	(None, 256, 64, 32)	0
max1 (MaxPooling2D)	(None, 128, 32, 32)	0
conv2 (Conv2D)	(None, 128, 32, 64)	18496
batch_normalization_1 (BatchNormalization)	(None, 128, 32, 64)	256
activation_1 (Activation)	(None, 128, 32, 64)	0
max2 (MaxPooling2D)	(None, 64, 16, 64)	0
dropout (Dropout)	(None, 64, 16, 64)	0
conv3 (Conv2D)	(None, 64, 16, 128)	73856
batch_normalization_2 (BatchNormalization)	(None, 64, 16, 128)	512
activation_2 (Activation)	(None, 64, 16, 128)	0
max3 (MaxPooling2D)	(None, 64, 8, 128)	0
dropout_1 (Dropout)	(None, 64, 8, 128)	0
reshape (Reshape)	(None, 64, 1024)	0
dense1 (Dense)	(None, 64, 64)	65600
lstm1 (Bidirectional)	(None, 64, 512)	657408
lstm2 (Bidirectional)	(None, 64, 512)	1574912
dense2 (Dense)	(None, 64, 30)	15390
softmax (Activation)	(None, 64, 30)	0

Total params: 2,406,878
Trainable params: 2,406,430
Non-trainable params: 448

Figure 10 – Details of the built model

Here are the details about the layers built into the model illustrated in Figure 9 and Figure 10.

For the comparative study purpose, just the activation function has been switched with the previously mentioned activation functions - Sigmoid, ELU and SELU as demonstrated in Figure 11, Figure 12, and Figure 13 respectively.

As demonstrated in Figure 4 and Figure 5, a single activation function has been used for all layers and a combination of different functions in different layers is not used.

```
input_data = Input(shape=(256, 64, 1), name='input')

inner = Conv2D(32, (3, 3), padding='same', name='conv1', kernel_initializer='he_normal')(input_data)
inner = BatchNormalization()(inner)
inner = Activation('sigmoid')(inner)
inner = MaxPooling2D(pool_size=(2, 2), name='max1')(inner)

inner = Conv2D(64, (3, 3), padding='same', name='conv2', kernel_initializer='he_normal')(inner)
inner = BatchNormalization()(inner)
inner = Activation('sigmoid')(inner)
inner = MaxPooling2D(pool_size=(2, 2), name='max2')(inner)
inner = Dropout(0.3)(inner)

inner = Conv2D(128, (3, 3), padding='same', name='conv3', kernel_initializer='he_normal')(inner)
inner = BatchNormalization()(inner)
inner = Activation('sigmoid')(inner)
inner = MaxPooling2D(pool_size=(1, 2), name='max3')(inner)
inner = Dropout(0.3)(inner)

# CNN to RNN
inner = Reshape(target_shape=((64, 1024)), name='reshape')(inner)
inner = Dense(64, activation='sigmoid', kernel_initializer='he_normal', name='dense1')(inner)

## RNN
inner = Bidirectional(LSTM(256, return_sequences=True), name = 'lstm1')(inner)
inner = Bidirectional(LSTM(256, return_sequences=True), name = 'lstm2')(inner)

## OUTPUT
inner = Dense(num_of_characters, kernel_initializer='he_normal',name='dense2')(inner)
y_pred = Activation('softmax', name='softmax')(inner)

model = Model(inputs=input_data, outputs=y_pred)
model.summary()
```

Figure 11 – Model code with activation function ‘Sigmoid’

```
input_data = Input(shape=(256, 64, 1), name='input')

inner = Conv2D(32, (3, 3), padding='same', name='conv1', kernel_initializer='he_normal')(input_data)
inner = BatchNormalization()(inner)
inner = Activation('elu')(inner)
inner = MaxPooling2D(pool_size=(2, 2), name='max1')(inner)

inner = Conv2D(64, (3, 3), padding='same', name='conv2', kernel_initializer='he_normal')(inner)
inner = BatchNormalization()(inner)
inner = Activation('elu')(inner)
inner = MaxPooling2D(pool_size=(2, 2), name='max2')(inner)
inner = Dropout(0.3)(inner)

inner = Conv2D(128, (3, 3), padding='same', name='conv3', kernel_initializer='he_normal')(inner)
inner = BatchNormalization()(inner)
inner = Activation('elu')(inner)
inner = MaxPooling2D(pool_size=(1, 2), name='max3')(inner)
inner = Dropout(0.3)(inner)

# CNN to RNN
inner = Reshape(target_shape=((64, 1024)), name='reshape')(inner)
inner = Dense(64, activation='elu', kernel_initializer='he_normal', name='dense1')(inner)

## RNN
inner = Bidirectional(LSTM(256, return_sequences=True), name = 'lstm1')(inner)
inner = Bidirectional(LSTM(256, return_sequences=True), name = 'lstm2')(inner)

## OUTPUT
inner = Dense(num_of_characters, kernel_initializer='he_normal',name='dense2')(inner)
y_pred = Activation('softmax', name='softmax')(inner)

model = Model(inputs=input_data, outputs=y_pred)
model.summary()
```

Figure 12 – Model code with activation function ‘ELU’

```
input_data = Input(shape=(256, 64, 1), name='input')

inner = Conv2D(32, (3, 3), padding='same', name='conv1', kernel_initializer='he_normal')(input_data)
inner = BatchNormalization()(inner)
inner = Activation('selu')(inner)
inner = MaxPooling2D(pool_size=(2, 2), name='max1')(inner)

inner = Conv2D(64, (3, 3), padding='same', name='conv2', kernel_initializer='he_normal')(inner)
inner = BatchNormalization()(inner)
inner = Activation('selu')(inner)
inner = MaxPooling2D(pool_size=(2, 2), name='max2')(inner)
inner = Dropout(0.3)(inner)

inner = Conv2D(128, (3, 3), padding='same', name='conv3', kernel_initializer='he_normal')(inner)
inner = BatchNormalization()(inner)
inner = Activation('selu')(inner)
inner = MaxPooling2D(pool_size=(1, 2), name='max3')(inner)
inner = Dropout(0.3)(inner)

# CNN to RNN
inner = Reshape(target_shape=((64, 1024)), name='reshape')(inner)
inner = Dense(64, activation='selu', kernel_initializer='he_normal', name='dense1')(inner)

## RNN
inner = Bidirectional(LSTM(256, return_sequences=True), name = 'lstm1')(inner)
inner = Bidirectional(LSTM(256, return_sequences=True), name = 'lstm2')(inner)

## OUTPUT
inner = Dense(num_of_characters, kernel_initializer='he_normal',name='dense2')(inner)
y_pred = Activation('softmax', name='softmax')(inner)

model = Model(inputs=input_data, outputs=y_pred)
model.summary()
```

Figure 13 – Model code with activation function ‘SELU’

After this, the model was trained and run to achieve different accuracies for all 4 activation functions. Following this, it was able to determine which activation

 DOI : 10.35291/2454-9150.2022.0161

is best for HTR.

VI. CONCLUSION

Thus, this work includes the implementation of different activation functions on the same dataset and same model. The constant parameters were the dataset and the model, and the only changing parameter was the activation function. The 4 activation functions were Sigmoid, SELU, ELU and ReLU. Sigmoid, SELU, ELU, and ReLU all 4 of these activation functions gave us different accuracies which are summarized with the help of a table below.

Sr. No.	Activation Function	Accuracy
1)	Sigmoid	5.90%
2)	ELU	58.43%
3)	SELU	71.20%
4)	ReLU	74.97%

Table 1- Comparative study of different activation functions and their accuracies

As demonstrated in this table, ReLU has the highest accuracy, i.e. 74.97%, thus surpassing all others. The lowest is that of Sigmoid at 5.90%. ELU stands at 58.43% and SELU at 71.20%.

On the basis of this comparative study, it can be thus concluded that for a fixed dataset consisting of 4,00,000 transcripted images of uppercase handwriting, and using the CRNN model with CTC loss, it is able to achieve the highest accuracy of 74.97% with the activation function of ReLU, opposed to three other activation functions - SELU, Sigmoid and ELU.

Thus, ReLU can be a preferred activation function choice when trying to solve a similar problem under the domain of handwritten text recognition.

VII. ACKNOWLEDGMENT

We have great pleasure in presenting our research paper on Handwritten Text Recognition. We take this opportunity to express our sincere thanks to our guide, Prof. Deepali Kayande, Department of Computer Engineering, A.P. Shah Institute of Technology, Thane for providing the technical guidelines and suggestions regarding the line of work. We would like to express our gratitude for his constant encouragement, support and guidance through the development of the project.

We thank Prof. Sachin Malave, Head of Department, Department of Computer Engineering, A.P. Shah Institute of Technology, Thane for his encouragement during the progress meeting and for providing guidelines to write this report.

We also thank the entire staff of A.P. Shah Institute of Technology, Thane for their invaluable help rendered during the course of this work. We wish to express our deep gratitude to all our colleagues at A.P. Shah Institute of Technology, Thane for their encouragement.

VIII. REFERENCES

[1] Fathma Siddique, Shadman Sakib, Md. Abu Bakr Siddique, "Recognition of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers", International University of Business Agriculture and Technology, Dhaka 1230, Bangladesh.

[2] Harald Scheidl, 2018, "Handwritten Text Recognition in Historical Documents," Technische Universität Wien.

[3] Jamshed Memon[1], Maira Sami[2], Rizwan Ahmed Khan[3] and Mueen Uddin[4], 2020, "Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR)", [1]School of Computing, Quest International University Perak, Ipoh 30250, Malaysia, [2]Department of Computer Science, Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology, Karachi 75600, Pakistan, [3]Faculty of IT, Barrett Hodgson University, Karachi 74900, Pakistan, [4]Department of Software Engineering, Faculty of Science and Technology, Ilma University, Karachi 75190, Pakistan.

[4] Shivangkumar R. Patel, Ms. Jasmine Jha, "Handwritten Character Recognition using Machine Learning Approach - A Survey", from L.J.I.E.T. PG Department Ahmedabad, Gujarat – India, presented in International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO) – 2015.

[5] Rohan Vaidya, Darshan Trivedi, Sagar Satra, "Handwritten Character Recognition Using Deep-Learning" Dwarkadas J Sanghvi College of Engineering, Mumbai, India, Proceedings of the 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT 2018) IEEE Xplore Compliant - Part Number: CFP18BAC-ART; ISBN:978-1-5386-1974-2.

[6] Monica Patel, Shital P. Thakkar, " Handwritten Character Recognition in English: A Survey", Department of Electronics and Communication, Dharmsinh Desai University, Nadiad, Gujarat, India, International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 2, February 2015.

[7] Ahmed Mahi Obaid[1], Hazem M. El Bakry[2], M.A. Eldosuky[3], A.I. Shehab[4], "Handwritten Text Recognition System Based on Neural Network", Dept. Information Systems[1, 2, 4], Dept. Computer Science[3] Faculty of Computer Science and Information Systems, Mansoura University[1, 2, 3, 4], Mansoura, Egypt, International Journal of Advanced Research in Computer Science & Technology (IJARCST 2016) 72 Vol. 4, Issue 1 (Jan. - Mar. 2016)

[8] David Goldberg, Cate Richardson, Xerox Cooperation, "Touch-Typing With a Stylus", CHI '93: Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems, May 1993.

 DOI : 10.35291/2454-9150.2022.0161