International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 07 | July 2021 www.irjet.net p-ISSN: 2395-0072

Handwritten Character Recognition using Deep Learning in Android Phones

Athira M Nair[1], Chrissie Aldo[1], Blessil Bose[1], Alex Joseph[1], Praseetha V.M[2], Sr. Elizabeth M J[3]

[1]UG Student, Dept. of CSE, SJCET, Palai, Kerala, India
[2]Associate Professor, Dept. of CSE, SJCET, Palai, Kerala, India
[3]Assistant Professor, Dept. of CSE, SJCET, Palai, Kerala, India

---

Abstract - *There are many things that humans have in common, yet there are other things that are very unique to every individual and one of them is handwriting. Handwriting is a skill that is personal to individuals. It has continued as a means of communication and recording information in day-to-day life. Because each person's handwriting is unique, it is sometimes hard to interpret the information they try to convey. As computerization is becoming more prominent these days, Handwriting Recognition is gaining importance in various fields. The major focus is to understand the handwriting and convert it into readable text. Deep learning, an ability of Artificial Intelligence (AI), is used for the system to learn the input automatically and convert the handwritten text to printed text.*

***Key Words*: Handwriting Recognition, Deep learning**

1. INTRODUCTION

Handwriting has continued as a means of communication in our day-to-day life. As each person's handwriting is unique, it is sometimes hard to interpret the information they try to convey. Handwriting Recognition is an ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-screens and other devices. Though it is a difficult problem due to the great variations of writing styles, different size and orientation angle of the characters, it is still found useful for the applications in some way. The main objective is to understand the handwriting and convert it into readable text, which includes characters, words, lines, paragraphs etc. In this project, the challenge is classifying the image of any handwritten word, which might be of the form of cursive or block writing. Along with this, Text-to-Speech is used to help people who have trouble reading on-screen text.

2. RELATED WORKS

Hao Zeng et al. proposes a method that focuses on using a simpler neural network instead of complicated ones that require high quality computer configuration to recognize handwritten digits with relatively promising accuracy[1]. MNIST dataset is used to train the neural network. An Efficient Algorithm for Real-Time Handwritten Character Recognition in Mobile Devices[2] that is used to measure the algorithm efficiency, tests were applied to 8 persons, who are related to the computing scene and outside it, with different writing styles. Mobile Client-Server Approach for handwritten digit recognition [3] is another one, in which CNN was also used to improve the performance of neural networks. The digit recognition consists of some modules for the processing.

Rohan Vaidya et al. designed an image segmentation based handwritten character recognition system using Deep-Learning [4]. OpenCV was used for performing Image processing and Tensor flow was used to train a neural network. Haishi Du et al. proposed a system that identifies words using acoustic signals generated by pens and paper using Deep-Learning [5]. The framework is created with three major components: segmentation, classification, and word suggestion. Handwritten Document into Digitized Text Using Segmentation Algorithm [6]. The main aim is to help in preserving history by making information searchable, easily, and reportable without the need for human labor.

Edgard Chammas et al. proposed a mobile application that is built on a distributed architecture that allows tourists to obtain additional information about location and menu entries in the Arabic language [7]. The recognition of printed texts is done using optical character recognition. SolveIt- an Application for Automated Recognition and Processing of Handwritten Mathematical Equations[8]. Here a convolutional neural network (CNN) is used to classify symbols. The recognized symbols are strung together to form an equation that can be parsed by the math engine (SymPy2). A framework that takes the image of multiple printed-papers using a mobile device's camera is used in Optical Character Recognition (OCR) Performance in Server-based Mobile Environment. After the first image is captured, the image is then directly sent to a server. Server processes the image using the OCR application directly and sends the text file back to the mobile device [9]. Handwritten Character Recognition to obtain Editable Text [10] is a system proposed by Ms. Jyoti A. Katkar. No internet connectivity is required for character recognition in the system. And the system offers 90% accuracy.

3. PROPOSED METHOD

This is a mobile application for handwriting recognition. When the user opens the application, the sign in / signup option appears. Registered users along with the guests can access the application. The user can register using their email-id and can set a password for their account. The user can upload the image either by capturing the image or input the image from the system's storage. The uploaded image is then processed in a neural network model (NN model) which identifies the characters, i.e.; the digits, alphabets or special symbols. After identifying these characters, they are converted into text (printed text) and this processed document is sent back to the user as output. The application has another feature which is text-to-speech conversion that converts the printed text into speech. This feature is mainly for the users who have difficulty in reading text from mobile phones. An overview of the system is given in figure 1.

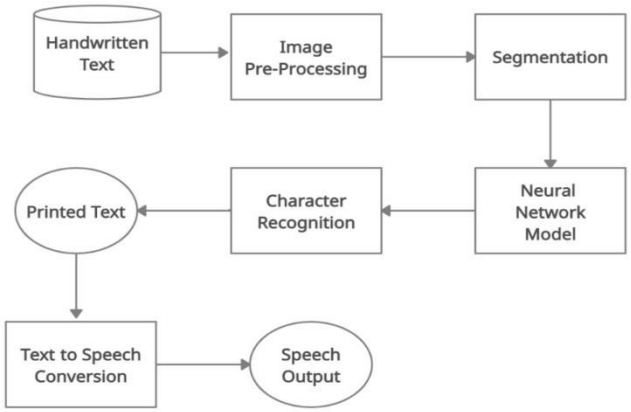

Fig - 1: An overview of the system

4. VARIOUS MODULES IN THE PROPOSED SYSTEM

The design of the proposed Handwriting recognition system can be broadly categorized into two significant parts – front-end and back-end, which is divided into five modules – Registration and authentication, Image processing, Neural network modeling and training, Character recognition and Text-to-speech conversion. For the front-end, Flutter is used. Flutter is an open-source UI software development kit created by Google. It is used to develop applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web from a single codebase. And for back-end keras, opencv, tensorflow lite, google text-to-speech, and firebase are used.

4.1 Registration and Authentication

The registration module consists of the user's sign-up and login procedures. The user can create an account by adding an email-id and password and can login from anywhere at any time using that email-id and password.

4.2 Image processing

The image processing module is divided into two phases:
a) Image Uploading, b) Image Preprocessing

In Image Uploading the image can be given as an input either by capturing the image through a mobile camera or by selecting an image from the mobile storage. Only one image can be given as the input. In Image Preprocessing, this phase is divided into two parts: Image Preprocessing and Segmentation. Image Preprocessing involves different stages of operations in order to enhance the image for further processing. Preprocessing involves resizing of image, gray-scaling and binarization. The input images can be of varying size, so the images are resized to a standard size before feeding the input to the model. Gray-scaling is the process of converting an image from other color shades to shades of gray. Gray-scaled images are typically composed of shades of gray, varying from black, at the weakest intensity to white, at the strongest. The values of intensity range from 0 to 255. Binarization is the process of transforming character image into the binary (0 and 1) form. It is an important stage to be performed on gray scale images. Segmentation and character recognition would be much easier once the process of binarization is carried out in a proper manner.

Image segmentation is the process of partitioning an image into multiple segments. Image classification is the process of predicting a specific class, or label, for something that is defined by a set of data points. A comparison takes place between the input and the stored values (patterns) to find the appropriate match class for the input images. The training and test dataset are reshaped so that the refined dataset can be given to the model.

Keras [12] is an open-source software library that provides a Python interface for artificial neural networks. It acts as an interface for the TensorFlow library. Keras has support for convolutional and recurrent neural networks and other common utility layers.

TensorFlow Lite [13] provides a framework for a trained TensorFlow model to be compressed and deployed to a mobile or embedded application. The computationally expensive process of training can still be performed by Tensor Flow in the environment that best suits it.

4.3 Neural Network Modeling and Training

A Convolutional Neural Network (CNN) [14] is used to perform recognition tasks and classification. The following steps are performed in the CNN model: 1. Input image is

fed into the CNN layers, 2. Image processing, 3. Comparison and prediction takes place 4. Output is the printed text. A CNN model for recognizing the handwritten characters was created using Keras and Tensorflow backend. Keras is an open-source software library that provides a Python interface for artificial neural networks and also acts as an interface for Tensorflow library.

The CNN model is followed by two bidirectional LSTMs consisting of 256 units each and the weights of the convolutional layers are initialized to he-normal. To introduce non-linearity in the convolutional layers, the activation function ReLu is used and Softmax activation function is used for the output layer.
The LSTM is designed to overcome the error backflow problems through carousels in their special units. This is all done with still a low computational complexity of *O(1)* and additionally the LSTM improves the RNN with the ability to bridge time intervals.

RNNs are limited in detecting cursive handwriting, where segmentation is difficult to determine. Therefore, Connectionist temporal classification is added as an output layer for sequence labeling tasks. The main different step to RNN is that the network output gets transformed into a conditional probability distribution over, for example, label sequences. Then the most probable labeling for a given input sequence is chosen. The key benefits of CTC are that it does not explicitly model dependencies between labels and it obviates the need for segmented data. Furthermore, it allows training directly for sequence labeling.

The Kaggle dataset for handwriting recognition is used in this model. This dataset contains 400,000 images of handwritten names of varying size and has image labels along with it. Here 30000 images are used for training the model and 3000 images for testing the model. The training and testing .csv file are accessed using pd.read_csv (). The dataset images are converted into grayscale and preprocessed i.e.; removing the white spaces and reshaping the image, before inputting the images to the model. CNN is used to take the input image and use filters on the pixels of the image. To create CNN, the following layers are defined-

1. Input layer: It takes the input image of size (256, 64, 1).
2. Convolutional layer: It is used to extract features from the given input image and ReLu activation function is used to add non-linearity to the network.
3. Batch Normalization layer: It is used to normalize the output of the previous layers.
4. Max pooling layer: It is used to reduce the mobility of the feature map to prevent overfitting.
5. Dropout layer: It is used to prevent the model from overfitting.
6. Dense layer: A fully connected layer which connects the previous layers to the next layers.

The last two layers are of LSTM, which learns from the extracted features obtained from the previous layer and predicts the output. The output layer uses the activation function called Softmax, which is used to predict the probability distribution. It is used to normalize the outputs, converting them from weighted sum values into probabilities that sum to one. The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. The ReLu activation function is used here. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

To calculate the loss function, CTC loss function is used where all the possible alignments' scores of the ground truth are summed up. In this manner, it is not significant where the character occurs in the image. After the model is created and loss is calculated, the model is configured with loss using model.compile (). The model is trained using model.fit () with epoch_size=50 and batch_size= 128. For the model, the model has 21 layers, all the layers and summary of the model is given in figure 2.

4.4 Character Recognition

The handwritten character image is converted into printed text by sending the image to the model. In the model, each character is recognized after preprocessing and they are converted into readable text.

4.5 Text to Speech conversion

The printed text is converted into audio by sending the printed text to the text-to-speech converter. Google Text-to-Speech is used for this. It is a screen reader application developed by Google for its Android operating system. It powers applications to read aloud (speak) the text on the screen with support for many languages. This audio feature can be used by the user only if they want to hear it. However the audio is generated after character recognition.

5. TECHNOLOGIES USED

Front-end:-
- Flutter

Back-end:-
- Keras
- OpenCV

- Tensorflow Lite
- Google Text-to-speech
- Firebase

Model: "model"

Layer (type)	Output Shape	Param #
input (InputLayer)	[(None, 256, 64, 1)]	0
conv1 (Conv2D)	(None, 256, 64, 32)	320
batch_normalization (BatchNo	(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 (Batch	(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 (Batch	(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

Fig - 2: Model summary

6. EXPERIMENTAL RESULTS

Training loss is the error on the training set of data. Validation loss is the error after running the validation set of data through the trained network. The loss obtained by the model is shown as a graph in figure 3, i.e.; how well or poorly the model behaves after each epoch. The red line represents the training loss and the blue line represents the validation loss. Our aim is to make the validation loss as low as possible.

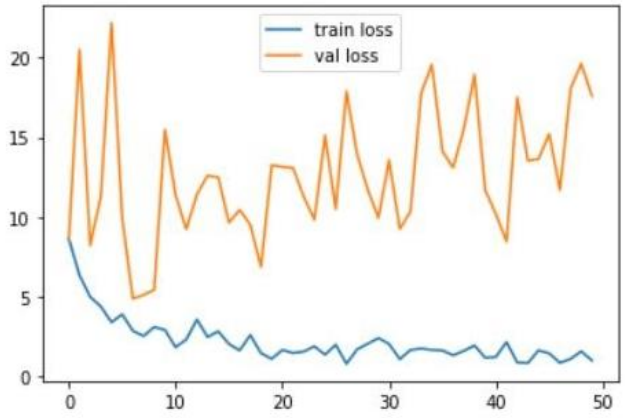

Fig - 3: Loss graph of the model

The accuracy of the model is represented in figure 4. The accuracy of a model on examples it was constructed on is called as the training accuracy and the validation accuracy is the accuracy you calculate on the data set you do not use for training, but you use (during the training process) for validating (or "testing") the generalization ability of your model or for "early stopping". The blue line indicates the training accuracy and the red line indicates the validation accuracy.

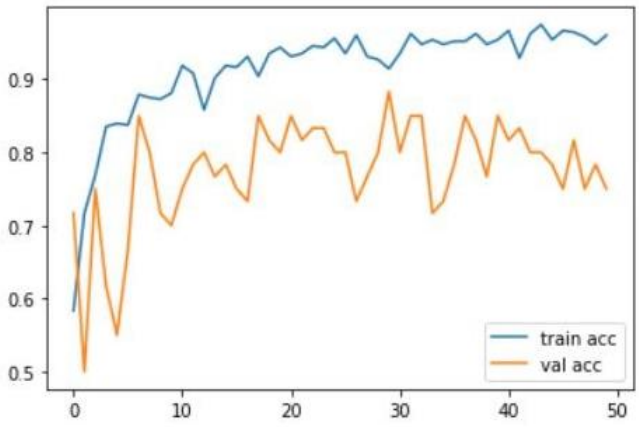

Fig - 4: Accuracy graph of the model

The login page of the system is shown in Figure 5. The user can login to the app using their credentials.

The image can be captured using the mobile camera or by uploading it from the device storage. This is shown in figure 6.

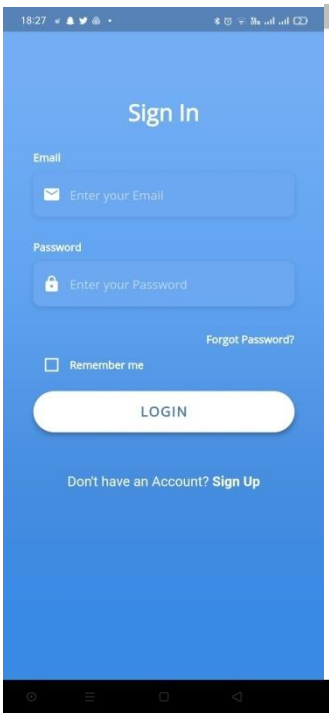

Fig -5: Login page

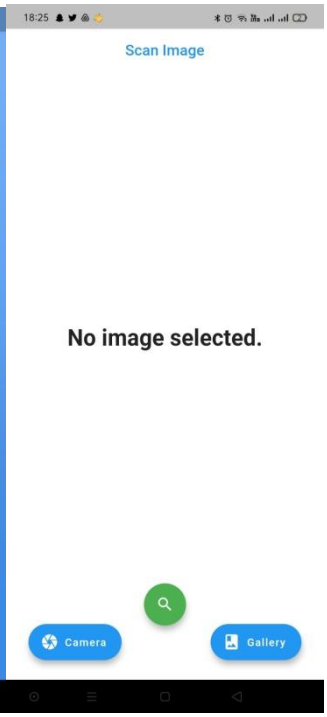

Fig -6: Scanning image

Here the image is captured using the device camera. It is shown in figure 7. The captured image is given to the model and it is preprocessed. Thus the printed text output is obtained. The text-to-speech conversion is done here and the audio output is generated. The generated printed text output is shown in figure 8.

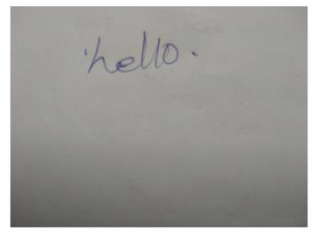

Fig - 7: Image capturing

Fig -8: Printed text output

CONCLUSION

A mobile application is designed to provide a facility to convert written text into printed text. Along with this, Text-to-Speech is incorporated to help people who have trouble reading on-screen text. This application in the wider range will help many people in their day to day activities. The model created for handwritten character recognition has proved to be of fairly good efficiency.

FUTURE SCOPE

In future work, we train the model to read a whole handwritten or non-handwritten document and convert the printed text to different languages. We also incorporate a personalized scanned file storage set-up for each registered user. The model can be trained further for specific purposes.

REFERENCES

[1] H. Zeng,(2020)An Off-line Handwriting Recognition Employing Tensorflow. International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE).

[2] A. Beltrán and S. Mendoza, “Efficient algorithm for real-time handwritten character recognition in mobile devices “,2011, 8th International Conference on Electrical Engineering, Computing Science and Automatic Control, 2011, pp. 1-6, doi: 10.1109/ICEEE.2011.6106583.

[3] H. A. Shiddieqy, T. Adiono and I. Syafalni, "Mobile Client-Server Approach for Handwriting Digit Recognition'',2019 International Symposium on Electronics and Smart Devices (ISESD), 2019, pp. 1-4, doi: 10.1109/ISESD.2019.8909448.

[4] R. Vaidya, D. Trivedi, S. Satra and P. M. Pimpale, "Handwritten Character Recognition Using Deep-Learning",2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018, pp. 772-775, doi: 10.1109/ICICCT.2018.8473291.

[5] H. Du, P. Li, H. Zhou, W. Gong, G. Luo and P. Yang, "WordRecorder: Accurate Acoustic-based Handwriting Recognition Using Deep Learning,"IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, 2018, pp. 1448-1456, doi: 10.1109/INFOCOM.2018.8486285.

[6] Kavitha, D., & Shamini, P. (2016),"Handwritten Document into Digitized Text Using Segmentation Algorithm", An International Journal of Advanced Computer Technology. Retrieved from https://ijact.in/index.php/ijact/article/view/465

[7] Chammas, C. Mokbel, R. Al Hajj Mohamad, C. Oprean, L. L. Sulem and G. Chollet, "Reducing language barriers for tourists using handwriting recognition enabled mobile application,"2012 2nd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA), 2012, pp. 20-23, doi: 10.1109/ICTEA.2012.6462868

[8] S. B. K.S., V. Bhat and A. S. Krishnan, "SolveIt: An Application for Automated Recognition and Processing of Handwritten Mathematical Equations, "2018 4th International Conference for Convergence in Technology (I2CT), 2018, pp. 1-8, doi: 10.1109/I2CT42659.2018.9058273.

[9] T. Mantoro, A. M. Sobri and W. Usino, "Optical Character Recognition (OCR) Performance in Server-Based Mobile Environment,"2013 International Conference on Advanced Computer Science Applications and Technologies, 2013, pp. 423-428, doi: 10.1109/ACSAT.2013.89.

[10] V. V. Mainkar, J. A. Katkar, A. B. Upade and P. R. Pednekar, "Handwritten Character Recognition to Obtain Editable Text," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), 2020, pp. 599-602, doi: 10.1109/ICESC48915.2020.9155786.

[11] Katsouros, Vassilis & Papavassiliou, Vassilis (2011), Segmentation of Handwritten Document Images into Text Lines 10.5772/15923.

[12] https://www.websequencedaigram.com

[13] https://www.tensorflow.org/lite

[14] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[15] https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts