2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN) | 979-8-3503-3802-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/CICTN57981.2023.10141358

Devnagari Character Recognition using Optical Character Recognition (OCR)

Ananya Shukla
Computer Science Engineering
ABES Engineering College
ananya.19b101169@abes.ac.in

Ayush Sharma
Computer Science Engineering
ABES Engineering College
ayush.20b0103009@abes.ac.in

Akriti Aggarwal
Computer Science Engineering
ABES Engineering College
akriti.19b101027@abes.ac.in

Shikha Jain
ABES Engineering College
shikha.jain@abes.ac.in

Abstract—**In the age of digital applications, optical character recognition (OCR) is a useful tool. There are many different languages and scripts used around the world, with more than 300 million individuals in India speak Hindi as their first language. OCR previously created for the Devnagari script has a very low recognition rate. In this paper, we propose an OCR system based on a Convolutional Neural Network (CNN) model. It increases the handwritten data's efficiency or accuracy by 98.32%. We do several steps for getting this efficiency like preprocessing of data, grayscale images, binarization, reshaping, normalization, modelling, and Evaluation of the data with the help of plotting.**

Keyword: OCR, CNN, grayscale, binarization, normalization, modelling, plotting.

I. Introduction

A tool called Optical Character Recognition (OCR) turns printed or scanned picture materials into text documents. It can be stored in UNICODE or ASCII format once it has been converted to text. OCR is a subject that pertains to digital imagery and its uses. The main grounds for stressing the Devanagari script are its intricacy and the populace that speaks and writes it. It's difficult to create an algorithm for Devanagari OCR, as it is with any foreign language. It has 36 consonants (क, ख, ग, घ, ङ, च, छ, ज, झ, ञ, ट, ठ, ड, ढ, ण, त, थ, द, ध, न, प, फ, ब, भ, म, य, र, ल, व, श, ष, स, ह, क्ष, त्र, ज्ञ) and 10 digits (०, १, २, ३, ४, ५, ६, ७, ८, ९).

A fundamental property of Devanagari is that it has its own pronunciation style unlike any other language. The letters in this script vary in size and font. A computer's main task is to identify fonts. Capturing, preprocessing, and turning an image into a binary stream are all steps in the character recognition process. identifies character features in a binary stream and extracts them. In this paper, we employ the sequential Convolutional Neural Network (CNN) model. Given that it is a neural network, it is modelled after the human brain and functions as such. The machine learns in a similar manner to how people do.

II. Stages Of Character Recognition

Character recognition has six main stages:

A. Image Acquisition

Use a digital camera to take an image of the document or scan it, then save it on a computer with the appropriate image extension.

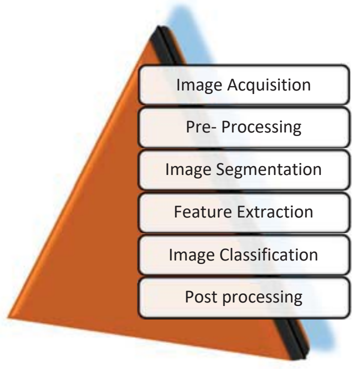


Fig. 1. Stages of Character Recognition

B. Pre-Processing

Because we are only interested in the character's pattern and not its color at this level, the saved image should be transformed from color to grayscale. This is because the grayscale image can be processed further with the least amount of memory use and information density. After conversion, the signal must be transformed into a binary image with just the values 1 and 0 of intensity. When the Devanagari script is converted to a binary picture, the characters have varying widths that must be fixed using image normalization techniques to create a consistent matrix, such as 16 by 16 or 128 by 128.

C. Image Segmentation

The core of OCR, image segmentation, divides the overall image into smaller sub-images based on the uniqueness of the content in order to boost the algorithm's hit rate.

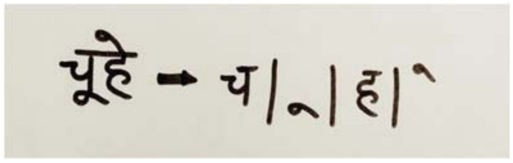


Fig. 2. Image Segmentation of a word

D. Feature extraction

In order to improve character recognition rates, this stage focuses on extracting a set of characteristics from the segmented image. These features are taken from global content that was made accessible following the segmentation stage. The major stage of the character identification procedure is called feature extraction; during this stage, each character is represented as a feature vector.

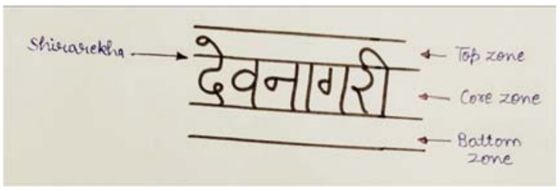


Fig. 3. Feature Extraction of a word

E. Image Classification

Following feature extraction, features are converted into feature vectors, which are then used by image classifiers like K-Nearest Neighbors (KNN), Bayesian Classifiers, Convolutional Neural Networks, and Hidden Markov Models (HMM). automated decision-maker.

F. Post-processing

At this stage, based totally on the decisions made in the classification stage, the recognized fonts are printed on the digital screen in an editable format.

III. RELATED WORK

An approach developed by researchers in [1] recognizes Hindi characters with 93% accuracy. Segment the images into known cases by classifying them with powerful filters, two distance-based classifiers, a degree splitting system, and a set of search algorithms, using binary function extraction and a modified back propagation neural community, developed by researchers in [2] a four-layer neural network to recognize Odia letters from 211 symbols, which performed more accurately than existing techniques. By employing Freeman chain code, SVM for classification, and Devanagari digits, Researchers in [3] presented threshold Abd blob evaluation for segmentation to detect overlapping portions of textual content regions and achieved an accuracy of 93%. In the feature extraction section on information of handwritten characters of diverse people from various eras, [4] suggested Euclidean distance-based completely k-NN classification using Gaussian and Robert filters applied to individual photographs. Gradient pictures were obtained (from 03 to 75, or H) and were then assessed and carried out with 95% accuracy on 7000 individuals of varied ages.

Researchers in [5] suggested using a help vector system (SVM) and a synthetic neural network for type to identify the output of characteristic vectors for photograph data retrieved using a gradient illustration. proposed. In [6] the records were used extensively by the authors. He achieved the best accuracy using a fixed set of 8000 samples of the Marathi script, normalized to a matrix size of 20x20.

According to [7], there is currently no single algorithm or OCR model that can recognize Devanagari characters with 100% accuracy, so existing classifiers that use convolutional neural networks to develop a set of rules show the satisfactory accuracy when compared to other classifiers. Additionally, even though there are numerous function extraction techniques using well-known changes, it has also been observed that noise filtering in the preprocessing section of the process can help reduce errors. enhanced algorithm. Using outstanding tools like pens, pencils, markers, doodles, and even Hindi handwritten characters with varied colorations and orientations, OCR reportedly recognizes all shapes of various handwritten characters with ease [8] Recognizable and effective but needs further improvement. It has some problems. One of the main concerns and something that needs additional development is that this OCR no longer recognizes pure forms and misses consonant components or half of the letters in Hindi scripts.

According to [9], a 48x57 input matrix produces better results than other options. The results section shows that the OCR recognition rate on actual Hindi files may be very high. However, various forms of preprocessing and neural network architectures may be investigated in the future to reduce detection costs.

When the numbers are trained separately and as well as characters separately before decision-making, the authors experienced very good results. The authors used feed forward neural network for classification stage, and they worked on the images captured with a digital camera. In [10] Their study compared the performance of the OCR algorithm in accuracy and the time to identify the character post processing.

Researchers in [11], to lower the noise levels and feature vectors produced by K-means clustering, the author suggested removing shirorekha in the preprocessing stage itself for the Hindi characters. Additionally, the author suggested utilizing a linear kernel-based technique for classification.

A text recognition system's two main steps are segmentation and classification. The segmentation process takes recognition units, often a character, out of the text [12]. According to [13], Each segmented character has specific attributes computed as part of the classification process, and these traits are then allocated to one of three classes: the correct class (correct recognition), the incorrect class (substitution mistake), or an unidentified class (rejection error).

According to [14-16], a good OCR system should have accuracy, adaptability, and speed as its defining characteristics. Dependence on font, size, and orientation has limited the performance of the majority of OCR systems created for Hindi. In [17], selection of characteristics affects how well these algorithms recognize objects. The majority of the current algorithms process the image in-depth before extracting the features, which lengthens the calculation time.

The methods for recognizing the fundamental symbols are covered in the current section. Researchers in [18], created classifiers based on artificial neural networks to categorize the symbols in Hindi fonts. The classification is carried out based on the attributes that have been derived from each particular sign or character. We used three separate approaches to compute each symbol's characteristics to verify that it was classified correctly. According to [19], these three separated approaches are:

1. A projection histogram based on mean distance
2. A projection's pixel-value-based histogram.
3. Vertical zero crossing

Researchers in the report [20-22] provided a comprehensive strategy for segmenting printed Hindi texts. The upper-strip is separated from the remainder of the character by a preliminary segmentation operation that extracts the header line. As a result, vertically separated character boxes are produced. These character boxes can be conjuncts, touching characters, characters with lower modifiers attached to them, characters combined with rakar symbols (a special kind of ligature made up of the addition of a consonant followed by a halanta to the letter "ra"), or any combination of these.

In the paper [23-25], Words cannot be recognized by this OCR since it is only designed to read single characters, and it also has trouble reading characters that are not written in black marker. Both warped vowels and consonants in Hindi script have been identified.

According to [26], there are many feature extraction techniques using standard transformations, but there is still a great deal of room for extracting high-quality features from binary image segments. It has also been observed that noise filtering at the preprocessing section improves the quality of the algorithm. The existing classifier algorithms with convolutional neural networks have the highest accuracy compared to other classifiers.

IV. PROPOSED METHODOLOGY

To analyze attributes properly, we use different types of algorithms and after Comparison we come across that the Convolutional Neural Network algorithms is most suitable for making an OCR which gave result on higher accuracy [27].

Deep learning neural networks like CNN are used to interpret structured data arrays. It is very satisfactory for designs in the input image, such as lines, curves etc. which make CNN

robust in the field of Computer Vision. After applying 3 to 4 convolutional layers CNN is viable to understand the handwritten digits or characters. A convolutional neural network is built as a multi-layered feed-forward network from the combination of numerous hidden layers.

This is the first layer which extract features from input as a image. By learning features of image using small portion of the data, the convolutional layer take the relation between pixels. It take two attributes suh as matrix of an image and a kernel.

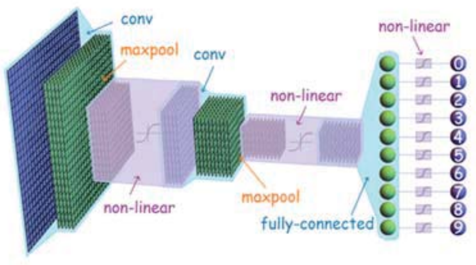


Fig. 4. Convolutional Neural Network using Keras Library

Layers make up a CNN, which filters (convolutes) the inputs to produce meaningful information. The parameters (kernel) of these convolutional layers are learned, allowing these filters to be automatically updated to extract the most relevant data for the job at hand without feature selection. CNN is stronger at handling visuals. Problems with image categorization do not lend themselves well to normal neural networks.

- Image matrix dimension – h*w*d
- Filter or Kernel dimension –$f_h * f_w * f$
- Output dimension – $(h-f_h+1) * (w-f_w+1) * 1$

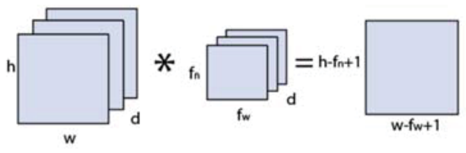


Fig. 5. Image matrix multiples of kernel matrix

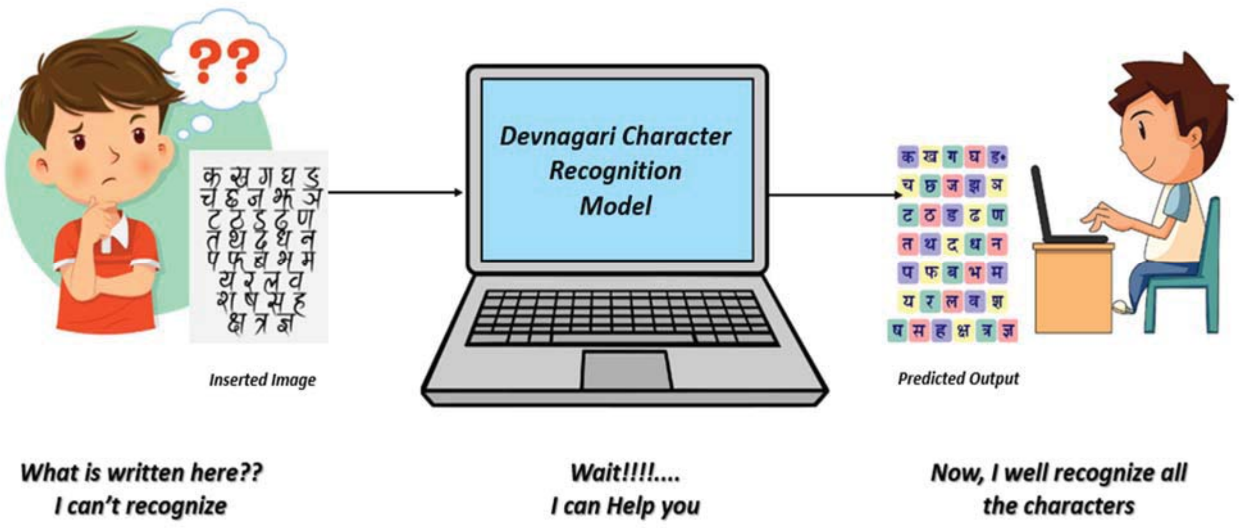


Fig. 6. System Architecture view of the Devnagari Character Recognition model

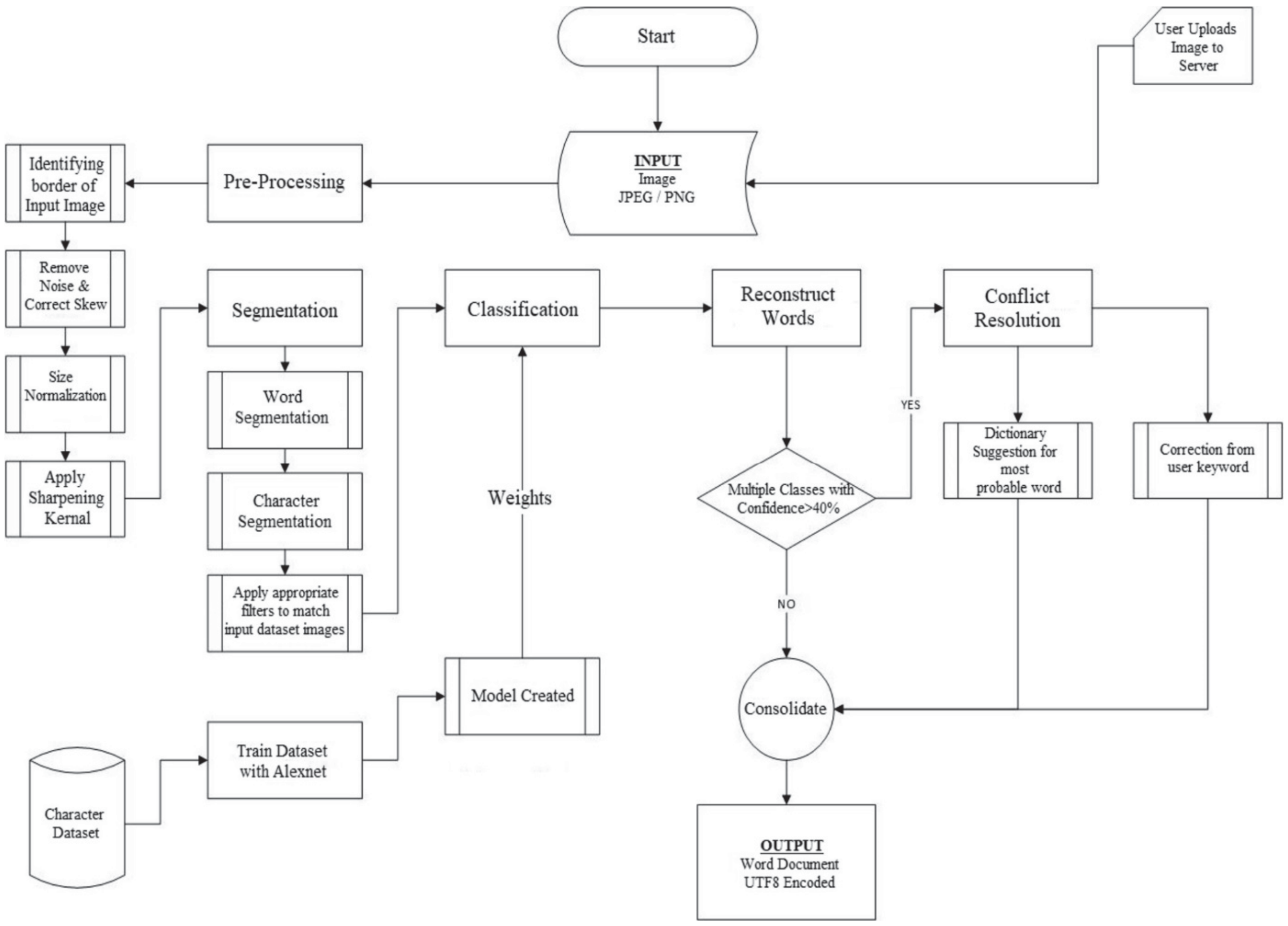

Start
User Uploads Image to Server
INPUT
Image
JPEG / PNG
Pre-Processing
Identifying border of Input Image
Remove Noise & Correct Skew
Size Normalization
Apply Sharpening Kernal
Segmentation
Word Segmentation
Character Segmentation
Apply appropriate filters to match input dataset images
Classification
Weights
Model Created
Train Dataset with Alexnet
Character Dataset
Reconstruct Words
Multiple Classes with Confidence>40%
YES
NO
Conflict Resolution
Dictionary Suggestion for most probable word
Correction from user keyword
Consolidate
OUTPUT
Word Document
UTF8 Encoded

Fig. 7. Flow chart of the Devnagari Character Recognition model

We follow several steps for getting the efficiency in our handwritten Devanagari Script Characters like-

A. *Introduction of Dataset*

We use the dataset of Devanagari Script Characters from Kaggle, which consists of 92000 images of 32 pixels and have 46 characters, 36 consonants from “ka” to “gya” and 10 digits from 0 to 9.

In this dataset we have a file named “data.csv” whose dimension 92000*1025. The columns represent all Devnagari character names corresponding to each image pixel.

This dataset originally created by “Computer Vision Research Group, Nepal.

B. *Reading Images from Folder with class name*

In this process our program read the data directly from the Kaggle website by using folder path in a class.

C. *Class Distribution*

We got the data in a balanced manner, so we do not perform image augmentation technique here.

D. *Image Characteristics*

Firstly, we get to read the images in gray form, which is only black and white and identify the type of image.

E. *Identifying Grayscale Images*

Before going to theory, we applied previously used pythonic way of checking that the image is grayscale or not. In this process basically, we checked every pixel to see whether an image is grayscale (R== G == B) or not.

An image is called grayscale image when the red, green and blue components of an image are in equal proportion.

In our dataset the values of pixels are not only present in 1 or 0 but other than that also. But, Since the red, green, and blue channels of all grayscale images have same value in pixels, it is easy to store the image in single 2-D array, so when the image is read then all the three channels merged.

We use keras library for Grayscaling the images.

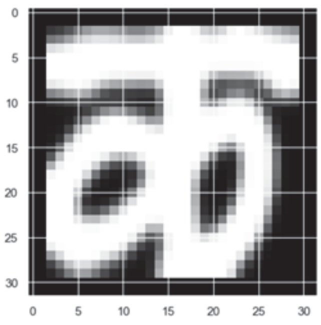


Fig. 8. Grayscale image of character “Ka”

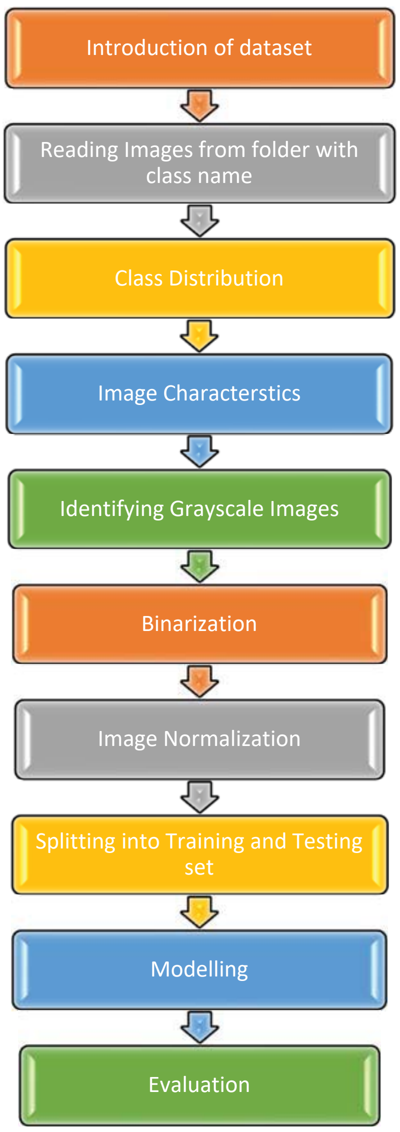


Fig. 9. Steps of OCR

F. *Binarization*

In our dataset, all the pixels are not in binary form, that is 1 or 0. Some pixels are lying between 0 and 1 so for converting all the pixels in 0 and 1 we use the technique called Binarization. In this process the colored image is converted into black and white pixels, for black the assigned value is 0 and for white the value is 255.

This is done with the help of sklearn library by using LabelBinarizer.

G. *Image Normalization*

Image Normalization is a technique that is used to change the intensity value by changing the range of pixels. In our dataset the maximum pixel is 255 so we change the value of every pixel by dividing the pixels with 255.

X (pixels) = X / 255.0

H. *Splitting into Training and Testing set*

The data had been divided into training and testing. We used 33% of the dataset for testing and 67% for training.

Train data – 1340 (each 46 classes)

Test data – 660 (each 46 classes)

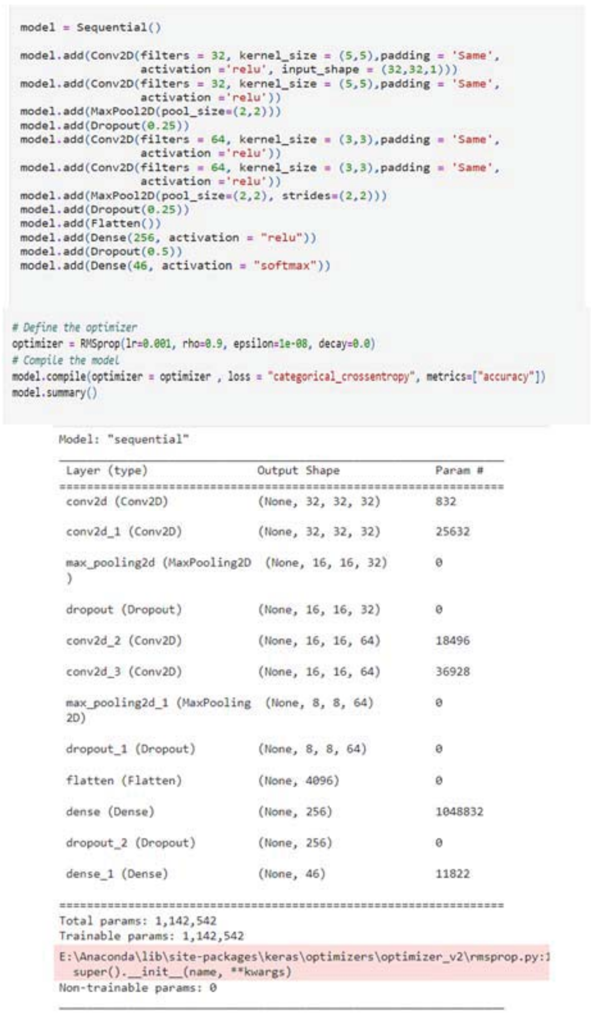
I. *Modelling*

Convolutional Neural Network (CNN) Sequential Model was utilised for modelling.

CNN's primary goal is to collect important features from the input image. It basically uses kernel for feature extraction, Pooling helps in preservation and finally flattening converted into a vector which is at last given to another network as well.

Here is how we implement CNN modelling with Model Summary.

The necessary libraries are all being imported.



```
model = Sequential()

model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',
                 activation ='relu', input_shape = (32,32,1)))
model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',
                 activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',
                 activation ='relu'))
model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',
                 activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(46, activation = "softmax"))
```

```
# Define the optimizer
optimizer = RMSprop(lr=0.001, rho=0.9, epsilon=1e-08, decay=0.0)
# Compile the model
model.compile(optimizer = optimizer , loss = "categorical_crossentropy", metrics=["accuracy"])
model.summary()
```

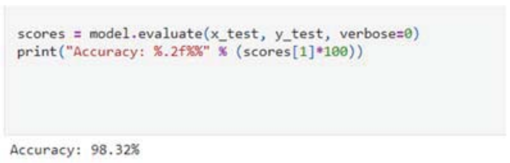
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        832

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        25632

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 dropout (Dropout)           (None, 16, 16, 32)        0

 conv2d_2 (Conv2D)           (None, 16, 16, 64)        18496

 conv2d_3 (Conv2D)           (None, 16, 16, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 dropout_1 (Dropout)         (None, 8, 8, 64)          0

 flatten (Flatten)           (None, 4096)              0

 dense (Dense)               (None, 256)               1048832

 dropout_2 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 46)                11822

=================================================================
Total params: 1,142,542
Trainable params: 1,142,542
E:\Anaconda\lib\site-packages\keras\optimizers\optimizer_v2\rmsprop.py:1
  super().__init__(name, **kwargs)
Non-trainable params: 0
_________________________________________________________________
```

With the use of the model summary, we can see that all the parameters have been programmed into the system, eliminating any potential for ambiguity when the dataset is tested.

J. *Evaluation*

For evaluation we trained our machine with the set of 86 characters by setting a learning rate annealer with the help of "ReduceLROnPlateau" function, by taking epochs =7 and batch_size of 86.

Finally, after fitting the model, we get the accuracy of 98.32%



```
scores = model.evaluate(x_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))

Accuracy: 98.32%
```

Also, we had shown the graph between the training and validation loss and accuracy. The graph below shows the fluctuation of the data points or pixel as training data and validation data. Firstly, it shows the losses which occur during data Epochs learning, as we see that the loss is less, when we do the epochs 6th time it become minimal.

The other part of graph show that the training and validation accuracy. In this part, we see that accuracy increases when we increase epochs.

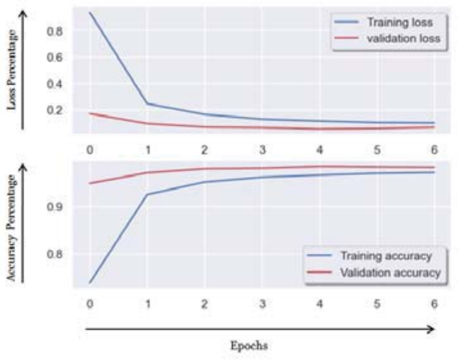


Fig. 10. Epochs VS Percentage Graph

Once we have this accuracy, we turn our CNN-sequential Model into an.h5 model and use it to build a website. By just submitting a photo to the internet, it becomes much easier for users to obtain the term. Python is used at the backend to upload the model.

V. CONCLUSION

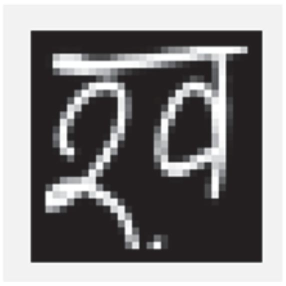
Numerous models are taken in this procedure like Lenet model, VGG-16 model, VGG model, Resnet model, efficient model, but the CNN-Sequential Model give the most accuracy on this dataset. So, we consider this model as the most suitable for making an OCR for Devnagari Script.

In future we will deploy this CNN Model to the OCR to get the most accurate output. However, this is not sufficient for typical words but worked properly when it is used for single characters.

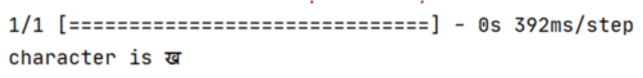
A. Working of model

Suppose we put the image of character “kha” in our model so after applying Algorithm our model gives the following result.

Inserted Image:



Predicted Output:



```
1/1 [==============================] - 0s 392ms/step
character is ख
```

REFERENCES

[1] Bansal, Veena & Sinha, R.. (2001). A Complete OCR for Printed Hindi Text in Devanagari Script.. 800-804. 10.1109/ICDAR.2001.953898.

[2] S.D. Chame and A.Kumar, "Overlapped Character Recognition: An Innovative Approach," 2016 IEEE 6th International Conference on Advanced Computing (IACC), Bhimavaram, 2016, pp. 464-469, doi: 10.1109/IACC.2016.92.

[3] Nayak, M. and Nayak, A.K. (2017) ‘Odia character recognition using backpropagation network with binary features’, Int. J. Computational Vision and Robotics, Vol. 7, No. 5, pp.588–604.

[4] Holambe, A. N., et al. "Brief review of research on Devanagari script." International Journal of Computational Intelligence Techniques 1.2 (2010): 06-09

[5] Gauri Katiyar Shabana Mehfuz, "Evolutionary Computing Techniques in Off-Line Handwritten Character Recognition: A Review ", In UACEE International Journal of Computer Science and its Applications - Volume 1 : Issue 1 [ISSN 2250 - 3765]

[6] Mr. Sunil Kumar Dasari Dr. Shilpa Mehta Ms. Diana Steffi D.D,Journal of Xi'an University of Architecture &TechnologyVolume, "Optical Character Recognition of Devanagari Script Using Machine Learning- A Survey", XII, Issue VIII, 2020 ISSN No : 1006-7930

[7] Nisha Goyal1,Er. Shilpa Jain (2015) 'Optimized Hindi Script Recognition using OCR Feature Extraction Technique',ISSN(Print) 2319 5940

[8] Divakar Yadav, Sonia Sánchez-Cuadrado and Jorge Morato, 'Optical Character Recognition for Hindi Language Using a Neural-network Approach', eISSN 2092-805X

[9] N.Arica, F.T.Y. Vural, “an overview of Character recognition focused on offline Handwriting”, IEEE Trans On System, Man, Cybernatics - Partc, VOL 31,N0.2(2001)

[10] Anamika Bhaduri1 , Deeksha Gulati2 , Sanvar Inamdar3, Mayuri Kachare “OPTICAL CHARACTER RECOGNITION USING ARTIFICIAL NEURAL NETWORK Extracting structured data from unstructured data using OCR and ANN”

[11] Ovind Trier, Anil Jain and Torfinn Taxt, “A Feature Extraction Methods for Character Recognition -A Survey”, Pattern Recognition, VOL 29, NO-4, and PP 641-662, 1996.

[12] N. Sankaran and C. V. Jawahar, "Recognition of printed Devanagari text using BLSTM Neural Network," Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), Tsukuba, 2012, pp. 322-325.

[13] U.Pal, T. Wakabayashi, F. Kimura, “Comparative Study of Devnagari Handwritten Character Recognition using different feature and classifiers”, 10th International Conference on document analysis and Recognition 2009.

[14] S. S. Marwah, S. K. Mullick and R. M. K. Sinha, “Recognition of Devanagari characters using a hierarchical binary decision tree classifier”, IEEE International Conference on Systems, Man and Cybernetics, October 1994.

[15] Hanmandalu M., and Murthy O. V. R.: Fuzzy model based recognition for handwritten Hindi numerals. International conference on Recognition, pp. 490-496, (2005).

[16] Bhopi, MsSmitaAshokrao, and Mr Manu Pratap Singh. "Performance analysis of Handwritten Devnagari Character Recognition using Feed Forward, Radial Basis, Elman Back Propagation, and Pattern Recognition Neural Network Model Using Different Feature Extraction Methods." May 18: 152-158.

[17] G. S. Lehal and Chandan Singh, ”A Gurmukhi script recognition system”, Proceedings 15th International Conference on Pattern Recognition, Barcelona, Spain, Vol 2, pp 557-560, 2000.

[18] Zhiyi Zhang, Lianwen Jin, Kai Ding, Xue Gao, “Charactersift: A Novel Feature For Offline Handwritten Chinese Character Recognition” 10th International Conference on Document Analysis And Recognition, 2009

[19] U.Pal and B.B. Chaudhuri, “An improved document skew angle estimation techniques”, Pattern Recognition” LETTERS 17:899-904, 1996.

[20] Hinduja, R. Dheebhika and T. P. Jacob, "Enhanced Character Recognition using Deep Neural Network-A Survey," 2019 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2019, pp. 0438-0440,doi:10.1109/ICCSP.2019.8698008.

[21] T.V.Aswin and P.S. Sastry, “A Font and Size-Independent OCR system for printed KANNADA Documents using SVM”, Sadhana VOL.27.PART I, PP.35-58, FEBRUARY 2002.

[22] Gaur A., and Yadav S.: Handwritten Hindi character recognition using K-means clustering and SVM. Fourth International Symposium on Emerging Trends and Technologies in Libraries and Information Services, pp. 65–70, (2015).

[23] Naresh Kumar Garg, Dr. Lakhwinder Kaur, Dr. Manish Jindal” Recognition of Offline Handwritten Hindi Text”,International Journal of Image Processing , 2013

[24] RejeanPlamondon and Sargur N. Srihari, “On-Line and Off-Line Handwritten Recognition” A Comprehensive Survey”, IEEE Pattern analysis and Machine Intelligence, VOL 22, NO. 1, January 2000.

[25] G. C. Cash and M. Hatamian, “Optical character recognition by the method of moments”, Computer Vision, Graphics and Image F’rocessing, vol. 39, pp.

[26] R.M.K.Sinha, “Rule based contextual postprocessing for Devanagari text recognition”, Pattern Recognition, 20(5), pp. 475-485, 1987. 11. I. K. Sethi, “Machine recognition of constrained handprinted Devanagari” , Pattern Recognition, vol. 18, pp. 690-706, 1996. 291-310, 1987. 9, pp. 69-75, 1977

[27] Chirag I Patel, Ripal Patel, Palak Patel, Handwritten Character Recognition using Neural Network,in International Journal of Scientific & Engineering Research Volume 2, Issue 3, March-2011 1 ISSN 2229-5518.