IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 10, May 2023

Classification of Ancient Modi Script Characters using Convolutional Neural Network

Panchfula Lohkare[1], Rutuja Pol[2], Snehal Shnde[3], Omkar Zaware[4], Prof. Vasudha Patil[5]

B.E Project Students, Department of Electronic and Telecommunication[1,2,3,4]

Assistant Professor, Department of Electronic and Telecommunication[5]

Rajiv Gandhi College of Engineering, Karjule, Harya, Maharashtra, India

Abstract*: This study aims to develop a character recognition system for the MODI language, which is challenging due to the complexity of character identification. The proposed system uses convolutional neural networks (CNN) and the VGG16 algorithm to accurately identify printed and handwritten MODI characters from scanned papers, regardless of the input paper quality. The training dataset consists of 48 distinct MODI script characters, including vowels and consonants, and is routinely updated with handwritten samples obtained from various sources and the IEEE DataPort dataset. The study demonstrates the feasibility of developing a highly accurate character recognition system that follows the established method of the MODI Script Character Recognizer System (MSCR).*

Keywords: Handwritten Character Recognition, MODI Script, Digital image, Pattern Recognition Techniques, Character segmentation Techniques;

I. INTRODUCTION

MODI script is the Marathi language's cursive writing script and historically has been used for writing only. Human Culture relies on documentation. Handwritten document analysis is needed for image and pattern recognition. Modern life is quick and mechanized. Automation and technology are connected. Everything is because humans desire to work swiftly and efficiently. Thus, automation simplifies and speeds up work. Automation is the future in today's fast-paced culture.

The Handwritten Character Recognition (HCR) technology is the subject of this discussion. HCR is becoming increasingly popular after a decade of research. Since computers can distinguish regional languages, it is necessary. This will unlock endless possibilities. Most HCR research identifies handwritten characters for computer input in English, French, Hindi, Chinese, Japanese, and others. The scanner first took the input image for an offline character recognition system. The second phase is pre-processing. It enhances the appearance and qualifies it for the category. Erosion, dilation, opening, closure, and smoothing are a few pre-processing methods. The binarization method turns a grayscale image into a binary image by using the global thresholding procedure. To create the pre-processed image appropriate for segmentation, the binarized image is dilated, and the holes are filled in the last two phases using the Sobel approach. Prior to scaling each character into m x n pixels for the training network, the input pictures are first divided into several characters.Border tracing and their Fourier Descriptors will be employed to extract the features. The character may be identified by looking at its structure and analyzing the properties of each character.

Today's "paperless workplace" requires more automated analysis capabilities. Digitalizing paper documents and files speeds up additions, searches, and edits and extends their lifespan. Thus, software that extracts, analyses, recognizes, and retrieves data from physical documents are in high demand. Document processing requires OCR text processing (OCR).

II. LITERATURE SURVEY

Sanjay S. Gharde et al.[1]demonstrate identifying and recognizing handwritten MODI characters. The ANESP programmer is utilized to create a database of handwritten examples. The MODI script manuscript has already been obtained and prepared. Affine and Moment Affine Moment Invariant and Moment Invariant are two methods for removing features from samples that have been manually separated. For identification and recognition, technology based on machine learning is utilized. SVM is one of the classification algorithms used in machine learning. This support-vector machine classifies using a linear kernel function. In the next two phases, a pre-processed picture appropriate for segmentation is created by using the Sobel approach to identify edges in the binarized image, dilation of the image, and filling of the holes. For the training network, each character is scaled into m x n pixels after the input pictures have first been split into various characters.

Manisha Deshmukh and her colleagues[2] present a method for offline recognition of handwritten Modi Numerals. The handwritten Modi numeral chain code feature extraction methodology is used to extract its properties using a non-overlapping blocking method. For Modi numeral recognition, a correlation coefficient is employed. Different non-overlapping numerical image divisions and data set sizes are used to evaluate the experimental outcomes. Through testing, a maximum identification rate of 85.21% was achieved on a database of 30000 images. The recognition results indicate that 5x5 grid divisions are more effective.

A CNN autoencoder is suggested by Solley Joseph and his coworkers [3] and used as a feature representation for text detection in the MODI script. The number of features was reduced from 3,600 to 300 using the CNN autoencoder. The obtained features were classified using SVM. More accurately than any other MODI script letter, MODI script text may be detected with a 99.3% accuracy rate. This study's primary contribution is achieving high accuracy in MODI text detection.

Tamhankar et al. [4] discuss each segmented character from ancient MODI Script manuscripts. Vertical Projection profiles (VPP) could only correctly discern characters from a line when a zero-pixel column breaks a series of letters. Based on the authors' earlier research, the paper suggests a unique method for separating each character from a line using dual thresholding criteria to minimize segmentation error. The methods used in this study's execution time analysis are essential and successful.

As part of their research, Savitri Chandure and Vandana Inamdar created a supervised Transfer Learning (TL)-based classification technique and an image dataset for MODI handwritten characters [5]. In order to retrain the network and transfer weights, Deep CNN Alexnet is used as a pre-trained network. This network extracts features from various network layers as a feature extractor. SVM is trained on the features of activation to produce classifier models. These models' precision and feature analysis are still being looked at. In-depth discriminant characteristics are chosen using subjective and objective assessments. Handwritten Devnagari character recognition was 97.25% accurate, compared to 92.32% for handwritten MODI character identification.

A chain coding and image centroid-based vowel recognition model in MODI script was proposed by Kulkarni [6]. The median filter, the global threshold, and flood fill were employed in this research to decrease noise, avoid boundary breaks, and normalize size. The data were classified using two feed-forward neural networks and SVM layers, and the recognition rate ranged from 65.3% to 73.5%.

Sidra Anam and co-workers[7] Using Otsu's Binarization technology and the Kohonen neural network approach, the Character Recognizer System for the Modi script was developed. The system was trained using 22 unique Modi script characters, including vowels and consonants, using a Kohonen neural network. Various people's handwritten samples are used to maintain the most recent version of the sample data set for the characters (training images). Users are shown how to utilise the system using these illustrations. The gathered data show that the suggested recognition approach works well. Character recognition rates are reduced by similar forms and structures. For handwritten characters, a 72.6 percent identification rate was reached.

III. PROPOSED SYSTEM

The block diagram of the proposed system is shown in Fig. 1. Block diagram is divided into training and testing.

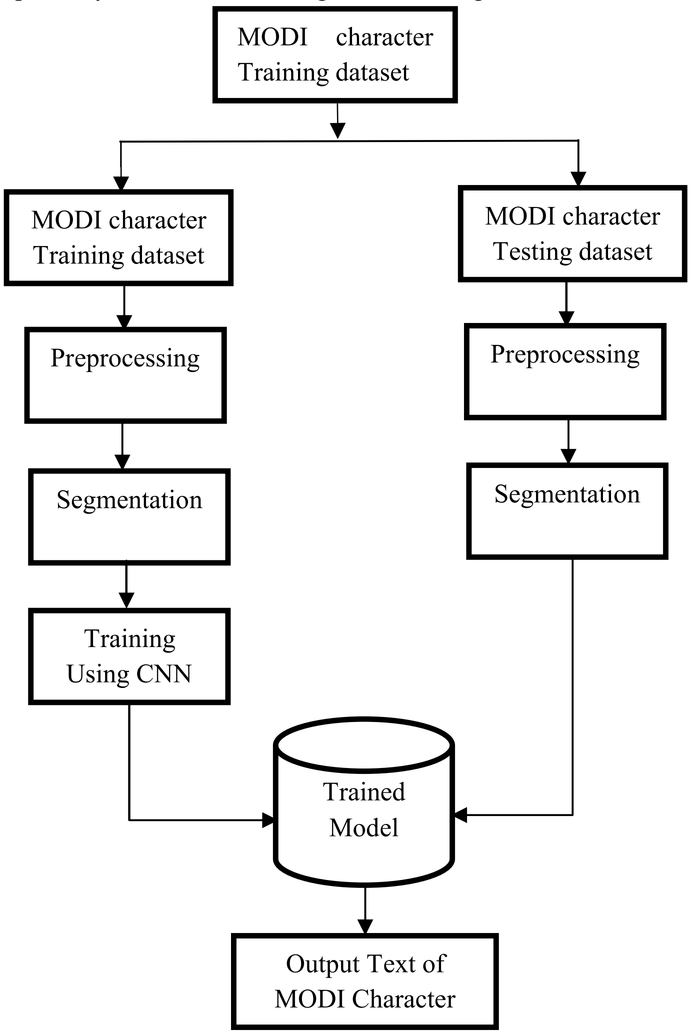

MODI character Training dataset
MODI character Training dataset
MODI character Testing dataset
Preprocessing
Preprocessing
Segmentation
Segmentation
Training Using CNN
Trained Model
Output Text of MODI Character

Fig. 1 Block diagram of MODI character recognition system

2.1 MODI character Dataset.

The proposed system uses a handwritten MODI characters dataset From the IEEE Data port. This dataset comprises 46 MODI characters, 10 vowels, and 36 consonants. Each character has 90 samples. The image size is 227x227x3 with PNG type. There is a total of 4,140character samples written by 90 participants. All 46 characters with its label are shown in Fig. 2.

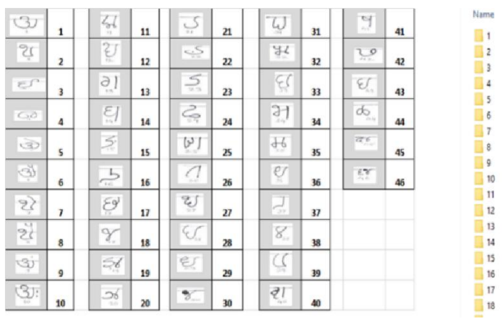


Fig.2. Basic MODI characters with their labels

IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 10, May 2023

Each character folder is created separately with its folder name as its label. So, there are 46 folders, as shown in Fig. 2. Each folder has 90 samples of respective characters, as shown in Fig.3.

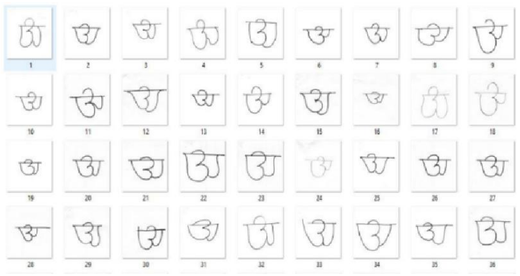


Fig.3. Characters image samples of class label 1

2.2 Data Pre-Processing

Since the taken images may be of varying sizes, each must be resized to the same size for the dataset images to be consistent. These photographs are scaled to 256x256 pixels. Various noises, including salt-and-pepper and Gaussian noise, influence the images. The median filter eliminates salt and pepper noise.

2.3 Segmentation

Thresholding is used to separate the recorded image's backdrop and character. The picture is converted to binary using the thresholding approach. The region of interest, i.e., nature, is clipped and stored for later use.

2.4 Training and testing using Convolutional Neural Networks and Vgg16

The convolutional neural network consists of Convolution, ReLu, Pooling and dense layers. Each layer is explained below.

Convolutional Layer: The convolutional network's primary element, which manages most of the computational work, is the convolutional layer. From photos, convolution layers extract features. Convolution uses small squares from the input image to learn its properties while preserving the spatial relationship between pixels. The input picture is distorted using a network of trainable neurons. The subsequent convolutional layer uses the output image as input data, and the result is a feature map or activation map. In mathematics, it is denoted as

$$G[m,n] = (f * h)[m,n] = \sum j \sum j h[j,k][m-j,n-k] \quad (1)$$

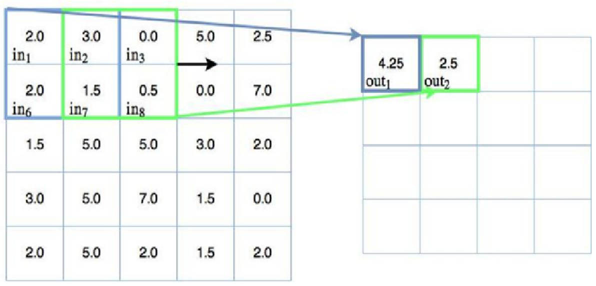


Fig.4. Moving 2x2 filter

The numbers m and n serve as the row and column indices of the output matrix, where our kernel is symbolized by h and the input picture by f. Two maps from a 2 × 2 input square are shown in Fig. 3. Each input square's weight for mapping is 0.5 for all four inputs. This convolutional step includes many features that reduce parameters and weights to facilitate training.

In contrast, in fully connected neural networks, each node in the subsequent layer is linked to every other node.Each filter has constant parameters or constant filter parameters. The filter applies the same weights to each 2x2 group of nodes as it goes around the picture. Therefore, each filter can be taught to change the input space in a particular way.

ISSN 2581-9429 IJARSCT

Each filter has a specific set of weights applied for each convolution process to limit the number of parameters.Since filters are applied over input nodes with constant weights, their weights can be learned to select input data characteristics. It could recognize lines, edges, and other geometrical elements in photos. This is how feature mapping got its name. Thus, any convolution layer needs several filters trained to recognize the distinctive characteristics. Several trained filters produce separate 2D outputs (for a 2D image). In deep learning, these many filters are frequently referred to as channels. These channels will learn critical visual cues. A convolution layer output for a grayscale image like the MNIST dataset will have three dimensions: two for each channel and a third for the number of channels.

ReLU Layer: Rectifier units are used in the non-linear process known as ReLU. Due to the element-wise nature of the process, every pixel is affected, and the feature map's negative values are all changed to zero. For the sake of this explanation, we will pretend that the rectifier has been defined and that x is the input to the neuron.

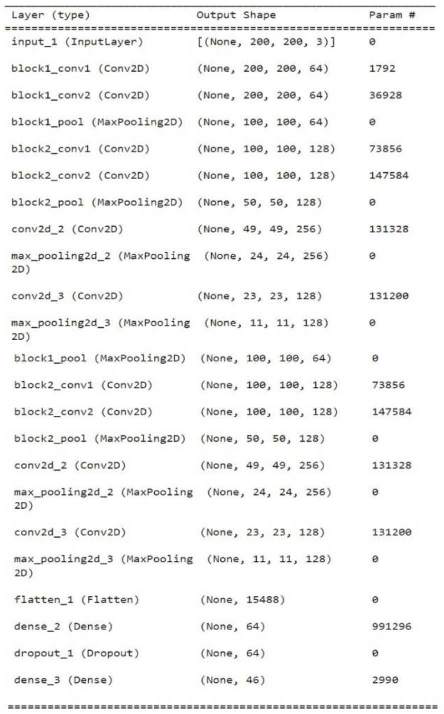
$$f(x) = \max(0, x) \qquad (2)$$

Pooling Layer:The pooling layer retains essential data while decreasing each activation map's dimensionality. It is a sliding window approach like others, but instead of using weights that may be taught, it applies some statistical function to the contents of its window. The max-pooling layers are quite straightforward and do not learn on their own. They merely output one number, the maximum in that one sliding window region.

Flattening Layer: High-resolution data is efficiently resolved into representations of objects using a convolutional neural network. Therefore, it is possible to see the fully connected layer as adding a conventional classifier to the network's information-rich output to" understand" the findings and provide a classification result. Linking this fully connected layer to the network requires flattening the convolutional neural network's dimensions output.

Fully Connected Layer: The FCL intends to use these characteristics to divide the input image into several categories depending on the training dataset. The SoftMax activation function classifier receives features from the final pooling layer, the FCL. The output probabilities of the FCL sum to 1. Using SoftMax as the activation function ensures this. In order to create a vector of summable values between zero and one, the SoftMax function compresses arbitrary real-valued scores.

The Layer information of the CNN algorithm for the proposed system is shown in Fig.5.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 200, 200, 3)]     0

 block1_conv1 (Conv2D)       (None, 200, 200, 64)      1792

 block1_conv2 (Conv2D)       (None, 200, 200, 64)      36928

 block1_pool (MaxPooling2D)  (None, 100, 100, 64)      0

 block2_conv1 (Conv2D)       (None, 100, 100, 128)     73856

 block2_conv2 (Conv2D)       (None, 100, 100, 128)     147584

 block2_pool (MaxPooling2D)  (None, 50, 50, 128)       0

 conv2d_2 (Conv2D)           (None, 49, 49, 256)       131328

 max_pooling2d_2 (MaxPooling  (None, 24, 24, 256)      0
 2D)

 conv2d_3 (Conv2D)           (None, 23, 23, 128)       131200

 max_pooling2d_3 (MaxPooling  (None, 11, 11, 128)      0
 2D)
 block1_pool (MaxPooling2D)  (None, 100, 100, 64)      0

 block2_conv1 (Conv2D)       (None, 100, 100, 128)     73856

 block2_conv2 (Conv2D)       (None, 100, 100, 128)     147584

 block2_pool (MaxPooling2D)  (None, 50, 50, 128)       0

 conv2d_2 (Conv2D)           (None, 49, 49, 256)       131328

 max_pooling2d_2 (MaxPooling  (None, 24, 24, 256)      0
 2D)

 conv2d_3 (Conv2D)           (None, 23, 23, 128)       131200

 max_pooling2d_3 (MaxPooling  (None, 11, 11, 128)      0
 2D)

 flatten_1 (Flatten)         (None, 15488)             0

 dense_2 (Dense)             (None, 64)                991296

 dropout_1 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 46)                2990

=================================================================
```

Fig.5. CNN architecture information

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 10, May 2023

VGG-16 is a convolutional neural network architecture that was proposed by researchers at the Visual Geometry Group (VGG) at the University of Oxford in 2014. It was designed for image classification tasks and achieved state-of-the-art results on the ImageNet dataset at the time of its introduction.
The "16" in the name refers to the fact that the network has 16 layers, including 13 convolutional layers and 3 fully connected layers. The convolutional layers use small 3x3 filters, which helps to capture local features of the input image. The network also uses max pooling layers to downsample the spatial dimensions of the feature maps.
One notable feature of the VGG-16 architecture is its simplicity and uniformity, which makes it easy to understand and modify. It has become a popular starting point for many researchers working in the field of computer vision and has been used as a backbone architecture for many state-of-the-art models in various domains, such as object detection, semantic segmentation, and image captioning.

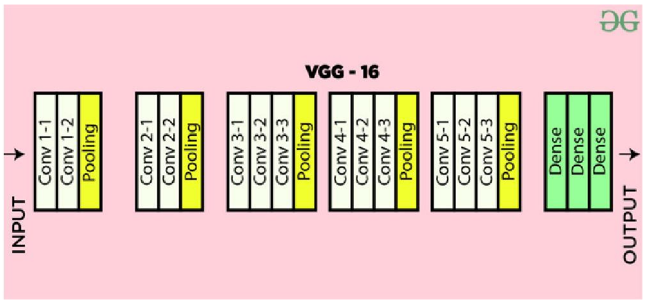
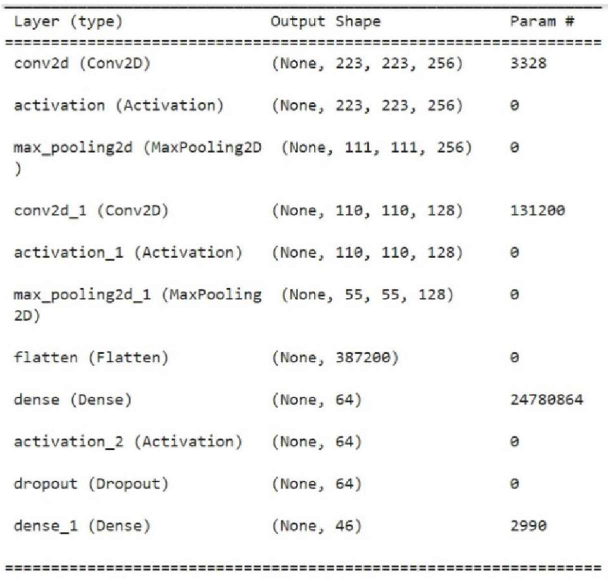


Fig.6.VGG-16 architecture map

This model was built from scratch and trained on millions of photographs from various image categories utilizing powerful GPUs. Due to its massive training data, the model has acquired a strong knowledge of low-level properties, including forms, edges, rotations, and illumination. Even though these incoming photos are from entirely different categories than the original dataset, the pre-trained model should be able to extract pertinent information from them using the transfer learning principles.
The Layer information of the Vgg16 algorithm for the proposed system is shown in Fig.7.


```
Layer (type)                Output Shape              Param #
=================================================================
conv2d (Conv2D)             (None, 223, 223, 256)     3328

activation (Activation)     (None, 223, 223, 256)     0

max_pooling2d (MaxPooling2D  (None, 111, 111, 256)    0
)

conv2d_1 (Conv2D)           (None, 110, 110, 128)     131200

activation_1 (Activation)   (None, 110, 110, 128)     0

max_pooling2d_1 (MaxPooling  (None, 55, 55, 128)      0
2D)

flatten (Flatten)           (None, 387200)            0

dense (Dense)               (None, 64)                24780864

activation_2 (Activation)   (None, 64)                0

dropout (Dropout)           (None, 64)                0

dense_1 (Dense)             (None, 46)                2990

=================================================================
```
Fig.7 Vgg16 architecture information

A classifier assigns each object to a class. This task is not optimal since samples could be in the wrong class. To evaluate a classifier, the actual class of the data must be known. To assess the classification quality, the given class by the classifier is contrasted with the actual class. It makes it possible to divide the things into the four groups indicated below:

- True Positive (TP): Positive classes correctly predicted by the classifier.

- True Negative (TN): Negative classes correctly predicted by the classifier.
- False Positive (FP): Positive classes incorrectly predicted by the classifier.
- False Negative (FN): Negative classes incorrectly predicted by the classifier.

- Accuracy

Statistical values for the classifier may now be computed based on the cardinality of these subgroups. For example, accuracy is a common and extensively used metric, but it is only valid if the different classes in the dataset are evenly distributed. The following formula determines the accuracy:

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \qquad (3)$$

It indicates the proportion of items that have been appropriately classified.

- Sensitivity

The sensitivity is determined by dividing the proportion of positive samples that were properly recognised by the total proportion of positive samples in the dataset.

$$Sensitivity = \frac{TP}{(TP+FP)} (4)$$

- Specificity

Specificity is measured as the percentage of negative samples correctly identified as negative out of the total negative samples in the dataset.

$$Specificity = \frac{TN}{(TN+FN)} (5)$$

IV RESULTS

Convolutional neural network-based MODI character recognition is presented in this research. The handwritten MODI SCRIPT character dataset is obtained in the IEEE Data PORT. This dataset has 46 MODI characters. Training (80%) and Testing (20%) datasets are used. The MODI character is trained and tested using the CNN method, and performance is assessed using the precision, recall, and F-measure parameters.

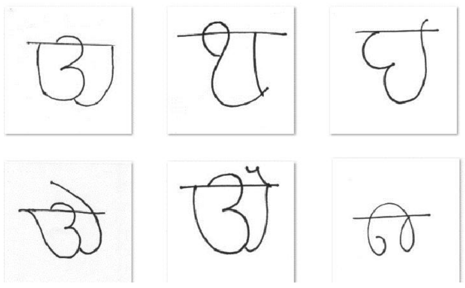


Fig.8. Input images of handwritten characters of MODI language

The character written on the sheet is segmented using thresholding techniques. The segmented characters sample is shown in Fig. 8

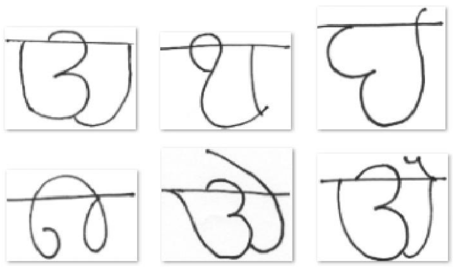


Fig.9 Samples of the segmented handwritten character of MODI language

Figs. 10 and 11 show the accuracy and loss of the proposed system utilizing CNN and vgg16.

IJARSCT

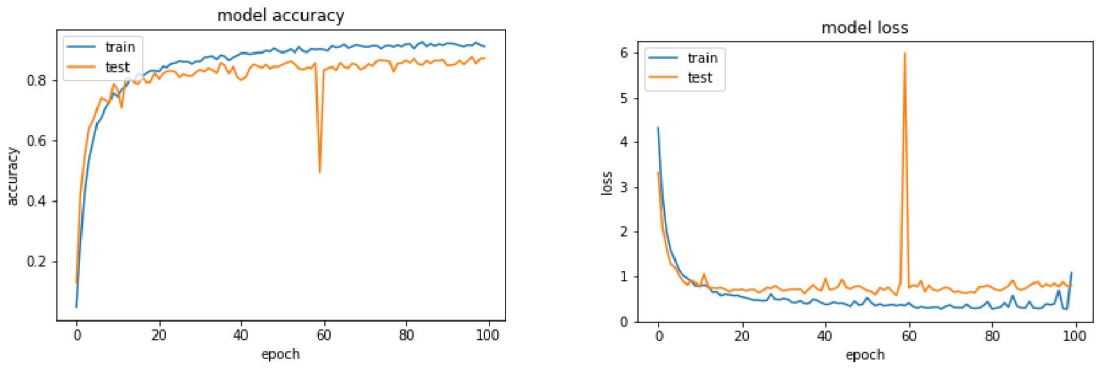
ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

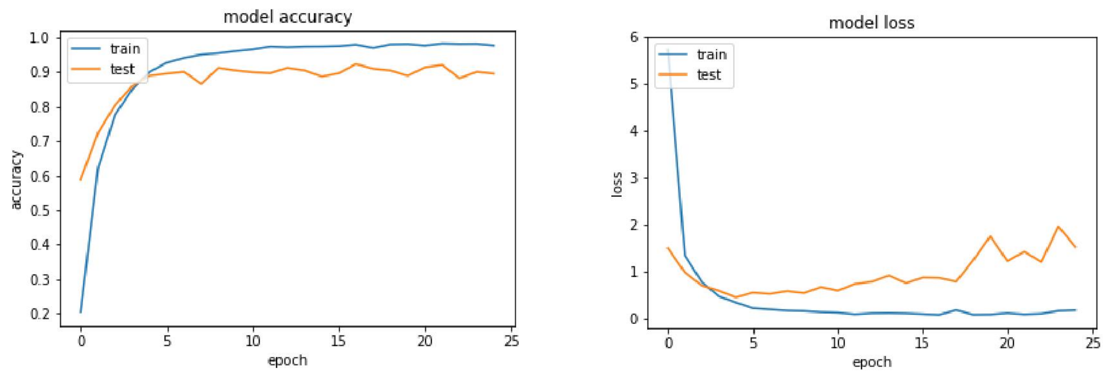
Impact Factor: 7.301

Volume 3, Issue 10, May 2023



(a)(b)

Fig.10 Training progress graph of CNN algorithm in terms of (a) Accuracy, (b) Loss



(a)(b)

Fig.11 Training progress graph of VGG-16 algorithm in terms of(a) Accuracy, (b) Loss

The accuracy and loss metrics for training and validation for the CNN and Vgg16 algorithms are compared in Table I.

AN EVALUATION OF THE TRAINING AND VALIDATION ACCURACY AND LOSS OF THE CNN AND VGG16 ALGORITHMS FOR THE IEEE DATA PORT DATASET

TABLE I
AN EVALUATION OF THE TRAINING AND VALIDATION ACCURACY AND LOSS OF THE CNN AND VGG16 ALGORITHMS FOR THE IEEE DATA PORT DATASET

Algorithm	**Training**		**Validation**		**Execution Time (Sec)**
	Accuracy	**Loss**	**Accuracy**	**Loss**	
CNN	0.9112	1.0773	0.8725	0.8074	3697.23
Vgg16	0.9762	0.1848	0.8950	1.5233	1273.99

The CNN algorithm's training process indicates a loss of 1.0773 and an accuracy of 91.12%, a Validation loss of 0.8074 and a validation accuracy of 87.25%. CNN's training period lasts 3697 seconds. The vgg16 algorithm's training process indicates a training accuracy of 97.62% with a loss of 0.1848. Validation loss of 1.5233 and validation accuracy of 89.50%. CNN's training period lasts 1274 seconds.

V. CONCLUSION

This method uses the Vgg16 and convolutional neural network algorithms to offer MODI character recognition. The performance of this system is evaluated using the IEEE data port handwritten MODI character datasets. Training (80%) and Testing (20%) datasets are used. The MODI character is trained and tested using the CNN and Vgg16 algorithms in

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 10, May 2023

this system. Accuracy and loss parameters are used to assess performance measures. For the IEEE Data Port Modi Character Dataset, the CNN algorithm's training process displays an accuracy of 91.12% and a loss of 1.0773. Validation loss of 0.8074 and validation accuracy of 87.25%. CNN's training period lasts 3697 seconds. The vgg16 algorithm's training process indicates a training accuracy of 97.62% with a loss of 0.1848. Validation loss of 1.5233 and validation accuracy of 89.50%. CNN's training period lasts 1274 seconds.
In future, there is a scope for the development of large datasets. More diverse datasets can help to improve the accuracy of Modi script character recognition systems. Modi script characters are often written in a non-standard way and may be affected by factors such as poor lighting, low contrast, and noise. Therefore, developing more advanced pre-processing techniques can help improve the recognition system's accuracy.
Modi script characters are often written in a non-standard way and may be affected by factors such as poor lighting, low contrast, and noise. Therefore, developing more advanced pre-processing techniques can help improve the recognition system's accuracy

V. ACKNOWLEDGMENT

All project students are thankful to project guide HOD & Assistant Prof. Vasudha Patil madam for the guidance during theproject work.

REFERENCES

[1]. S. S. Gharde and R. J. Ramteke, "Recognition of characters in Indian modi script," International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), p. 236–240, 2016.
[2]. M. S. Deshmukh, M. P. Patil and S. R. Kolhe, "Offline handwritten modi numerals recognition using chain code.," Third International Symposium on Women in Computing and Informatics, p. 388–393.
[3]. S. Joseph and J. George, "Handwritten character recognition of Modi script using convolutional neural network based feature extraction method and supportvector machine classifier," IEEE 5th International Conference on Signal and Image Processing (ICSIP), pp. 34-36, 2020.
[4]. P. Tamhankar, K. Masalkar and S. Kolhe, "A novel approach for character segmentation of offline handwritten Marathi documents written in Modi script.," Procedia Computer Science, 2020.
[5]. S. Chandure and V. Inamdar, "Handwritten modi character recognition using transfer learning with discriminant feature analysis.," IETE Journal of Research, pp. 1-11, 2022.
[6]. S. Kulkarni, P. Borde, R. Manza and P. Yannawar, "Recognition of handwritten modi numerals using hu and Zernike features.," 2014.
[7]. S. Anam and S. Gupta, "An approach for recognizing Modi lipi using Otsu's binarization algorithm and kohenen neural network.," International Journal of Computer Applications, pp. 29-35, 2015.