Pakistan Journal of Engineering and Technology, PakJET
Multidisciplinary | Peer Reviewed | Open Access
Volume: 2, Number: 1, Pages: 1- 8, Year: 2019

Recognition of Off-Line Hand-Written Alphabets Using Knowledge-Based Computational Intelligence

Muhammad Haseeb [1] and Ghulam Abbas [*,1]

[1] Electrical Engineering Department, The University of Lahore, Lahore 54000, Pakistan

[*] Corresponding author: Ghulam Abbas (e-mail: ghulam.abbas@ee.uol.edu.pk)

***Abstract*—The handwritten character recognition is considered an active recognition problem under the field of image processing and pattern recognition. Both areas of research have gained so much popularity in few last years due to technology advancements and necessity of time. The handwritten character recognition also becomes popular due to globalization and technology improvements. In previous time, the information was stored in written form only. To utilize that information, one needs such an advanced method which can do this task efficiently and without reliance on human. This work provides such a simplified and accurate method for the recognition of handwritten characters. A system is proposed to recognize characters using convolutional neural network (CNN) and is evaluated on a benchmark dataset named as EMNIST to show the performance of the proposed technique. The proposed system gains the accuracy of 97.62% which is better than various existing methods.**

***Index Terms*—Convolutional neural network, handwritten character recognition, EMNIST dataset.**

I. Introduction

Automatic recognition of characters is an important area and significant development in this area is already performed, although there is much scope to carry the research further. Successful models are deployed to recognize printed text for English and many other languages where the problem needs much attention for some languages. Handwritten character recognition is a similar area where the characters are not consistent and are written in cursive manner. There are two type of approaches to solve the problem of handwritten character recognition: one is online handwritten character recognition in which the writing strokes are noted and text is recognized; the other one is offline mode which is more complex as it involves the scanning of the handwritten text and then recognition of this text. Handwritten character recognition has many applications especially in digitization of historical documents and records, postal address and other such information. The task of offline handwritten character recognition has some complexities such as localization and recognition of handwritten text, and segmentation of characters in continuous text. Recognition of text can be important in helping visually impaired persons [1]. Such a system can be of help in provision of addition information to an autonomous navigation system. Moreover, text in natural images can provide rich information about the underlying scenes or images.

The current research focuses on unsupervised feature learning. There are few feature-learning algorithms which extract low-level representations of underlying data [2], [3] and provide an alternative to hand-engineered features. These types of algorithms have achieved success in some visual recognition tasks [4]. In text recognition task, the model in [5] has achieved good performance in text localization and character recognition. The model relies on scalable and simple feature-learning architecture which does not depend on hand-crafted features or prior knowledge. The task of feature learning and character recognition is performed using convolutional neural network (CNN). CNNs are layered neural networks which have good representational capacity and are applied to various problems such as object recognition [6], handwritten text recognition [7], and character recognition [8].

The proposed method is experimented using scanned images which are not noise-free and the neural network is used for recognition of character. The characters which are used for recognition are English alphabets. Surf Feature Extraction method is used to perform the complete recognition along with Neural Network. The proposed algorithm provides usable results in terms of PSNR and MSE (which are used for evaluation of the technique) as compared to existing methodologies for the task of character recognition. The overall work is based on character recognition with reduced noise and enhancement of image quality with different levels of noise and eventually makes the technique suitable for both

image processing and pattern recognizing tasks.

II. Neural Networks

A. Neural Networks' Background

Neural networks try to solve the problem with a different approach inspired from human visual cortex. The neural network is formed with several neurons, cascaded in progressive layers with multiple interconnections. These neural networks are provided with a large number of handwritten characters with labels, referred as training set, and automatically form rules by adjusting the weights of neural connections for recognition of handwritten characters. The recognition accuracy of the neural network increases by increasing the number of training examples.

B. Neural Networks' Architecture

Neural networks have different parts and each of them can be described with their specific name. The leftmost layer of neural network shown in Fig. 1 is named input layer and the neurons in this layer are called input neurons. All the inputs to the neural network are provided through input neurons only. The rightmost layer of the neural network is called the output layer and neurons contained in this layer are called output neurons. The neural network contains only one output neuron but it could be more than one, depending on the number of output states. The middle layers of the neural networks are called hidden layers because these neurons are neither accessible at input nor at output. The term hidden layer sounds mysterious but it is called so only because it is not visible to input or the output. Neural network shown in Fig. 1 has two hidden layers but it can have single or multiple layers depending on complexity.

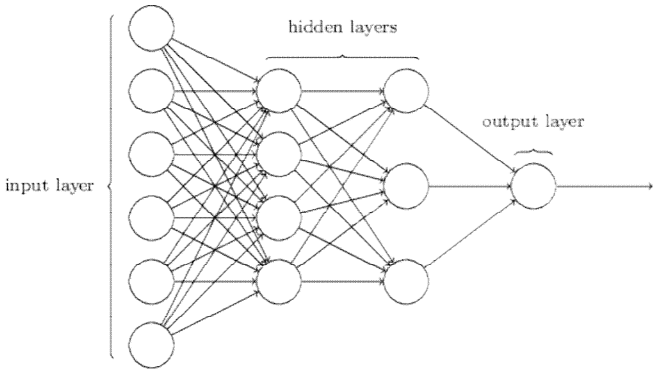

Fig. 1. Architecture of neural network with two hidden layers.

The number of input and output neurons depends on the input variables and the output states respectively. For example, if we want to determine whether the character is "*C*" or not. If the input image size is 32 by 32 pixels with grayscale intensities, there would be a total of 32×32=1024 neurons in the input layers encoding the grayscale intensities from zero to one. The output layer will contain only one neuron with values greater than or less than 0.5, indicating "*C*" if the value is greater than 0.5 and "not a *C*" in case the value is less than 0.5.

III. Convolutional Neural Networks

The feed forward neural networks have input and output layers along with single or multiple hidden layers. All the neurons in each subsequent layer are fully connected to each neuron in next layer as shown in Fig. 2. The problem with such network is that fully connected layers do not make much sense in character recognition. The reason is that they do not take into account the spatial structure of pixels, i.e. the pixels which are adjacent and far away from one another have similar value. The concept of spatial structure is needed to be incorporated in such a way that instead of starting from a series of input neurons, we have a matrix of input neurons. This concept results in the development of convolutional neural networks (CNNs). CNNs use a special architectural design which has particular adaptability for image classification. The use of this special architectural design makes them faster to train. This strategy is useful in training of many-layered networks (deep) which are especially useful at image classification.

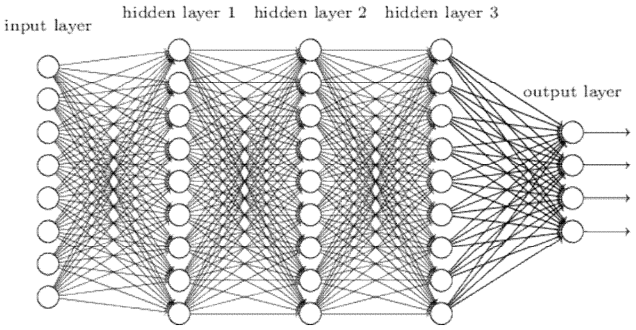

Fig. 2. Fully-connected feed forward neural network.

CNNs are based on three basic ideas: pooling, shared weights and local receptive fields.

A. Local Receptive Fields

To be more exact, every neuron in the main shrouded layer associates with a little locale of the information neurons, say, for instance, a 5×5 area, relating to 25×25 entered pixels. In this way, for a specific shrouded neuron, we may have associations that resemble as shown in Fig. 3.

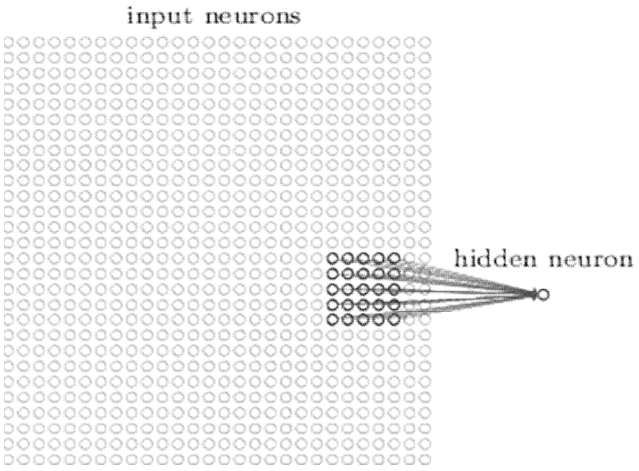

Fig. 3. Local receptive field of 5×5.

That region of the input image is known as the local receptive field for the hidden neuron. It is a little window on the input pixels. Every association learns a weight. Furthermore, the hidden neuron also calculates a bias. We, at that point, slide the local receptive field over the whole input image. For every local receptive field, there is an alternate hidden neuron in the primary hidden layer. The local receptive field is slided to right by one neuron (i.e. one pixel) and it is connected to another neuron in the hidden layer. Then we slide the local receptive field over by one pixel to the right, to connect to a second hidden neuron (see Fig. 4).

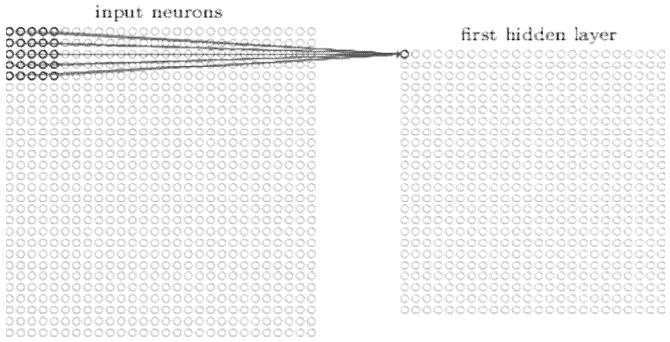

Fig. 4. Connection of local receptive field with hidden neurons.

Continuation of the same operation yields first hidden layer. When we have a 28×28 neurons in the input image, and length of local receptive field five, there will be 24×24 neurons in the first hidden layer. This is on the grounds that we can just move the local receptive field crosswise over or downward, before crashing into the right-hand side or base of the input image. The movement of local receptive field in this image is 1 pixel but it could be increased as per the requirement. This is called stride length and researchers have experimented with different stride lengths to observe the behavior of deep convolution CNN.

B. Shared Weights and Biases

In this demonstration, the local receptive field has a size of 5×5 weights connected to hidden neurons. It is worth mentioning that the weight and biases are shared and same weights and biases are used for all 24×24 neurons in the first hidden layer. Mathematically, for all i and j_{th} hidden neurons, the output is:

$$\sigma\left(b+\sum_{i=0}^{4}\sum_{j=0}^{4}w_{l,m}\alpha_{i+l,j+m}\right) \tag{1}$$

In (1), σ is the sigmoid activation function; b is the bias value shared with other pixels; w is the array of shared weights; α denotes the input activation at a specified position represented by i, j.

This phenomenon of shared weights and biases means that first hidden layer detects exactly the same feature. The notion of the feature is to extract hidden layer from local receptive field. This learned feature may be an edge or corner in the image which could occur anywhere in the image and can be used as a useful feature. CNNs are adjusted well in respect of translation invariance. The map which gets translated from input image to hidden layer is sometimes called as feature map. The feature map can be more than one in most of the cases and each feature learns a specific image feature. Fig. 5 shows 20 different feature maps learned during first hidden layer. Each small image indicates a 5×5 weight of local receptive layer.

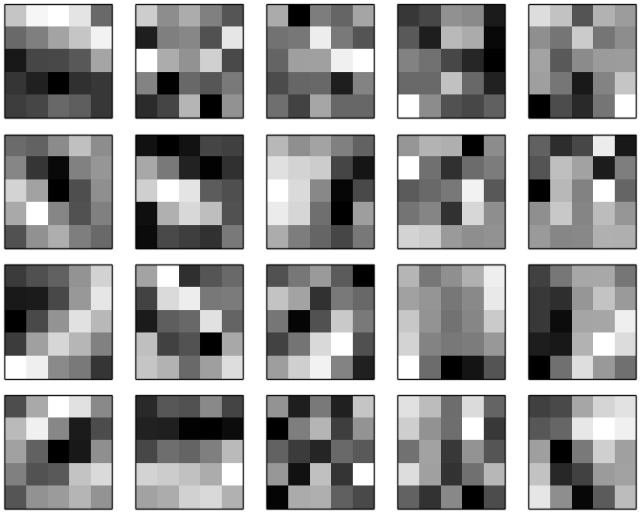

Fig. 5. 20 feature maps of 5×5 learned by convolution net.

C. Pooling Layers

Convolution neural network's basic unit is convolution layer described previously. Pooling layer is also an integral layer which is used just after the convolutional layer. This layer simplifies the information from the previous convolutional layer. A pooling layer simplifies the output from the convolutional layer. For example, a unit of pooling layer can take a 2×2 block and condense the information into a single neuron. Some common pooling operation may be max-pooling or average-pooling in which the output takes the maximum or average of 2×2 input block, same as demonstrated in Fig. 6.

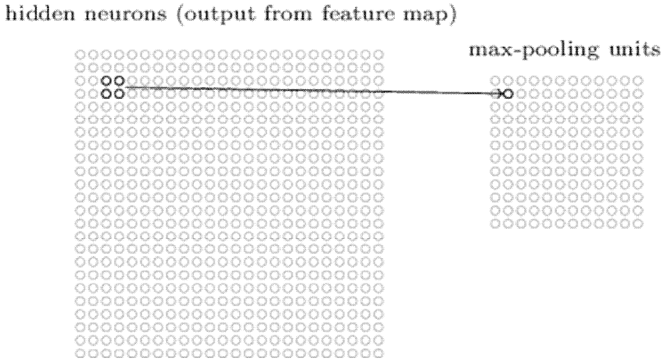

Fig. 6. Max-pooling operation on 2×2 block.

Pooling layer results in further reduction in size of convolution layer's output. For a 24×24 layer produced from the convolutional layer, the pooling layer of size 2×2 produces an output of 12×12 neurons as shown in Fig. 7.

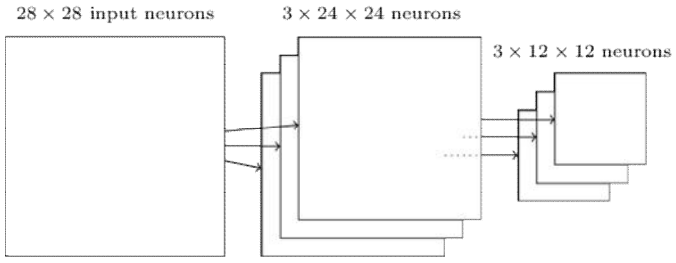

Fig. 7. Structure of cascaded convolution neural network's layer after pooling operation.

Putting all the configurations in final form results into a convolution neural network of Fig. 8. The output of pooling layer is connected to sigmoid neurons placed in vertical fashion which may be less than the total neurons in the previous layer. This layer is a fully-connected layer, i.e. every neuron is connected with all the neurons in the sigmoid layer. The final (output) layer is a softmax layer which provides a normalized output for output. The number of neurons in the output layer depends on the total number of classes, and the neuron of softmax layer with highest normalized value describes the correct class of the image.

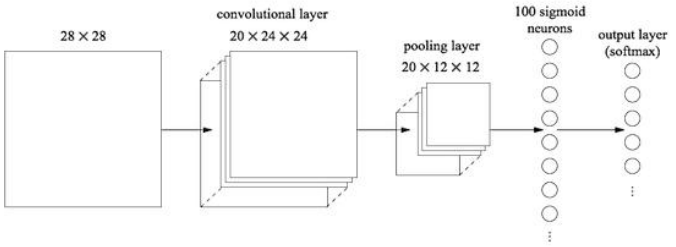

Fig. 8. Simplified structure of a convolution neural network for handwritten characters recognition.

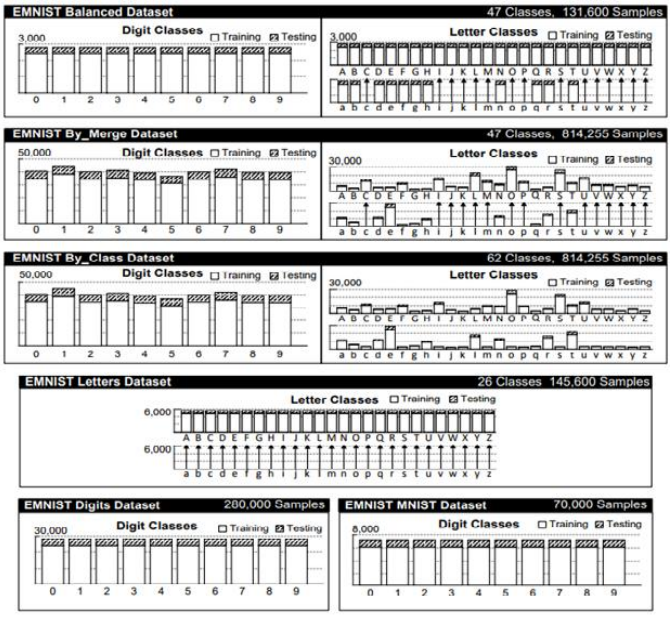

Fig. 9. Visual representation of the EMNIST dataset.

IV. Methodology

A. Handwritten Character Recognition

Despite the inherent differences between the text detection and character recognition tasks, structurally identical network architectures for both the text detector and character classifier are used. State-of-the-art performance is to be achieved on the EMNIST dataset.

B. Dataset

For the task of handwritten character recognition large number of handwritten character images are required and the plenty of datasets are available online to use. To evaluate the performance of the proposed work, the EMNIST dataset [9], which is the derived form of NIST special database 19, is used. This dataset contains handwritten character images in form of six different splits as shown in Fig. 9.

C. Hand Character Recognition Steps

Various steps involved in recognizing the hand-written characters are depicted in Fig. 10.

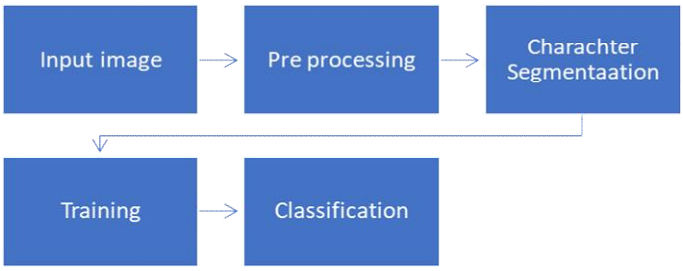

Fig. 10. Hand character recognition steps.

1) Preprocessing

The first task for hand character recognition is preprocessing of the input image. First, an image input is given to the system and then skew correction of the image is performed to get the correct alignment. After converting RGB image into grayscale, we have used Otsu algorithm for black and white image conversion. Conversion of image is performed using Otsu algorithm [10]. The next task is to remove the noise from black & white image and for that first image is inverted and then 3×3 median filter is applied. The morphological operations have their own importance in recognition task and in this scenario, we have used 60 pixels as a threshold and images less than 60 pixels are removed. Further, we need clear stroke of image and that is done by applying morphological thinning filter.

2) Character Segmentation

The next task is to segment the single character as a single image and for that first connected regions are selected ranging from 200-30000 pixels. Then character extraction is achieved by applying horizontal thresholding and a single character is stored as an image. As EMNIST dataset has images of 28×28 pixels, the input image is also resized by 28×28 after applying centralization of image. All above steps are applied during testing phase and the system learns from data.

3) Training

When dealing with neural networks, training is an important phase as the whole performance of the system relies on how your system is trained. In short, the performance is as better as your training of the data. For the present scenario, the EMNIST dataset is already trained but we need to explain the process for testing of system as whenever a new input comes, how it will learn from the system.

a) Architecture of Neural Network

Neural network basically has input, hidden and output layer and in the present study we have used a seven-layered neural network in which:

- First layer is of 28×28 pixel input layer or of 784 neurons (28×28=784).
- The second layer is 2-dimensional convolutional layer with local receptive field of 5×5 pixels and extracts 20 features, and rectified linear unit is used as an activation function.
- The next layer is 2-dimensional max-pooling layer of 2×2 with stride length of 2 pixels.
- The fourth layer is the fully connected layer of 62 elements neurons.
- Then there is the softmax layer which helps into mapping of the input to the real or correct value.
- The sixth layer is classification layer and classifies the input as a correct character.
- The last is the output layer which shows the recognized character.

b) Training Parameters

Stochastic Gradient Descent: The stochastic gradient descent is basically gradient vector and the purpose of using this is learning of neural network as it provides suitable weights and biases which are helpful in minimizing the cost of network. It works by recurrently calculating the gradient ΔC which is further passed in opposite direction, falling down, as shown in Fig. 11 [11]. Training options used for our work are shown in Fig. 12.

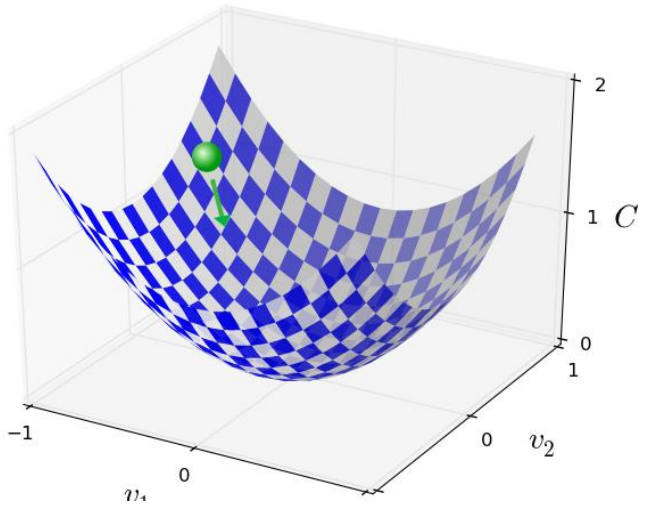

Fig. 11. Slope of stochastic gradient descent.

```
options =

  TrainingOptionsSGDM with properties:

                     Momentum: 0.9000
                 InitialLearnRate: 1.0000e-04
        LearnRateScheduleSettings: [1×1 struct]
                 L2Regularization: 1.0000e-04
          GradientThresholdMethod: 'l2norm'
                GradientThreshold: Inf
                        MaxEpochs: 20
                    MiniBatchSize: 128
                          Verbose: 1
                  VerboseFrequency: 50
                    ValidationData: []
               ValidationFrequency: 50
                ValidationPatience: 5
                          Shuffle: 'once'
                   CheckpointPath: ''
             ExecutionEnvironment: 'auto'
                       WorkerLoad: []
                        OutputFcn: []
                            Plots: 'none'
                  SequenceLength: 'longest'
            SequencePaddingValue: 0
```

Fig. 12. Training options.

Epochs: In neural network while training of data, we need to update the weights and for that purpose we need a parameter to define the number of iterations of training vector before updating a weight value. The maximum number of epochs used is twenty (20). Progression of epochs is shown in Fig. 13.

```
Training on single CPU.
Initializing image normalization.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:00 |       2.34% |       5.4756 |    1.0000e-04 |
|       1 |          50 |       00:00:02 |      70.31% |       1.0910 |    1.0000e-04 |
|       1 |         100 |       00:00:05 |      76.56% |       0.9263 |    1.0000e-04 |
|       1 |         150 |       00:00:07 |      76.56% |       0.8804 |    1.0000e-04 |
|       1 |         200 |       00:00:10 |      79.69% |       0.6904 |    1.0000e-04 |
|       1 |         250 |       00:00:13 |      85.94% |       0.4960 |    1.0000e-04 |
|       1 |         300 |       00:00:15 |      81.25% |       0.5008 |    1.0000e-04 |
|       1 |         350 |       00:00:18 |      82.81% |       0.5121 |    1.0000e-04 |
|       1 |         400 |       00:00:21 |      89.84% |       0.3411 |    1.0000e-04 |
|       1 |         450 |       00:00:24 |      89.06% |       0.2709 |    1.0000e-04 |
|       1 |         500 |       00:00:26 |      92.97% |       0.2707 |    1.0000e-04 |
|       1 |         550 |       00:00:29 |      94.53% |       0.2032 |    1.0000e-04 |
|       1 |         600 |       00:00:32 |      95.31% |       0.1603 |    1.0000e-04 |
|       1 |         650 |       00:00:35 |      88.28% |       0.4730 |    1.0000e-04 |
|       1 |         700 |       00:00:38 |      85.16% |       0.4002 |    1.0000e-04 |
|       1 |         750 |       00:00:41 |      91.41% |       0.3464 |    1.0000e-04 |
|       1 |         800 |       00:00:44 |      87.50% |       0.3873 |    1.0000e-04 |
|       1 |         850 |       00:00:47 |      87.50% |       0.4149 |    1.0000e-04 |
|       1 |         900 |       00:00:49 |      91.41% |       0.2744 |    1.0000e-04 |
```

Fig. 13. Training of data.

Learning Rate: The learning rate of network is kept 0.0001.

4) Classification

After the training of the data, next is to use it for system evaluation which means to correctly classify the input image with what handwritten character it is. The 80% of the data is used for training and the 20% is used for the testing task. It means that the 20% data is used to classify and, the collected classification measure is the achieved accuracy of our handwritten character recognition system.

V. Results and Discussion

After explaining the complete methodology used for handwritten character recognition, we are in a state to explain the results achieved through the process and also the

evaluation measures used. The proposed system uses convolutional neural network for character recognition task. By performing all the essential steps, our system correctly identifies the characters as shown in Figs. 14 to 21.

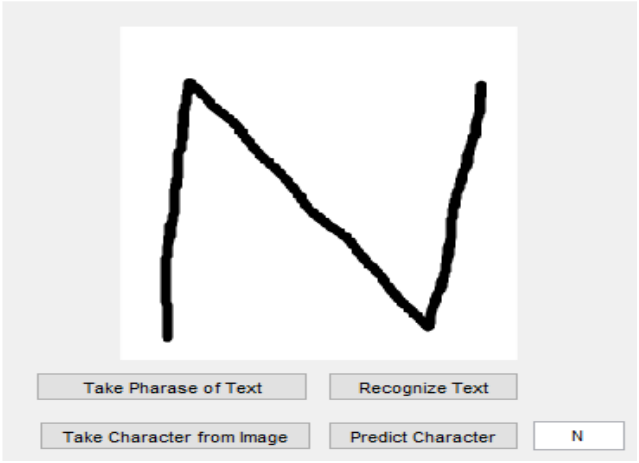

Fig. 14. Single character recognition (1).

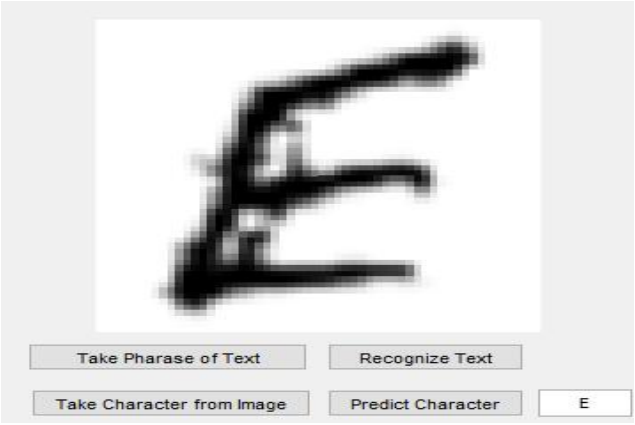

Fig. 15. Single character recognition (2).

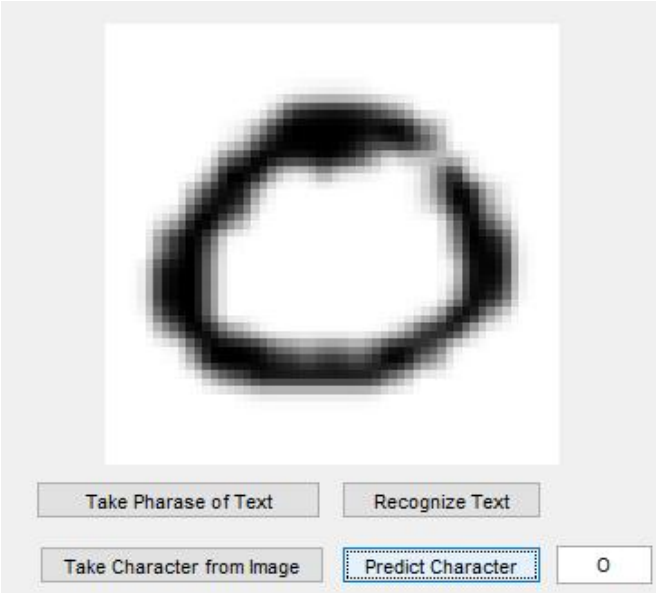

Fig. 16. Single character recognition (3).

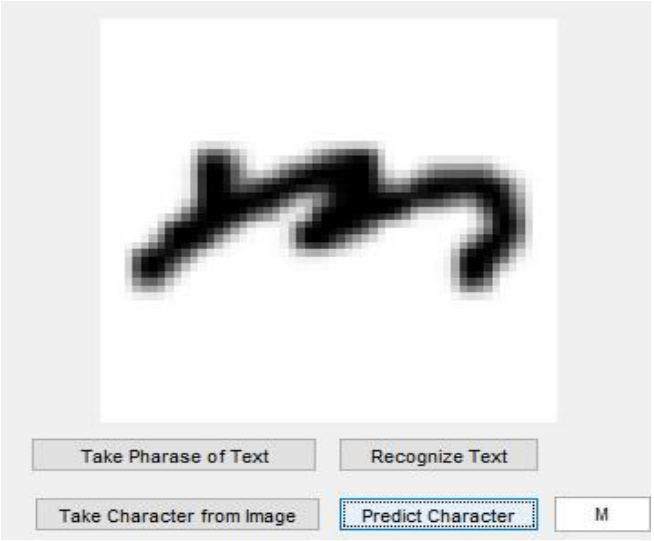

Fig. 17. Single character recognition (4).

Fig. 18. Single character recognition (5).

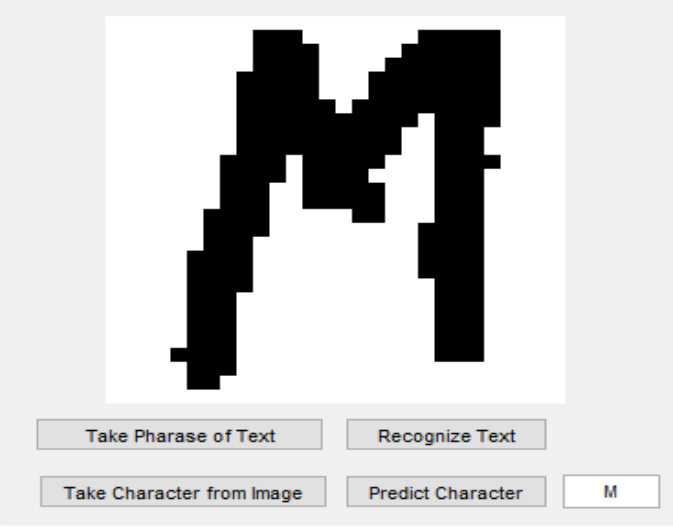

Fig. 19. Single character recognition (6).

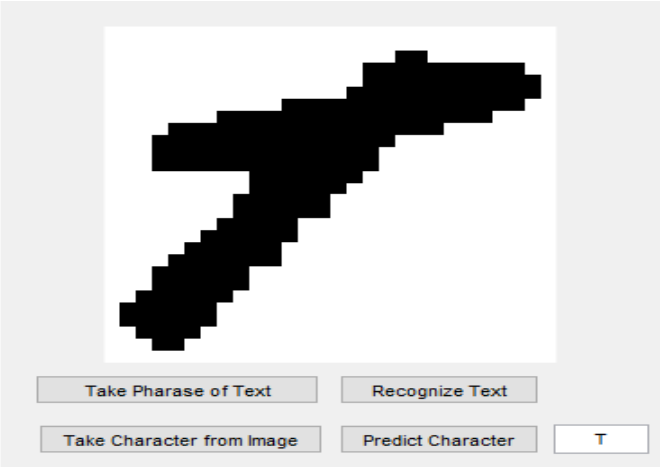

Fig. 20. Single character recognition (7).

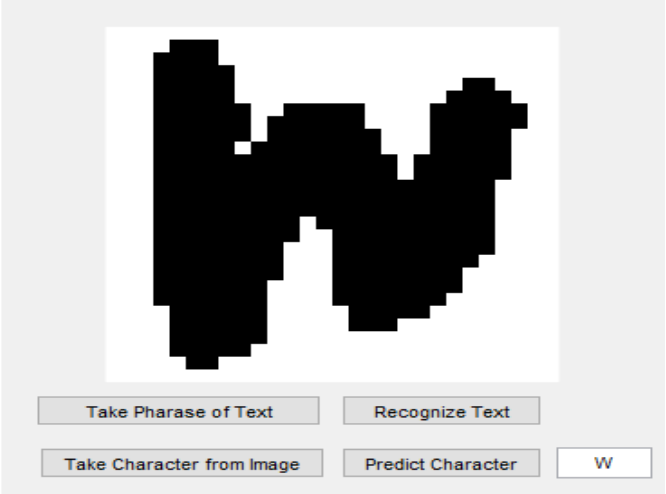

Fig. 21. Single character recognition (8).

Not only the single characters are identified through the system but also the phrase is also correctly recognized as shown in Figs. 22 to 24. The main difference between single character and phrase recognition is that for single character recognition we need the character segmentation but in case of phrase recognition we have to skip segmentation step.

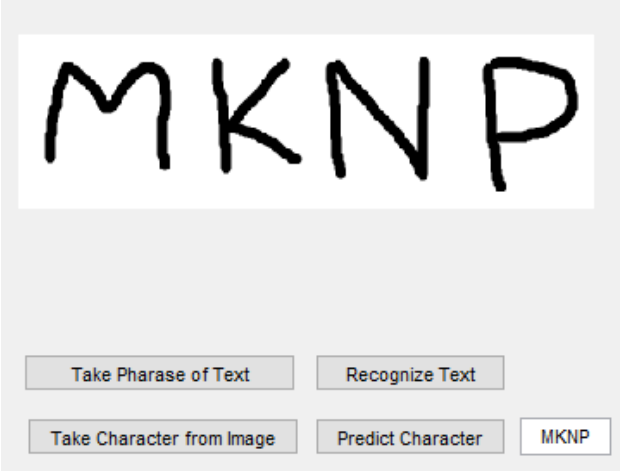

Fig. 22. Phrase recognition (1).

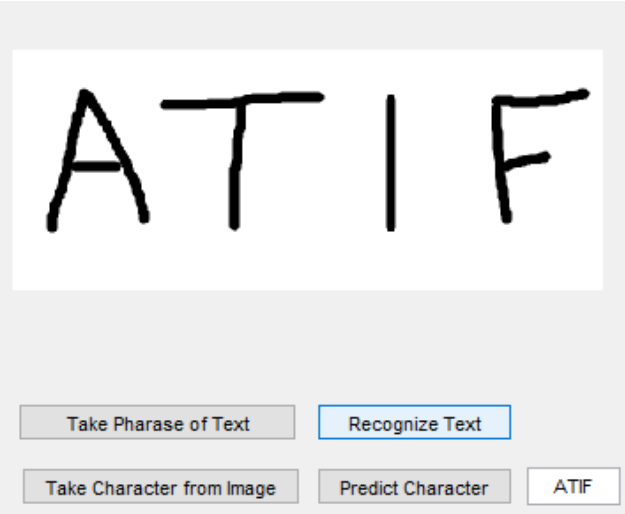

Fig. 23. Phrase recognition (2).

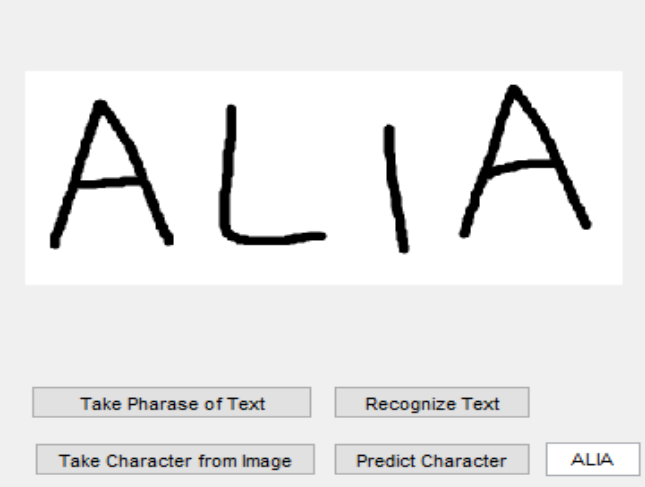

Fig. 24. Phrase recognition (3).

This study is mainly carried out for handwritten character recognition problem using convolutional neural network, which means to train your computer to perform the task automatically instead of human dependence. The convolutional neural networks are famous these days due to their outstanding performance on almost every kind of problem. For the present case, we have used a seven layered network which takes input of 28×28 pixel image. The recognition problem depends on the type of characters you are dealing with. Say for example if we only have to recognize single digit then we can simply use ten output neurons which is required to match the digit from 0 to 9. Furthermore, task is slightly different when the recognition subject is different but the main functionality of the system remains same throughout the process. In training parameters, we have used stochastic gradient descent method during the learning of neural network which is commonly used among researchers for learning of neural network. The mini batch size during training is 128.

To evaluate the performance of our system we need to define some kind of evaluation measure and the %Accuracy is the widely used measure to compare or analyze the system

performance. The accuracy can be calculated as:

$$\%Accuracy = \frac{Correctly\ recognized\ samples}{Total\ number\ of\ samples\ in\ dataset} \times 100 \qquad (2)$$

The proposed system is tested using EMNIST dataset and on the given dataset our system has achieved best accuracy of 97.62% which means our system performs significantly better and also outperforms various major methods.

VI. CONCLUSIONS

The task of recognition is so far an active research area due to its high demand and dependence of human writing scripts. Even if every type of structural, topological and statistical measures of characters are obtained, even then still the work of recognition is left due to a lot of variation in languages, their writing styles and also the writing mood of a person. Hand character recognition system is the capability of a system to take input of a written image to recognize it automatically. The input is normally used in the form of a scanned image of required character. The hand-written characters are recognized successfully in this work using CNN. This network uses different layer for the task completion and the purpose of these layers is to reduce the processing. The performance of the system on the dataset was as good as of humans in a lot of cases which have made these networks a real use of the time. But also, it is tough to recognize small characters which are written in smaller fronts because the parameters for such characters and their recognition need other careful implementation. The recognition of handwritten character is done using its scanned image of 28×28 pixels. Our proposed system works on seven-layered convolutional neural network, and characters are initially separated through using segmentation technique. Through the proposed system, we can recognize individual characters as well as phrases. The performance of the system is evaluated using accuracy measure on EMNIST dataset which is a most popular dataset for training and testing of machine learning applications. The overall proposed system after training and testing has proved its worth. For the recorded observations, the system achieves accuracy of 97.62% which makes the proposed method usable for hand character recognition tasks.

REFERENCES

[1] R. Ani, E. Maria, J. J. Joyce, V. Sakkaravarthy and M. A. Raja, "Smart Specs: Voice assisted text reading system for visually impaired persons using TTS method," *2017 International Conference on Innovations in Green Energy and Healthcare Technologies (IGEHT)*, Coimbatore, 2017, pp. 1-6.

[2] J. Pradeep, E. Srinivasan and S. Himavathi, "Diagonal based feature extraction for handwritten character recognition system using neural network," *2011 3rd International Conference on Electronics Computer Technology*, Kanyakumari, 2011, pp. 364-368.

[3] R. Verma and R. Kaur, "An Efficient Technique for character recognition using Neural Network & Surf Feature Extraction," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 2, pp. 1995-1997, 2014.

[4] M. I. Jubair and P. Banik "An Approach to Extract Features from Document Image for Character Recognition," *Global Journal of Computer Science and Technology*, 2013.

[5] M. A. Rahiman and M. S. Rajasree, "Printed Malayalam Character Recognition Using Back-propagation Neural Networks," *2009 IEEE International Advance Computing Conference*, Patiala, 2009, pp. 197-201.

[6] V. P. Agnihotri, "Offline Handwritten Devanagari Script Recognition," *Information Technology and Computer Science*, vol. 8, pp. 37-42, 2012.

[7] R. Alaasam, B. Kurar, M. Kassis and J. El-Sana, "Experiment study on utilizing convolutional neural networks to recognize historical Arabic handwritten text," *2017 1st International Workshop on Arabic Script Analysis and Recognition (ASAR)*, Nancy, 2017, pp. 124-128.

[8] C. Wu, W. Fan, Y. He, J. Sun and S. Naoi, "Handwritten Character Recognition by Alternately Trained Relaxation Convolutional Neural Network," *2014 14th International Conference on Frontiers in Handwriting Recognition*, Heraklion, 2014, pp. 291-296.

[9] G. Cohen et al, "EMNIST: an extension of MNIST to handwritten letters," *arXiv preprint arXiv:1702.05373*, 2017.

[10] M. Yongli and H. Zhikai, "The study of segmentation of rubbings image based on OTSU&histogram differential algorithm," *2017 International Conference on Computer, Information and Telecommunication Systems (CITS)*, Dalian, 2017, pp. 171-175.

[11] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," *Advances in neural information processing systems*, 2013.