

# A DEEP FEED FORWARD NEURAL NETWORK FOR HANDWRITTEN HINDI DIGITS RECOGNITION USING CELSOFTMAX

R. Vijaya Kumar Reddy<sup>1</sup>

Research Scholar

Department of Computer Science and Engineering

Acharya Nagarjuna University, Guntur, India.

Dr. U. Ravi Babu<sup>2</sup>

Principal

DRK College of Engineering & Technology, Hyderabad, India

## **Abstract**

In this paper we present an efficient handwritten numeral digit recognition system based on a Deep Feed Forward Neural Network (DFFNN) for Handwritten Hindi Digits Recognition using Cross Entropy Logistic Softmax (CELSoftmax) technique. Handwritten digit recognition plays an important role in with the explosive growth in size of datasets of document processing and it becomes more significant to develop effective learning schemes for neural networks to deal with large scale data modeling. The proposed system has been trained on samples of 15000 images and tested on samples of 5000 images from kaggle data set and from this experiment we achieved 99.52% recognition An experimental result showed a very good recognition rate in comparison with other neural network based algorithm that use the same recognition classifier.

*Keywords: Deep Feed Forward Neural Network, Recognition, CELSoftmax classifier, HHDR.*

## **I. INTRODUCTION**

Handwritten digit recognition is a challenging task in the field of Numeral recognition because different writers have different writing styles. During the process of solving the problem of hand written recognition many challenges are faced like offline handwritten numerals may have different size, there may be a variation in their thickness, their orientation and the different kinds of noise that changes the strokes in numbers or change their structure [1]. Therefore, handwritten OCR is an area of active research.

Automatic recognition of digits has been introduced in numerous fields such as recognize zip codes on mail for postal address sorting, processing bank check amounts, numeric entries in forms filled up by hand, share certificate sorting and so on. There is a great necessity for development of the advanced techniques for recognition of numerals written in Indian scripts.

The feature extraction plays major role in both offline and online handwritten digit recognition system. Different no of feature extraction methods are stated by different users in past, like pattern matching, projection histogram, zoning, and various moment techniques [2]. But in Deep Feed-Forward Neural Networks (DFFNN) the features is done automatically to extracts features from trainable dataset which are unchanging and a certain degree to shifting and shape distortions of the initial characters. Traditional way of feature extraction is time taking processes and cannot work on initial images, but whereas the automatic feature extraction method can refurbish and rebuild features directly from initial images [2].

## II. RELATED WORK

Ahmed et.al [2] proposed a deep learning technique for recognition of Arabic handwritten digits. The proposed method Convolutional Neural Network (CNN) with LeNet-5 is trained and tested on MADBase database of Arabic handwritten digits that consists of 60000 training and 10000 testing images.

Verma [3] used radial basis function and multilayer perceptron neural networks for recognizing the handwritten characters of Devanagari script. Back propagation error algorithm is also used to improve the recognition rate. In this proposed system, they compare the results obtained from radial basis function (RBF) networks and multi-layer perceptron. Dataset consists of 245 samples written by five different users. The results so obtained show that multilayer perceptron (MLP) networks performs better than that of radial basis functions. But MLP networks training time is more as compared to radial basis function networks. Highest recognition rate so obtained using radial basis function and multilayer perceptron (MLP) networks are 85.01% and 70.8% respectively.

A. Elnagar et al. [4] proposed a method for the recognition of handwritten Hindi numerals which depends upon structural descriptors of the numeral shapes. In this method different stages are done before passing through the algorithm which segments the input images into strokes to extract features and results the syntactic representation of numeral image and then recognition is performed based on all structural descriptors obtained above and resulted the recognition rate 90 to 94%.

Banashree N. P [5] proposed a recognition scheme for handwritten Hindi numerals by using global based feature extraction approach using end-points information, which is extracted from images of isolated numerals. These feature vectors are fed to neuromemetic model that has been trained to recognize a Hindi numeral. In proposed scheme data sets are fed to neuromemetic algorithm, which identifies the rule with highest fitness value of nearly 100 % & template associates with this rule is nothing but identified numerals. The recognition rate by using this approach is 92-97 %.

Baheti M. J et al. [6] proposed a comparison of the offline handwritten character recognition system for the isolated Gujarati numerals. They used affine invariant moments based model for the feature extraction. They used KNN classifier and PCA to reduce dimensions of feature space and used Euclidean similarity measure to classify the numerals. KNN classifier obtained 90 % as recognition rate whereas PCA obtained recognition rate of 84%. After the comparison it is observed that KNN classifier has shown better results as compared to PCA classifier.

Akhilesh Pandey et al. [7] proposed a majority voting scheme for off-line hand written Hindi numbers recognitions. The main objective of this research is to find out best recognition result using multiple classifiers. The proposed technique uses simple profile and contour base triangular area representation technique for finding feature extraction and majority voting scheme on back propagation and cascade feed forward neural network for classification. The average recognition result of this approach is 94.16%.

## III. PROPOSED APPROACH

The proposed method is mainly consists of 4 phases steps. In the primary phase, collecting the numerals data from kaggle dataset and gathering images from different users . After collecting the numeral data of gray scale images will be preprocessing by checking null and missing values. Using the normalization techniques to convert the gray level values to range of 0 to 1 values, and then labelling Hindi numerals from 0 to 9. In the third phase, extracted the features atomically from DFFNN with different Hidden layer and individual digit image and derive an algorithm for recognition of handwritten numerals system in this phase. In finally phase we applied an optimization technique to get very promising results. The proposed method block diagram is shown below in Figure 1.

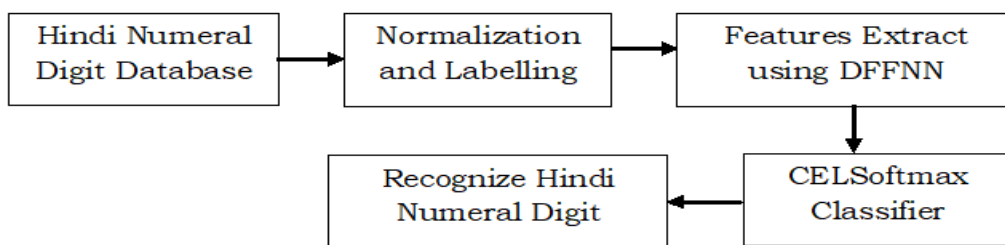


Figure 1: Block diagram proposed handwritten digit recognition system.

**Load data**

For recognizing handwritten forms, the first step is to gather data from different resource of datasets, in which a considerable amount is used for training and testing. There are total 1024 columns, each row consists an image of Hindi numeral digit. The first column represents the Hindi digit numbering from 0-9. The dataset contains 9 folders containing handwritten images in size 32\*32 pixels, each digit in the image is centre fitted to 28\*28 pixel box and padding of 2 pixels is added on all four sides of actual digit with gray-level pixels. Check for null and missing values to check for corrupted images i.e. missing values inside. There are no missing values in the train and test dataset. The Figure 2 is shown below are sample of digits from Kaggle database.



Figure 2: Illustrates some different samples extracted from the kaggle database.

**Normalization**

Normalization is a process of converting the intensity of a pixel from actual intensity to new intensity. Normalization transforms an n-dimensional grayscale image to with intensity values in the range I (Min,Max), into a new image  $I_N(\text{NewMin}, \text{NewMax})$

$$I = \{X \subseteq R^n\} \rightarrow \{\text{Min}, \text{Max}\} \tag{1}$$

$$I_N: \{X \subseteq R^n\} \rightarrow \{\text{NewMin}, \text{NewMax}\} \tag{2}$$

The linear normalization of a grayscale digital image is performed according to the formula is given below

$$I_N = a + \frac{(I - \text{Min}) * (b - a)}{(\text{Max} - \text{Min})} \tag{3}$$

Where a and b are initialized to 0.00 and 1.00

We perform a grayscale normalization to reduce the effect of illumination's differences. Moreover the DFFNN converges faster on [0..1] data than on [0..255].

## Label encoding

One hot coding is used to representation the categorical variables as binary vectors because Machine learning algorithms cannot work with categorical data directly. So, we mapped the Categorical data into integer values. Since our problem is sequence classification type problem we use one hot coding for conversion. Each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1. Next, we can create a binary vector to represent each integer value. The vector will have a length of 10 for possible 10 Hindi digit integer's values. We map each Hindi numeral to number from 0 to 9 and total of 10 integers.

{0: '0', 1: '1', 2: '2', 3: '3', 4: '4', 5: '5', 6: '6', 7: '7', 8: '8', 9: '9'}

Digit	Name	Hindi Digit	Name of Hindi Digit
0	zero	०	शून्य
1	one	१	एक
2	two	२	दो
3	three	३	तीन
4	four	४	चार
5	five	५	पांच
6	six	६	छह
7	seven	७	सात
8	eight	८	आठ
9	nine	९	नौ

Table 1: One hot coding labelling representation of numerals.

## Split training and validation set

I choosed to split the train set in two parts: a small fraction (50%) became the validation set which the model is evaluated and the rest (50%) is used to train the model.

Since we have training images of balanced labels see Load data, a random split of the train set doesn't cause some labels to be over represented in the validation set. Be careful with some unbalanced dataset a simple random split could cause inaccurate evaluation during the validation. We can get a better sense for one of these examples by visualising the image and looking at the label.

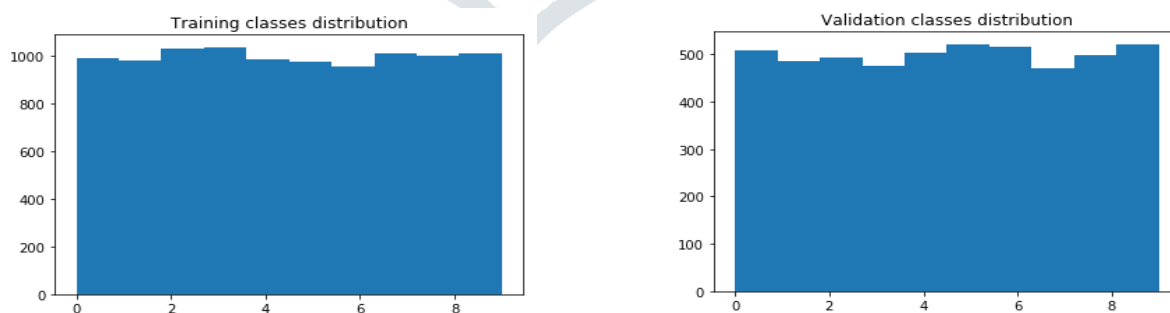


Figure 3: Data distribution of training and validation classes.



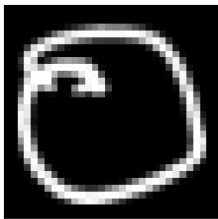



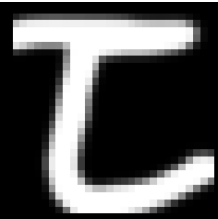
0	5	3	2	8
				

Table2: Sample data representation of labeling.

### Deep Feed-Forward Neural Network

This neural network is known as feed-forward networks, this network is simplest to analyze. The feed-forward Neural Network input layer contains  $n$ -dimensional vector as input to the network and contains  $L-1$  hidden layers as middle layers mostly two hidden layers are used and may increase based upon the requirement. Finally there is one output layer containing  $k$  number of output classes. Each neuron in the hidden layer and output layer can be split into two parts: pre-activation and activation. In Deep feed-forward network is the hidden layer are taken of size  $m$ - dimensional vector size which quite larger than the normal hidden layers of feed-forward network, so that the hidden layer can generated a huge network of features. From this huge number of features we can extract right match for predicated classes[8][9]. A deep feed-forward network is used to extract features of image automatic to recognition the classes of testing image of the data set.

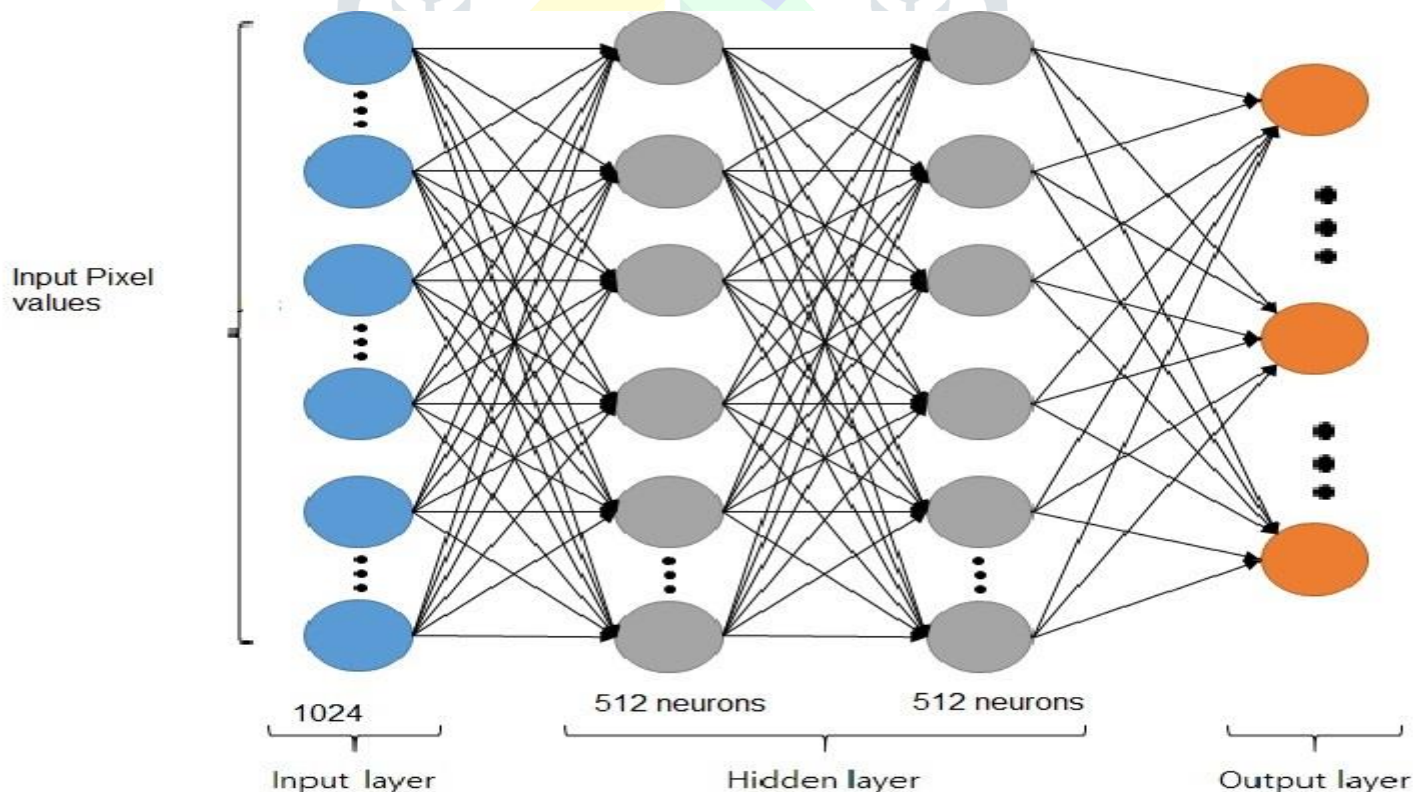


Figure 4: The overall structural design of the DFFNN Model of our proposed system with different layer.

The 'relu' is the rectifier activation function is used to add non linearity to the network [10]. The hidden layers are activated with relu activation function. The Figure 5 is shown below how the relu activation function does

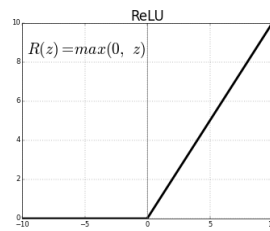


Figure 5: Activation function relu.

### Softmax classifier

Multiclass categorical probability distribution is done by Softmax activation function. Softmax activation function  $\sigma$  takes as input a n-dimensional vector  $z$  and outputs a n-dimensional vector  $y$  of real values between 0 and 1 [11]. This function is a normalized exponential and is defined as:

$$y_c = \sigma(z)_c \quad (4)$$

$$\sigma(z)_c = \frac{e^{z_c}}{\sum_{d=1}^n e^{z_d}} \quad \text{for } c=1\dots n \quad (5)$$

### Cross Entropy Loss

The cross-entropy error function over a batch of multiple samples of size  $n$  can be calculated as:

$$\epsilon(T, Y) = - \sum_{i=1}^n \sum_{c=1}^c \log(y_{ic}) * t_{ic} \quad (6)$$

Where  $t_{ic}$  is 1 if and only if sample  $i$  belongs to class  $c$ , and  $y_{ic}$  is the output probability that sample  $i$  belongs to class  $c$ .

### Algorithm

**Algorithm:** Handwritten Hindi Digits Classification

**Input:** Hindi Numeral images from test database

**Output:** Numerals Recognition

**Method:** Deep feed forward Neural Network using CELSoftmax

**Step1:** A gray level image of fixed size is taken.

**Step2:** Preprocessing the gray level image to Check for null and missing values.

**Step3:** Normalization of gray level image i.e. ranges 0 to 255 into 0 to 1 scale.

**Step4:** Mapping each Hindi digit to number from 0 to 9 and total of 10 integers.

{0: '०', 1: '१', 2: '२', 3: '३', 4: '४', 5: '५', 6: '६', 7: '७', 8: '८', 9: '९'}

**Step5:** label encoding these to one hot vector of the digits from 0 to 9.i.e for digit zero [1,0,0,0,0,0,0,0,0].

**Step6:** A deep feed-forward network is used to extract features of image automatic to recognition the target classes of image. This contains input layer of size  $n$ -dimensional vector as input,  $L-1$  hidden layers as middle layers and finally one output layer containing  $k$  number of output classes.

**Step7:** Input images are classified into suitable class label using CELSoftmax classifier.

**End**

#### IV. RESULTS AND DISCUSSION

##### Dataset description

We evaluated the performance of DFFNN using CELSoftmax classifier on a kaggle dataset. In that dataset we selected numerals of 10 classes with 2000 images for each class of handwritten isolated Hindi digits are considered. Each digit is of resolution size of  $32 \times 32$  pixels is generated from original image of actual size  $28 \times 28$  padding with 2 pixels is added on all four sides of actual character [12]. Some sample set of images used for testing in our experiment from the database is shown below in Figure 6. The data set was split into a training set and a test set where randomly selected 15000 images for the training set and 5000 images randomly for testing.

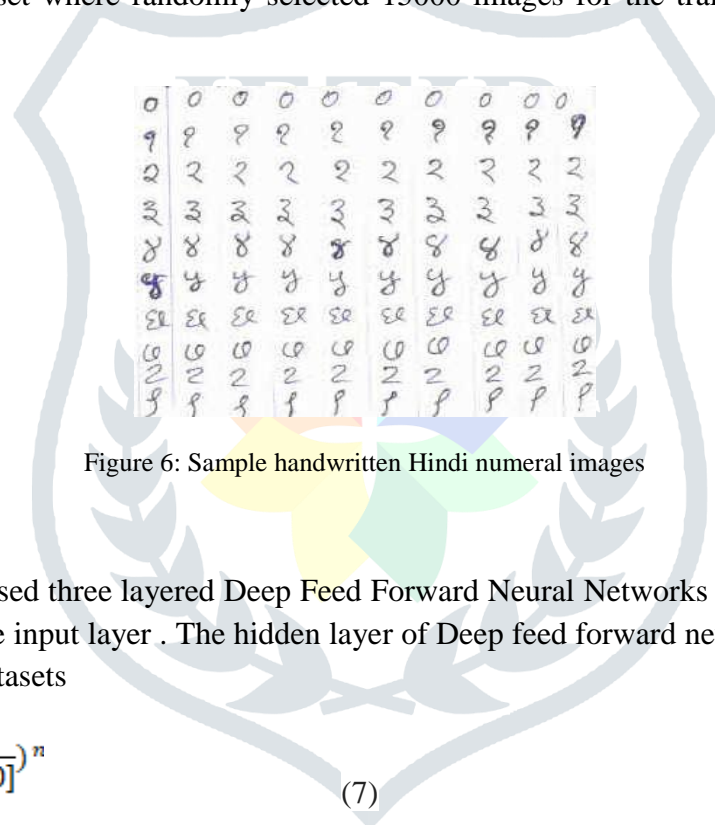


Figure 6: Sample handwritten Hindi numeral images

##### DFFNN Architecture

In this testing phase we used three layered Deep Feed Forward Neural Networks (DFFNN) model. On them two hidden layers and one input layer. The hidden layer of Deep feed forward neural networks with random weights for large scale datasets

$$WSD = \left( \frac{2.0}{Weight\_Shape[0]} \right)^n \quad (7)$$

The pre-activation at layer  $i$  is given by

$$a_i(x) = W_i * h_{i-1}(x) + b_i \quad (8)$$

$W_i$  and  $b_i$  are weight and bias between layers

$h_{i-1}$  is output from each neuron from Hidden layer

The activation at layer  $i$  is given by

$$h_i(x) = g(a_i(x)) \quad (9)$$

Where  $g$  is called the activation function mostly relu activation function is used.

The activation at the output layer is given by

$$h_L(x) = O(a_L(x)) \tag{10}$$

Where O is the output activation function i.e. CELSoftmax classifier is used.

The validation accuracy is greater than the training accuracy almost every time during the training. That means that our model does not overfit the training set. The image below shows an example of training and validation accuracy and learning rate are low, high and very high and overfitting or not. Here we can see that our DFFNN performs very well on all.

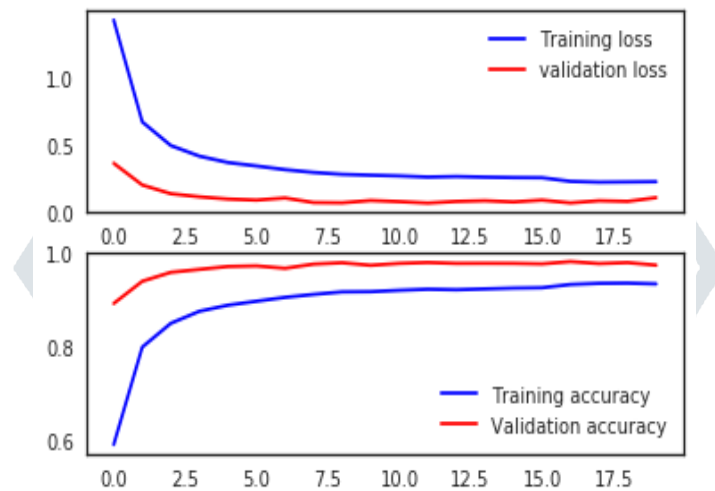


Figure 7: Training and Validation Loss and Accuracy

By using the algorithm defined in section 3, the digits are classified. The proposed algorithm is tested on 5000 digits and the individual outcome of the test database is given in Table 3.

Digit	No. of images	Correctly Classified	Not Correctly Classified	% Accuracy
0	513	511	2	99.61
1	486	479	7	98.56
2	494	488	6	98.79
3	487	480	7	98.56
4	502	494	8	98.41
5	523	513	10	98.09
6	515	504	11	97.86
7	479	471	8	98.33
8	491	483	8	98.37
9	510	503	7	98.63
Average recognition percentage				99.52

Table 3: Individual results of the test database.

Different approaches	%Accuracy
Kohenen Neural Network Classifier	93.82
Structural Descriptors	94



Neuro-Memetic Model	95.1
LDA Classifier	92.97
Majority Voting Scheme	94.16
DFFNN with CELSoftmax Classifier	99.52

Table 9: Accuracy of HHND recognition of Different approaches

Comparison graph of the proposed approach with other existing approaches

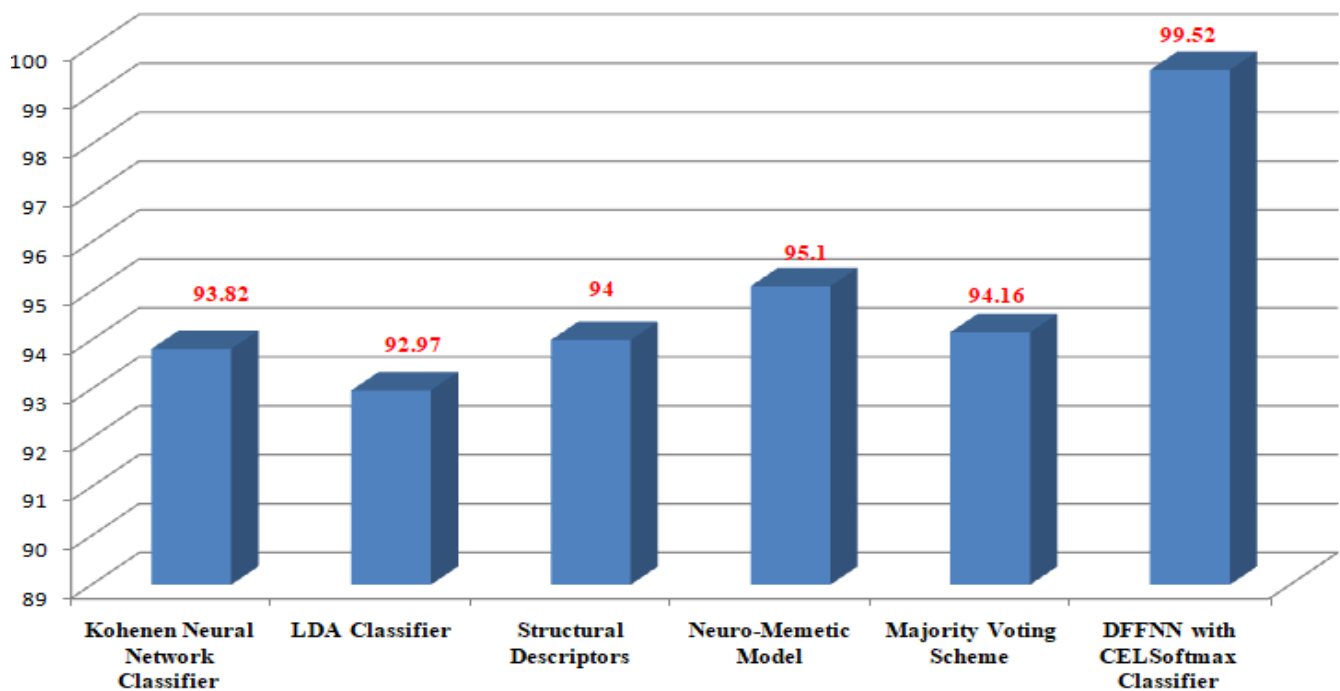


Figure8: Graphical representation of the % recognition of the proposed method and other existing method

### V. CONCLUSION

In this paper a neural network approach for Handwritten Hindi Numeral Digit Recognition (HHNDR) is been proposed. We evaluated the performance using Deep Feed Forward Neural Network using CELSoftmax Classifier is used on a standard a kaggle dataset. In that dataset we selected numerals of 10 classes with 2000 images for each class of Handwritten Isolated Hindi Digits are considered. Our training data contains total 15000 training samples and 5000 Testing data contains different characters. From experimental results, it is observed that DFFNN using CELSoftmax Classifier yields the best accuracy for HHNDR compared to the alternative techniques. We achieved 99.52% recognition rate using the proposed method with high yield of accuracy.

### REFERENCES

- [1] Amandeep Kaur, Er. Manish Mittal, "Survey Paper on Hindi Digit Recognition", International Journal of Computer Science and Technology. Vol. 7, Issue 2, April - June 2016, pp: 113-117.
- [2] Ahmed El-Sawy, Hazem EL-Bakry , "CNN for Handwritten Arabic Digits Recognition Based on LeNet-5" Advances in Intelligent Systems and Computing, Springer International Publishing AG 2017.
- [3] B. K. Verma, "Handwritten Hindi character recognition using multilayer perceptron and radial basis function neural networks," in Neural Networks, 1995. Proceedings, IEEE International Conference on. vol. 4, pp. 2111-2115, 1995.
- [4] A. Elnagar, F.Al-Kharousi, S. Harous,"Recognition of Handwritten Hindi Numerals using Structural Descriptors", 1997 IEEE.
- [5] Banashree N. P., R. Vasanta,"OCR for Script Identification of Hindi (Devnagari) Numerals using Feature Sub Selection by Means of End-Point with Neuro-Memetic Model", World Academy of Science, Engineering and Technology, 4, 2007.
- [6] BAHETI M. J., KALE K.V., JADHAV M.E., "Comparison of Classifiers For Gujarati Numeral Recognition", International Journal of Machine Intelligence, Vol. 3, Issue 3, pp. 160-163, 2011.
- [7] Akhilesh Pandey, Amresh Kumar, Rajiv Kumar, Amod Tiwari,"Handwritten Devanagari Number Recognition using Majority Voting Scheme", International Journal of Computer Science and Information Technology & Security (IJCSITS), Vol. 2, No. 3, June 2012.
- [8] Hailiang Ye, Feilong Cao, Dianhui Wang, Hong Li , "Building feed forward neural networks with random weights for large scale datasets", Expert Systems with Applications Volume 106, 2018, pp. 233-243
- [9] Feilong Cao, Dianhui Wang, Houying Zhu, Yuguang Wang, "An iterative learning algorithm for feedforward neural networks with random weights" Information Sciences, Volume 328, 2016, pp. 546-557
- [10] Andres Calderon, Sergio Roa , Jorge Victorino , "Handwritten Digit Recognition using Convolutional Neural Networks and Gabor filters", Proceedings of the International Congress on Computational Intelligence CIIC 2003.
- [11] Weiyang Liu, YandongWen, Zhiding Yu, Meng Yang," Large-Margin Softmax Loss for Convolutional Neural Networks ", Proceedings of the 33 rd International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48
- [12] B.V.Dhandra, R.G.Benne, Mallikarjun Hangarge, "Kannada, Telugu and Devanagari Handwritten Numeral Recognition with Probabilistic Neural Network: A Script Independent Approach", International Journal of Computer Applications (0975 – 8887) Volume26–No.9,July 2011