A Mobile Application For Handwritten Text Recognition

[1]Mr. Ved Bhanushali, [2]Mr. Harsh Zota, [3]Mr. Krimesh Shah, [4]Dr. Manimala Mahato
[1](Computer Engineering, Shah and Anchor Kutchhi Engineering College, Mumbai, India
Email: ved.bhanushali619@gmail.com)
[2](Computer Engineering, Shah and Anchor Kutchhi Engineering College, Mumbai, India
Email: hzota1042@gmail.com)
[3](Computer Engineering, Shah and Anchor Kutchhi Engineering College, Mumbai, India
Email: shahkrimesh3@gmail.com)
[4](Computer Engineering, Shah and Anchor Kutchhi Engineering College, Mumbai, India
Email: manimala.mahato@sakec.ac.in)

Abstract— **In this pandemic, one of the sectors which have been affected the most is the educational sector where students are required to attend lectures online and submit handwritten assignments. Students need to scan these handwritten assignments with the help of pdf scanners. Teachers need to maintain online records of each PDF submitted by students. It's very cumbersome for the teachers to grade the assignments via pdf. Here the solution to convert the handwritten documents to text comes into the picture. It is an arduous task that requires a lot of data to train the model. This project focuses on algorithms including CNN, RNN, and CTC. The only way in which people can take an advantage of this project is by rolling out an android application. The hardship of this project comprises of different handwritings of each individual. Other applications include preserving manuscripts, editing a written letter into text. Accuracy varies upon the writing style of an individual. Accuracy might be low in case of shabby handwriting and would even touch a perfect score if written neatly.**

Keywords— Pattern recognition, Handwritten documents recognition, Convolutional Neural Networks (CNN), Recurrent neural network (RNN), Connectionist Temporal Classification (CTC), Deep Learning.

I. INTRODUCTION

Since computers have evolved, a lot of research has been carried in the area of Computer and Human interface. The Education industry is one of the most affected industries due to the pandemic today. A majority of colleges have shifted to online learning management systems to continue their classes in order to encourage social distancing. This has resulted in many colleges encouraging handwritten assignments and exams. These assignments and exams have to be scanned, uploaded, and sent to the respective faculty. The faculty in turn has to download and take a look at these. The following problems arise in this situation:

1] The student's handwriting may be illegible.

2] The student's handwriting may be legible but the contrast in the picture may not highlight the written letters properly.

This makes it difficult for the faculty to read what the students have written and provide marks accordingly. Hence, to provide a solution to this problem we have designed an application replete with a handwriting recognition system. In handwriting recognition (HWR) the tool interprets the user's handwritten characters or idioms right into an arrangement that the pc. There is a large need these days for saving information available in documents in written format on a computer storage device and then retrieving it later. The input device usually accommodates a stylus and a touch screen. There are many degrees of HWR, beginning from the popularity of simplified individual characters to the recognition of complete phrases and sentences of cursive handwriting.

Automated handwriting recognition has attained a noteworthy accomplishment in targeting specific applications like address recognition on postal pieces.

In this application, the users would be provided with the option to either scan the handwritten document via camera or upload an existing image from the gallery. After the user opens the application and scans the image necessary pre-processing steps will be applied to the same image. Firstly, one of the major problems is the clarity of the image and in order to resolve that necessary noise removal and increased contrast, techniques are first applied to the image. The next step involves segmentation of the image. After segmenting the letters, the image then goes through multiple CNN and RNN layers. After this step, the model is successfully able to decode and extract the letters from the image. This is further discussed in the implementation section of our paper below.

II. LITERATURE REVIEW

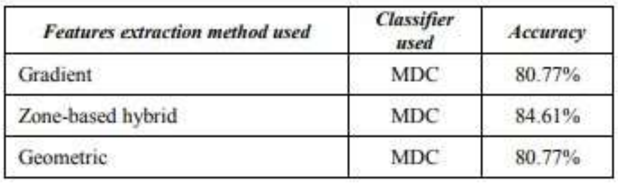
2.1 Handwritten Character Recognition Using Unique Feature Extraction Technique.

This research proposes a novel feature extraction technique based on selected properties of existing techniques. Also three different neural
Networks are compared and a conclusion is reached on which of these neural networks is the most efficient of the lot.

In this paper, two different proposals were put forward. The first one was a new feature extraction technique that combined select characteristics of three different existing feature extraction techniques. The results showed that the proposed technique was comparatively more efficient. The demerits of the existing algorithms are that they consider a restricted number of features to aid their extraction process. The suggested approach, on the other hand, combines the best features of all of these algorithms to produce a more reliable extraction strategy that outperforms the existing ones.

The second one was a comparison of the performance of three distinct neural networks for two different feature extraction techniques: geometric and gradient. The neural networks were: the MLP neural network using the backpropagation algorithm, the MLP neural network using the Levenberg Marquardt algorithm, and the CNN.

The results portrayed that CNN was followed closely by the Levenberg-Marquardt algorithm. The proposed algorithm was trained with 78 images and tested on 26 images. With 78 character images, the training matrix has dimensions of 145*78 and was used for the classification of the test character. The suggested algorithm's performance was evaluated using the above-mentioned training and testing sets, and the results were compared to a geometric, zone-based hybrid, and gradient features extraction approaches. The analysis of the three types of feature extraction algorithms according to the paper is shown below.



Features extraction method used	*Classifier used*	*Accuracy*
Gradient	MDC	80.77%
Zone-based hybrid	MDC	84.61%
Geometric	MDC	80.77%

2.2 Optical Character Recognition for English Handwritten Text Using Recurrent Neural Network.

The proposed approach during this paper was:

A-Dataset: Gathering transcribed information from various writers.

B-Pre-processing: Resize the picture to 30*30 pixels and Convert to the grayscale structure. Alter pixels force. Include cushioning of 2 pixels on all sides.it is a grayscale picture of size 128×32.

C-Training and Testing: Set Randomly split data into training and testing sets. the characters are anticipated precisely at the position they show up in the picture.

D-Recurrent Neural Network: As it is required to foresee the expression of a sentence, the past words are required and consequently there is a need to recollect the past words so RNN was used. The well-known. Long Short-Term Memory (LSTM) execution of RNNs was utilized, as it can spread data through longer separations and gives more vigorous preparing qualities than vanilla RNN.

E-Output: The output image is given as formal printed text.

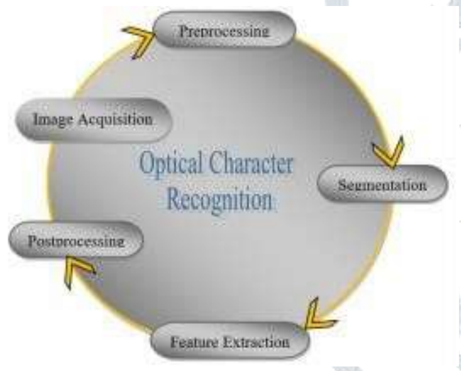
So, in general RNN algorithm was used so as to get an output as printed text with 90% accuracy.

2.3 *A Scalable Handwritten Text Recognition System.*

The team believed that the most difficult aspect of designing an HTR system is getting adequate amounts of high-quality training data, rather than modeling. Our online handwriting recognition system's ink data was used to train the line recognizer in their current OCR system. Their HTR (Handwriting Text Recognition) Is trained on images generated from the online handwriting recognition system's trajectory data. The photos are treated with an image degradation pipeline that applies realistic image changes to the data after producing clean handwriting images from trajectory data. They employed a handwriting synthesis pipeline to enrich the heterogeneity in their training dataset in order to get better results, in addition to the online handwriting data they acquired. They tested two distinct model architectures for the line recognizer: An LSTM-based architecture that is comparable to many state-of-the-art approaches. Recurrent architectures, on the other hand, have the drawback of being difficult to train and run on specialized hardware. As a result, we proposed a fully feed-forward network architecture with similar accuracy to our LSTM-based model. While HTR research frequently focuses on the line recognition engine, it is only one component of a complete text recognition system. We have detailed the extra steps they took to incorporate HTR support into a text recognition system that includes text detection, direction identification, script identification, and text line recognition models. Sections in the paper explained the following things: Description of line recognition model. The description on how to generate training images suitable for training an offline recognizer using online handwriting data, comprising the rendering process, the image degradation pipeline, and the handwriting synthesis system. How the line recognizer is integrated into our comprehensive OCR system is described.

3.4 *Handwritten Character Recognition to Obtain Editable Text*

The team developed an android app giving access to the user to recognize the text from saved images from the gallery or capture images through the camera. It can make users easily edit and store the written data in text file format. The app uses a camera of an Android mobile device that takes the input. The input could be a binary image scanned by the camera. The OCR engine processes the image data and converts it into text. The OCR technology works by scanning the document or image with a scanner first. Once the image is scanned OCR software converts the image into a black and white version. Then the image is analyzed by dark and bright areas. When a dark area is identified as the character and a bright area is identified as the backdrop, the dark area is taken into consideration for further processing. The steps used in the OCR algorithm is shown in the image below:



The steps involved in their work that was used to successfully implement the concept were:
1. Image acquisition by the android camera
2. The image is loaded into the Android Studio Graphical User Interface (GUI).
3. Preprocessing of the image.
4. Feature extraction from the supplied image.
5. OCR was used to convert recognized data into text format.

To conclude their work and according to the paper
Implementation gives conversion of Handwritten
Using the Android app, convert character recognition into editable text. The image is captured by the camera and loaded into the android app and choice is provided to the user to select a part of an image that is to be converted. Further processing is done by the OCR engine and produces the converted text on the screen. The recognized text is saved in text format. A choice is provided for editing the identified text and saving it in the appropriate location.
The test results were as follows:

Input	Total Character	Recognized Character	Accuracy
Handwritten text	221	208	94.12%
Printed text	221	221	100%

More accuracy was achieved when text is in printed form rather than in handwritten.

III. METHODOLOGY

For this paper, we have proposed a method that involves multiple Convolutional Neural Networks. (CNN) and Recurrent Neural Networks(RNN). The following stages make up the procedure:

1. Collecting the dataset.
2. Preprocessing the scanned image.
3. Preparing the model.
4. Train the model with the collected dataset.
5. Convert to Tflite file to be used in mobile applications.

The process is described as follows:

3.1 Collecting the dataset

There were multiple datasets scrutinized before finalizing one. Datasets from Kaggle, Bretta dataset, and IAM Dataset were looked at and studied. Datasets from Kaggle and Bretta were not up to the mark as per the requirements of this particular project. IAM dataset on the other hand contains 13353 images of handwritten lines of the text created by 657 writers.
This annotated dataset was used to train the created model.

3.2 Preprocessing the scanned image

This process is further divided into other small processes which in turn helps to create the model.

3.2.1 Image Preprocessing

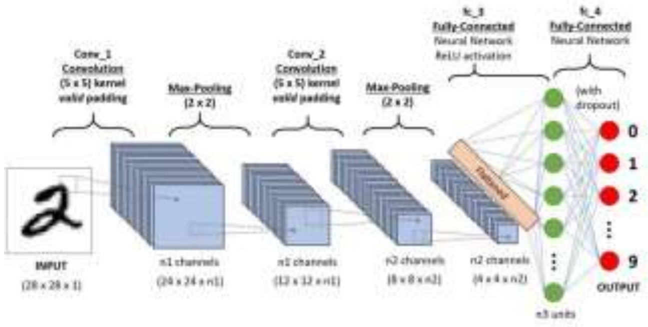
Here the image is preprocessed. For preprocessing we remove the noise from the image. Noise in an image is the variation of contrast or color information. This may vary due to various reasons like the angle from which the image is taken, also the brightness in the room while the image is taken. Factors like how steadily the image is clicked are also considered. Here to get rid of the noise the Time domain Wiener Filter method was used. The Weiner filter smoothens the image.

3.2.2 Image Segmentation

To get higher accuracy, divide and conquer were used. The scanned paragraph was initially divided into lines and then into words. Then it is then fed into the model. Here we group together pixels with similar attributes, all handwritten content in black Ink. The paragraph is divided into multiple lines and those lines are divided forward into multiple words.

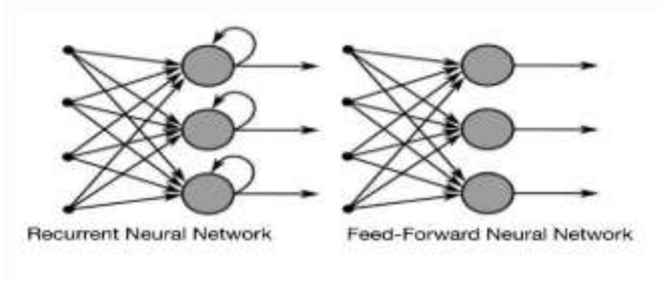
3.3 Preparing the model

The model is composed of multiple CNN and RNN layers. The model is composed of multiple CNN and RNN layers.
Convolutional Neural Networks (CNN): They are a class of deep neural networks that are mainly used for visual image classification. They are composed of multiple layers(Fig) of artificial neurons which are able to calculate the weighted sum of multiple input and output values. CNN performs best on sequential data.



Figure

Recurrent Neural Networks(RNN): They are networks composed of multiple layers. They are able to handle robust data with ease. They give better results as compared to CNN as they not only work for sequential data, but they extract data from key data point which make then much more efficient. They not only take data given from previous layers but also process the data which has just been processed by them (Fig).



Figure

In the gives the system a combination of both CNN and In the gives the system a combination of both CNN and RNN layers was used to get the most optimum results.

A combination of 5 CNN layers along with 2 RNN layers has been used.
Along with this, we use a Connectionist Temporal Classification layer suggested by Herald Scheidl.
Connectionist Temporal Classification Layer(CTC): This layer helps to decode the text from binary to the actual text which is to be given as output to the user.

3.4 Training the model with a collected dataset

Once the model is ready it has to be trained on the collected dataset. By doing this the model will be able to give the required results. The data was divided as follows. 95% of the data was used for training, rest 5% was kept for testing the model once it was trained using the dataset.

3.5 Convert to Tflite life to be used in the mobile Application

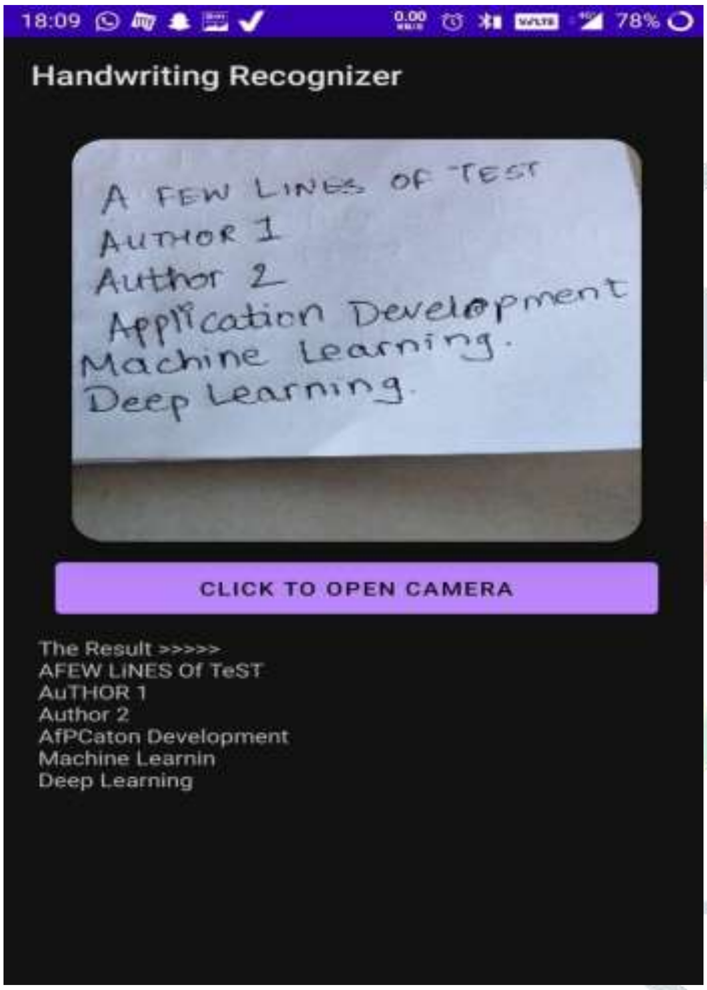
At this point there are 2 options to run the machine learning model in the android application, one is by creating an API other is by simply creating a Tflite file. The Tflite file makes it very much easy to perform the same task. Instead of making a round trip to the server then model, Tflite simply runs the model in the mobile application.
Tflife is most optimum when the size of the model is small and hence you do not need an external server. There were multiple other reasons to select a Tflite method and not a server because, when you use a Tflite model, no data leaves the user's device, the model can work offline also and no latency is observed.

IV. ANDROID APPLICATION

Once the model is ready along with the Tflite file to export the trained model, the android application is built. The technology to build the android application was Android studio. The language used to create this was Java. The Tflite life is then imported into the code and different libraries like TensorflowLite Task Library TensorflowLite Android Support Library, are used to deploy the model in the application. The image is clicked with a few editing options and once the user feeds the image, to the deployed Tflite model within the application.
The UI of the application consists of an option to select the open camera option and click the image. The rest is done all by itself on the model.

V. RESULTS

After training and deploying the model into the mobile application we observe that there are accurate results of up to 80%. The accuracy is calculated using the logarithmic function with the loss value calculated by the CTC layer. It is also observed that results were better with the better quality of the images.
Given are a few images of the scanned images with results.



```
The Result >>>>>
AFEW LiNES Of TeST
AuTHOR 1
Author 2
AfPCaton Development
Machine Learnin
Deep Learning
```

VI. CONCLUSION

In this project, a handwriting recognizer is made. The application is in process of making use of NLP and OCR to detect the words written in the handwritten document. The accuracy of the word detected depends on many factors, which also include the physical quality of the paper. The parchment if in a bad shape can give vague results.
For such scenarios, we make our program as optimum as possible. Here we use methods like noise removal and increase of contrast so the best results can be obtained. Using such factors we can make a model predict the best possible outcomes.
At the current stage, our program is able to detect a small paragraph written on the document.

VII. REFERENCES

[1] S. A. Ayyadevara, P. N. V. S. R. Teja and M. Rajesh Kumar, "Handwritten Character Recognition Using Unique Feature Extraction Technique," 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2018, pp. 12391243, doi: 10.1109/RTEICT42901.2018.9012248.

[2] R. Parthiban, R. Ezhilarasi and D. Saravanan, "Optical Character Recognition for English Handwritten Text Using Recurrent Neural Network," 2020 International Conference on System, Computation, Automation and Networking
(ICSCAN), 2020, pp. 1-5, doi: 10.1109/ICSCAN49426.2020.9262379.

[3] R. R. Ingle, Y. Fujii, T. Deselaers, J. Baccash and A. C. Popat, "A Scalable Handwritten Text Recognition System," 2019 International Conference on Document Analysis and
Recognition (ICDAR), 2019, pp. 17-24, doi: 10.1109/ICDAR.2019.00013.

[4] 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.

[5]https://fki.tic.heia-fr.ch/databases/iam-handwritingdatabase#:~:text=The%20IAM%20Handwriting%20Database%20contains,writer%20identification%20and%20verification%20experiments.&text=All%20texts%20in%20the%20IAM,the%20LOB%20Corpus%20%5B5%5D.

[6]https://www.analyticsvidhya.com/blog/2021/05/convoluti onal-neural-networks-cnn/

[7]https://www.analyticsvidhya.com/blog/2017/12/introduct ion-to-recurrent-neural-networks/

[8] https://towardsdatascience.com/beam-search-decodingin-ctc-trained-neural-networks-5a889a3d85a7