Hindi Character Recognition

Sameeksha Sharma[1], Sanskriti Ahlawat[2], Sakshi Gupta[3], Prerna Chaudhary[4]
[1,2,3,4]*Department of Computer Science and Engineering, Meerut Institute of Engineering and Technology, Meerut, U.P., India*

Abstract **- In this paper, we gave a new technique and theory of Hindi Character Recognition. OCR is a very trendy topic nowadays in research and development field. Devanagari Script provides a bunch of 49 characters which includes 13 vowels and 33 consonants. Many Indian languages like Hindi, Nepali, Sindhi uses Devanagari script. It structures the creation of many languages like Hindi, which the most spoken language and also the National Language of India. In this research the aim & focus is given to detect the characters of Hindi characters from images. In this paper, we gave a new theory of extracting and detecting the Hindi vowels and consonants from the image file. Since it is the hot topic in R&D, we can find multiple theories to get the characters from the image file. In our theory, we have used EasyOCR API for extracting Hindi characters from the image and CNN2D architecture for recognizing characters from the hand gestures.**

Index Terms **- CNN2D, AVERAGEPOOL2D, EASYOCR, PYTESSERACT, OCR**

I.INTRODUCTION

Now a days, people prefer to communicate in the natural languages. As it is very easier and comfortable to communicate in their native language. India is a country where we find many languages lie Hindi, Gujarati, Punjabi, Urdu, Telegu and so on. It has 22 different languages and 11 different scripts to write them.

Time by time automation is increasing in every field and nowadays people showed their interests in automation of character recognition field as it makes people communicate easily and fast.

Hindi is the most popular language. Handwritten text recognition technology is quite helpful and needed in today's world. The physical data formation is prone to errors, it can help in storing data correctly and efficiently.

Storing text digitally is that it can be accessed from any place. This is a very important advantage; we don't have to be at the place where data is stored.

This technology has made storing and analyzing data much easier. Devanagari is an Indian script. Our dataset contains 36 characters. Most of the Indian literature such as Vedas, Ramayana is written in Devanagari. The system is trained with CNN2D architecture with Sequential model. Handwritten text recognition technology is very helpful, if the text is present in a digital format then error scanning mechanisms and autocorrect tools can help in storing data correctly and efficiently. People prefer to use their native language at their workplace and for communication.

The theory of character recognition got much highlights due to its applications in various fields like, online checking of papers. Hindi Character Recognition from the image is very tough task to perform because of various reasons like, it is written in various methods and the size and orientation of the characters. Hence Devanagari should be given more attention in development of character recognition field.

In this paper, Various character recognition approaches have been applied such as EasyOCR, CNN2D, Average pooling, max pooling. OCR is one of the most experimented area of machine learning and deep learning.

II. PROPOSED SYSTEM DESIGN

The given theory consists three phase:

COLLECTION AND CONVERSION OF DATASET

In this phase, we collected the character dataset from UCI Machine Learning Repository. The DHCD (Devanagari Character Dataset), had a training set of 72,000 total sets of training and testing images for 36 characters from क to ज्ञ. The dataset was arranged in

two separate folders: Training and Testing. Each of them consists of 36 sub folders for each character.
The dataset consists of 72000 rows (sample images), and 1025 columns. Each row contains the pixel data ("pixel0000" to "pixel1023"), in grayscale values (0 to 255).

TEXT EXTRACTION FROM IMAGE USING EASYOCR

Since many days, OCR (Optical Character Recognition) is been most searched and developed field. Several researchers worked on multiple theories to get the best and accurate method for OCR. OCR is method of extracting the text part from the image file. Our Project Used EasyOCR method for the same.

EasyOCR includes following steps in its backend:

1. **Pre-processing**: - Pre-processing means to remove noise and errors from the dataset. So that maximum accuracy can be achieved and model can be trained with the best possible data.

We need to apply following processes on the raw data files:

1. Threshold: It refers to conversion of image file to binary data. For faster execution and better understanding
2. Noise reduction: We need to remove the unwanted data or pixels from the image. It is done by various techniques like, applying morphological operations on it.
3. Normalization: we need to reshape the images either of 32*32matrix or 64*64matrix after the segmentation process.

2.Segmentation: - In this stage we break down the image consisting of sequence of characters into various sub images of individual characters. After that we do labeling process to assign number to each character or each sub image. This phase plays very crucial role in OCR as, we get each separated words or lines which led to detection of Script.
Once the system (OCR model) identifies the block of text, it can easily extract the individual lines, words and even the characters.
OCR system uses dimensional information of images for segmentation and recognition.
Transferred Learning is a ML theory in which the model gets training from other pre-trained models and pre-processed datasets to get the useful information about the simple rules.

3. Feature Extraction: After segmentation, we need to find various features like top & bottom end, height-width of characters, etc.
Zoning: Frame containing the text is divided into various zones. And then the density of the zone is calculated by following formula

$$Density = \frac{Number\ of\ pixels\ in\ zone}{Total\ Pixels}$$

4. Classification: Classification is process of determining the class of unknown pattern.
Multiple researches shown that Support Vector Machines (SVMs) are best for this process and can be utilized with images and human written characters detection. SVM is accurate in high dimensional space, so SVM can be used for our proposed theory too. But the disadvantage of SVM is that, it doesn't give accuracy with large datasets. As time required = $(\text{dataset size})^3$. Which is the biggest challenge to overcome when we deal with large datasets. So EasyOCR adopted a new technique in which training is done using SVM on bulk of Nearest neighbors, and it is thus known as "SVM-KNN".
The tool uses KNN in its initial stage and then it performs SVM when the dataset becomes smaller. But it is more complex and relevant set of data which requires very careful discrimination.

GESTURE BASED CHARACTER RECOGNTION

In Character Detection area, gesture detection is very difficult and challenging task to accomplish. Several research are done to achieve the best accuracy and better results for the same.
We trained the model using the Neural network technology, CNN2D architecture.

It includes following steps for training the model:

1) **Pre-Processing:** images in the dataset need to be cleaned and we need to remove noise from the images using various techniques like gaussian blur.

2) **Training Model:** Model is trained using DHCD (Devanagari Character Dataset), found from UCI machine learning repository.
We used CNN2D architecture with sequential model for the same. It includes layers like Conv2d, Average Pool etc.

3) **Visualizing:** We visualized the accuracy and loss of the model using matplotlib

ANALYSIS

We used the accuracy score to find the performance of our trained model of gesture-based recognition. The accuracy gives us the percentage of our correctly classified test data. More the accuracy score of the model, better it is.

III ALGORITHMS

First Attempt:
Architecture of Model
-- CONV2D --> MAXPOOL --> CONV2D --> MAXPOOL --> FC --> SoftMax --> Classification

Second Attempt
Architecture of Model
-- CONV2D > AVERAGEPOOLING2D > DROPOUT > CONV2D > AVERAGEPOOLING2D >
DROPOUT > FLATTEN > DENSE > DROPOUT > DENSE

Algorithm for extracting characters from image

1. Downloading the Hindi recognizer module for EasyOCR; reader = easyocr.Reader(['hi'])
2. Reading the image using OpenCV/PIL
3. Giving the image as a input to "reader.readtext(filename)" function
4. EasyOCR will extract the Hindi characters and give us in text format.

Algorithm for Gesture based Hindi Character Recognition

It includes two phases; 1st phase is training & testing the model & 2nd phase is using the model.

A. Algorithm for Training and Testing the model: -

1)Downloaded Dataset includes, png format images of resolution 32*32, so we need to convert the dataset to csv file.
2) We fetched all the images and stored the binary formatted value of image in csv.
3) Dataset is ready to use.
4) After getting the dataset, we will train the model.
5) For training CNN2D sequential model is used.
6) First of all we need to prepare two parts of dataset for training and testing purpose.
7) For dividing, firstly the dataset is shuffled and then divided to 80-20 ratio.
8) Since the dataset is all set, prepare the model's architecture.
9) Layers of sequential model are:
CONV2D > AVERAGEPOOLING2D > DROPOUT > CONV2D > AVERAGEPOOLING2D > DROPOUT > FLATTEN > DENSE > DROPOUT > DENSE
10) Use the activation function as ReLu.
11) After passing from all these layers, we will fit our training data to Model. And set epochs as 35. with batch size of 64
12) After finishing it, we will send testing data to evaluate the testing.
13) Visualizing the results using matplotlib module.
14) Saving the model.

B. Algorithm for using the model: -

1) Load the model
2) Load the module OpenCV for getting live frames from webcam
3) Setting the upper and lower range of blue color, for detecting the blue color object.
4) Applying flip, cvtColor, medianBlur, GaussianBlur & threshold layers of OpenCV into frame for removing noise and detecting the blue color.
5) Tracking and tracing of blue object in the frame.
6) If the Blue object is not found, we will send the image for prediction to our model.
7) Before prediction we need to preprocess the image by, resizing it, converting to NumPy array, and reshaping it.
8) This array is used as parameter for keras function "predict".
9) Predict function gives some value between 0 to 37.
10) This value is searched in dictionary of characters (we already made to store characters)
11) If found, value is printed.

IV. EXPERIMENTAL RESULT AND DISCUSSION

B. We have Successfully developed HindiOCR tool's dataset for experiments. Handwritten cahracters are stored in Image format and then segmentation is done for extracting every individual characters from it. All the experiments were performed on jupyter notebook. The goal of our project is to achieve

comparable accuracy. Our approach could be useful to be applied to character recognition tasks when there are limited resources. It went on like we first used simple algorithms to make sure the data was formatted correctly and that our approach would work, and then moved on to more complex algorithm. Each of the following are different characters, although some of them appear quite similar hence this is the problem that our model attempted to resolve. We measured success by measuring how many of the test set images were correctly categorized into their respective category bin out of all the categories. We didn't choose top 5 accuracies because it does not make sense to allow a model to guess multiple times on character recognition. It is very important to be correct on the first try of our project. If accuracy is already high, suppose top 5 accuracies are likely 100% or close to it, then we selected the top one.

LOSS GRAPH COMPARISON

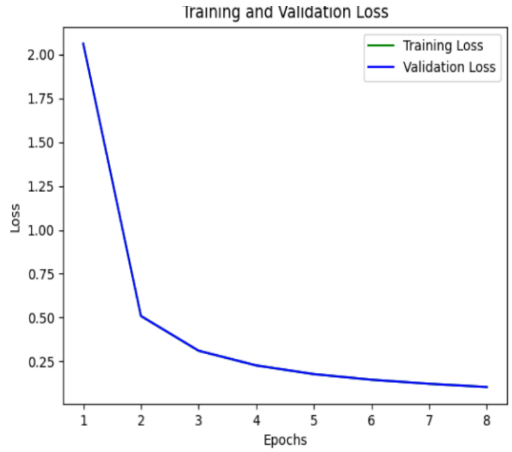

Fig : AVERAGEPOOL "loss" and "Val_loss"

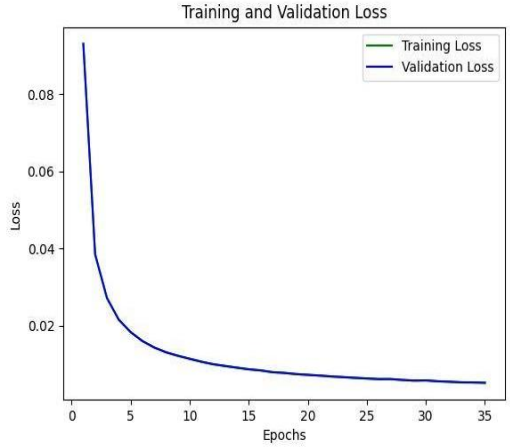

Fig : MAXPOOL "loss" and "Val_loss"

ACCURACY GRAPH COMPARISON

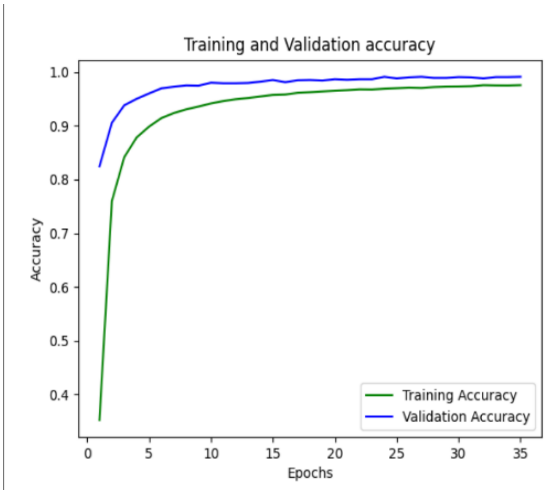

Fig: AVERAGEPOOL "Accuracy" and "Val_Accuracy"

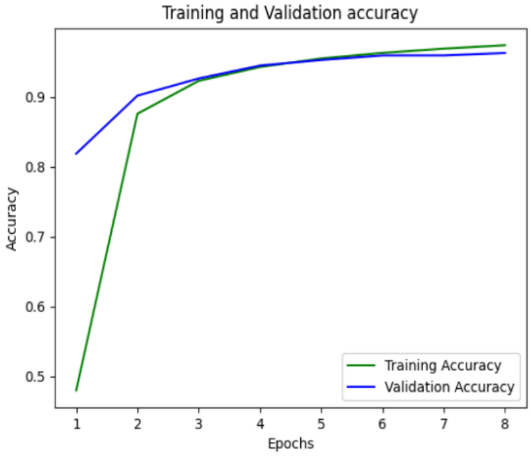

Fig: MAXPOOL "Accuracy" and "Val_Accuracy"

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        832
_________________________________________________________________
average_pooling2d (AveragePo (None, 14, 14, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 10, 10, 64)        51264
_________________________________________________________________
average_pooling2d_1 (Average (None, 3, 3, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 3, 3, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 256)               147712
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 37)                9509
=================================================================

Total params: 209,317
Trainable params: 209,317
Non-trainable params: 0
```

Fig: AVERAGEPOOL Model Summary

```
CNN Error: 3.75%
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        832
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 10, 10, 64)        51264
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 2, 2, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 37)                9509
=================================================================
Total params: 61,605
Trainable params: 61,605
Non-trainable params: 0
_________________________________________________________________
None
```

Fig: MAXPOOL Model Summary

V. CONCLUSION

In the project, work is done in recognition of handwritten characters in addition to various modifications. The proposed work focuses on recognizing handwritten Devanagari script with more accuracy and precision. EasyOCR is best and the most time efficient technique for recognition of characters. This technique is trained and tested on different types of images collected from various sources.

We achieved great and accurate outputs from given methods. Efficient automatic translation will help in conversion of Devanagari documents making it easier for a person, unfamiliar with Devanagari script, to understand them. It is almost impossible even for a human to achieve 100% accuracy in recognizing characters written by so many people but still we obtained promising and accurate results.

The highest detection accuracy 95% was gained by EasyOCR. Further, we again trained our model using our training dataset and we obtained 97% recognition accuracy, which is highest and very great results achieved.

Also, gesture-based character recognition proved to be the best of all models, by giving the accuracy of 97-98% on training data and 96% accuracy on testing data.

Hence, from above results of both the models we concluded that our Average Pooling Model with Conv2d is better for gesture-based recognition and EasyOCR is best for Devanagari OCR.

REFERENCES

[1] Hanmandlu, M., Ramana Murthy, O.V., Madasu, V.K.: Fuzzy Model Based Recognition of Handwritten Hindi Characters. In: 9th Biennial Conf. on Digital Image Computing Tech. and Appli., Glenelg, Australia, pp. 454–461.

[2] Lu, Y. and M. Shridar, “Character Segmentation in Handwritten Words - An Overview”, Pattern Recognition.

[3] P. Banumathi and Dr. G. M. Nasira, "Handwritten Tamil Character Recognition using Artificial Neural Networks", 2011 International Conference on Process Automation Control and Computing.

[4] V. Bansal and R.M.K. Sinha, “Segmentation of touching and fused Devanagari characters,” Pattern Recogn. 35(4), 875-893 (2002) [doi:10.1016/S0031-3203(01)00081-4].

[5] Handwritten Devanagari Character Recognition Using Layer-Wise Training of Deep Convolutional Neural Networks and Adaptive Gradient Methods (Published: 13 February 2018).

[6] Elliman, D. and I. Lancaster, “A Review of Segmentation and Contextual Analysis Techniques for Text Recognition”, Pattern Recognition, 23(4/5), pp. 337-346.

[7] Siham Tabik, Daniel Peralta, Andrs Herrera-Poyatos and Francisco Herrera, "A snapshot of image Pre-Processing for convolutional neural networks: Case study of MNIST", International Journal of Computational Intelligence Systems 10, vol. 555, no. 1, January 2017.

[8] Fatih Ertam and Galip AydOn, "Data Classification with Deep Learning using Tensorflow", 2017 International Conference on Computer Science and Engineering (UBMK).

[9] J. Pradeep, E. Srinivasan and S. Himavathi, "Neural Network based Handwritten Character Recognition system without feature extraction", International Conference on Computer Communication and Electrical Technology ICCCET 2011.

[10] This DCNN and Adaptive Gradient based neural model published in February 2018 reported maximum accuracy of 98%. Despite the higher recognition rate achieved by existing methods there was still room for improvement which was covered by our Average Pooling Model achieving an accuracy of 99%.