Handwritten Text Recognition

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

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

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

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 specific language. This project involves recognition of individual words. We are using 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. We used a Kaggle dataset with 4 lakh transcripted images of names written in capital letters. This is a comparative study by using 4 different activation functions and by keeping the model, algorithm and dataset fixed.

Index Terms – HTR: Handwritten Text Recognition.

Keywords* — *comparative study, CRNN, elu, Handwritten text recognition, OCR, relu, selu, signmoid,

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 an 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. Although it was discontinued after some time, but 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). 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.

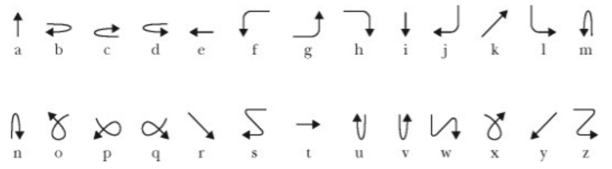


Figure 1 - Unistrokes character set by Goldberg and Richardson

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) could be again further classified into many different levels. The most basic level is where we deal with recognition of individual characters of a language, in which sense, we call it as 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.

In this research project, we worked on handwritten text recognition of English language up to the level of recognizing words. The dataset we 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 the society and is considered indigenous to a lot of practices we 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 as more convenient, we still have 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 these 2 media.

A lot of people often face slight inconveniences, when they need a text format of 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. Our mini project tries to research on the methods which do this efficiently.

For this project, we aim to research about which activation function when used with CRNN for this project gives the best accuracy. We used 'sigmoid', 'selu', 'elu', and 'relu' activation functions. We used a dataset from Kaggle, which includes data in the form of words in capital letters and are names and surnames of French students. Due to this, we would 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 same and change activation functions to get a comparison between these different activation functions. The functions we 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 added here.

IV. Algorithm

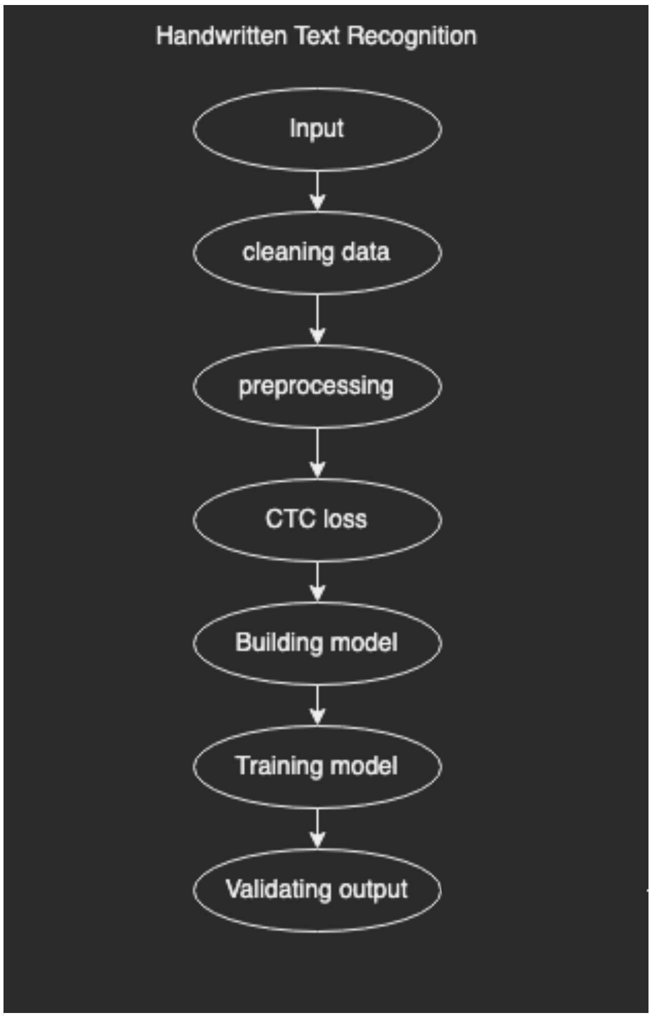


Figure 2 – Flow of modules

A. *Input*

We take input in the form of images from the user.

B. *Cleaning Data*

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

C. *Preprocessing*

We need to make all the data uniform so that the model can train on it. Here we set the dimensions and crop the image if it is larger, or pad it with white pixels if it is smaller.

D. *CTC loss*

We even take care about 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 we add the layers and build the model. We built a CRNN model. (Convolution Recurrent Neural Network.

F. *Training the model*

Here we train model on the training set. In the dataset we chose, the data was already divided into three groups namely, training, testing and validation.

G. *Predictions*

Based on the model, we predict the result.

Expected input:

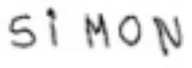
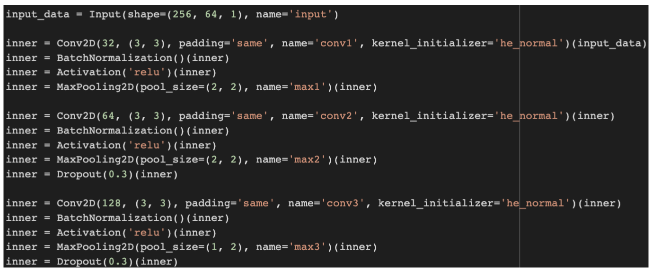


Figure 3 – Sample data

Expected output: SIMON

V. IMPLEMENTATION

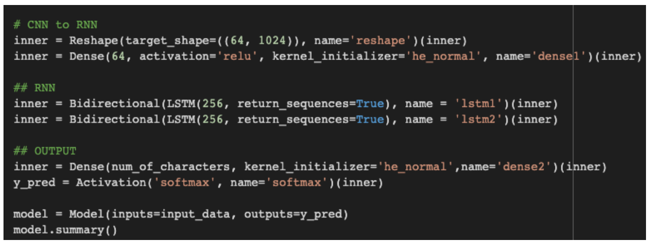
We built a CRNN, that is Convolutional Recurrent Neural Network model, as several researches recommend this model for handwritten text recognition. There have been myriad of research papers comparing different ANNs, or Artificial Neural Networks to see which type of model gives the best accuracy under which specific conditions. CRNN was a model which proved to be the best for HTR.



```
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)
```

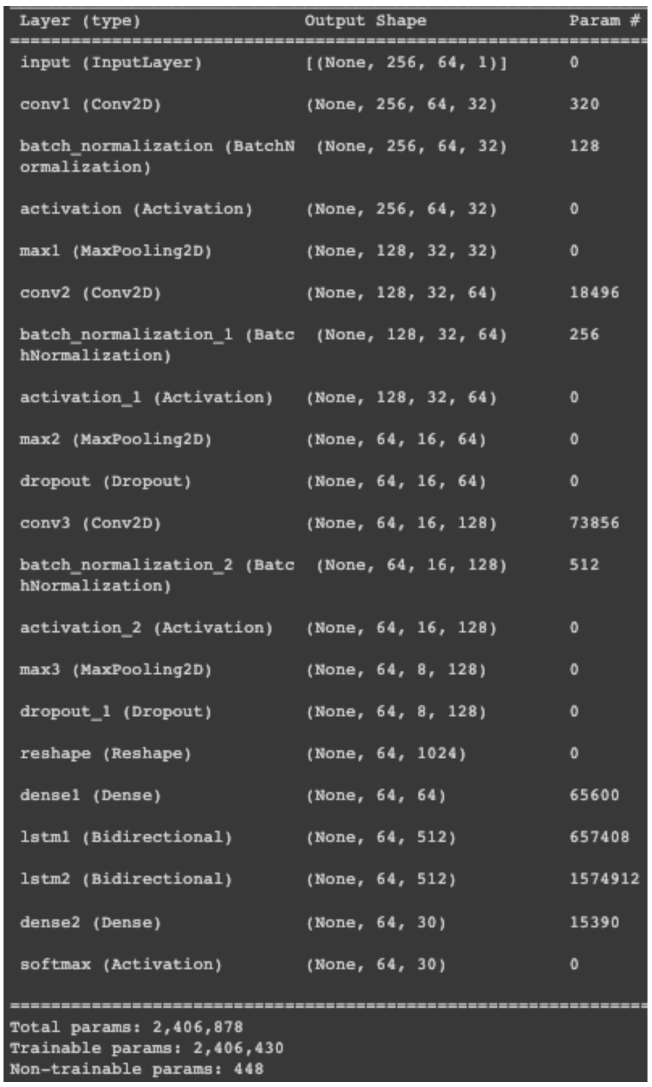
Figure 4 – Layers of the model

Here we have 3 layers, with the activation function relu.



```
# 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 5 – CRNN model code

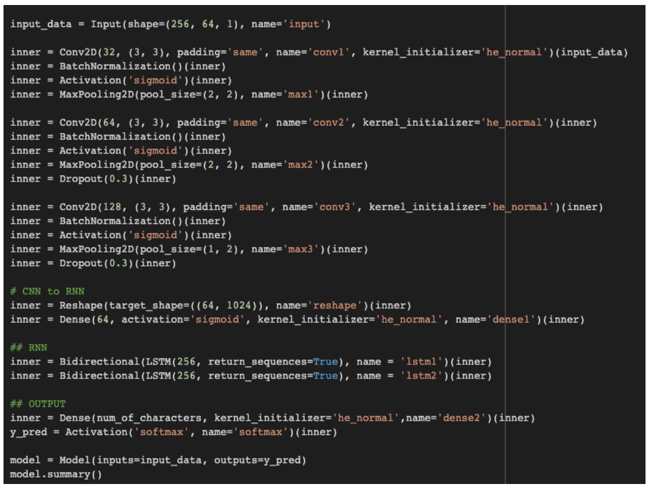


```
 Layer (type)                 Output Shape              Param #
=================================================================
 input (InputLayer)           [(None, 256, 64, 1)]      0

 conv1 (Conv2D)               (None, 256, 64, 32)       320

 batch_normalization (BatchN  (None, 256, 64, 32)       128
 ormalization)

 activation (Activation)      (None, 256, 64, 32)       0

 max1 (MaxPooling2D)          (None, 128, 32, 32)       0

 conv2 (Conv2D)               (None, 128, 32, 64)       18496

 batch_normalization_1 (Batc  (None, 128, 32, 64)       256
 hNormalization)

 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 (Batc  (None, 64, 16, 128)       512
 hNormalization)

 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 6 – Details of the built model

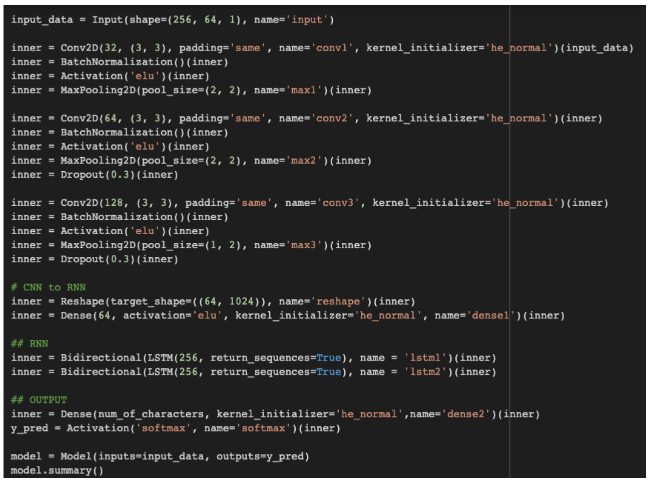
Here are the details about the layers we built into our model.

For the comparative study purpose, we just switched the relu function we mentioned previously with sigmoid, elu and selu like demonstrated in Figure 7, Figure 8, and Figure 9 respectively.



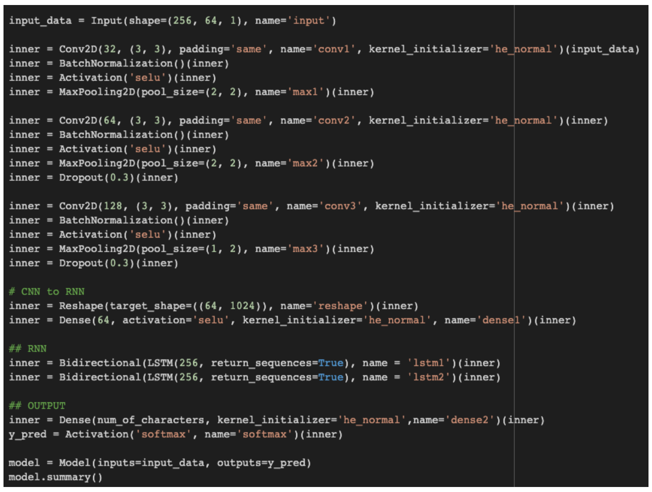
```
input_data = Input(shape=(256, 64, 1), name='input')

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 7 – 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 8 – 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 9 – Model code with activation function 'selu'

After this, we trained and run the model to achieve different accuracies for all 4 activation functions. Following this, we were able to determine which activation is best for HTR.

VI. CONCLUSION

We thus, implemented for different activation functions on the same dataset and same model. So 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, we thus conclude that for a fixed dataset consisting of 4,00,000 transcripted images of uppercase handwriting, and using the CRNN model with CTC loss, we were 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 towards our guide, Prof. Deepali Kayande, Department of Computer Engineering, A.P. Shah Institute of Technology, Thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of project.

We thank Prof. Sachin Malave, Head of Department, Department of Computer Engineering, A.P. Shah Institute of Technology, Thane for his encouragement during progress meeting and 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 towards all our colleagues of 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)