NOVEL FRAMEWORK FOR HANDWRITTEN DIGIT RECOGNITION THROUGH NEURAL NETWORKS

Manjot Kaur
Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM). New Delhi (India)
E–mail: manjot.kaur97@gmail.com

Tanya Garg
Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM). New Delhi (India)
E–mail: tanya.pathak@gmail.com

Ritika Wason
Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM). New Delhi (India)
E–mail: rit_2282@yahoo.co.in

Vishal Jain
Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM). New Delhi (India)
E–mail: drvishaljain83@yahoo.com

Recepción: 05/03/2019 **Aceptación:** 09/04/2019 **Publicación:** 17/05/2019

Citación sugerida:
Kaur, M., Garg, T., Wason, R. y Jain, V. (2019). Novel framework for handwritten digit recognition through neural networks. *3C Tecnología. Glosas de innovación aplicadas a la pyme. Edición Especial, Mayo 2019*, pp. 448–467. doi: http://dx.doi.org/10.17993/3ctecno.2019.specialissue2.448–467

Suggested citation:
Kaur, M., Garg, T., Wason, R. & Jain, V. (2019). Novel framework for handwritten digit recognition through neural networks. *3C Tecnología. Glosas de innovación aplicadas a la pyme. Special Issue, May 2019*, pp. 448–467. doi: http://dx.doi.org/10.17993/3ctecno.2019.specialissue2.448–467

ABSTRACT

The biggest challenge for natural language processing systems is to accurately identify and classify the hand–written characters. Accurate Handwritten Character recognition is a challenging task for humans too as the style, size and other handwriting parameters may vary from human to human. Though a relatively straightforward machine vision task but improved accuracy as compared to the existing implementations is still desirable. This manuscript aims to propose a novel neural network based framework for handwritten character recognition.

The proposed neural network based framework, transforms the raw data set to a NumPy array to achieve image flattening and feeds the same into a pixel vector before feeding it into the network. In the neural network, the activation function is applied to transfer the resultant value to the hidden layer where it is further minimized through the use of minimized mean square and back propagation algorithms before applying a stochastic gradient on the resultant mini–batches.

After a detailed study, the optimal algorithm for effective handwritten character recognition was proposed. Initially, the framework has been simulated only on digits through 50,000 training data samples, 10,000 validation data set and 10,000 test data set, the accuracy of 96.08.

This manuscript aims to give the reader an insight into how the proposed neural network based framework has been applied for handwritten digit recognition. It highlights the successful applications of the same while laying down the directions for the enhancements possible.

KEYWORDS

Natural Language Processing, Handwritten Character Recognition, Neural Networks, Machine Vision; Digit Recognition.

1. INTRODUCTION

Many literate humans effortlessly recognize the decimal digit set (0–9). A sample of the same is depicted in Figure 1 below. This natural attribute of mankind is actually due to the seemingly simple human brain. The human brain is a supercomputer in itself as each of its hemispheres has a visual cortex containing 140 million neurons with approximately tens of billions of connections between them (LeCun, *et al.*, 1990). So this supercomputer tuned by evolution over hundreds of millions of years on this earth is superbly adapted to understand this complex, colourful visual world.

This masterpiece–human brain can solve a tough problem like recognizing any entity in this world in moments of seconds. The difficulty bubbles up when we attempt to automate the same task by writing a computer program and applying computer vision for character/digit recognition (Bottou, *et al.*, 1994) (Nielsen, 2018). Recognizing digits, in particular, is a simple task as the input is simple black and white pixels with only 10 well–defined outputs. However, the accurate recognition of the handwritten shapes in different styles, fonts etc is a complex task in itself. A simple 6 has a loop on bottom and vertical or curved stroke on top which can be written in varied styles and is difficult to express algorithmically to a machine which is none other than a newborn baby born after every turn off (Bottou, *et al.*, 1994; Nielsen, 2018; Hegen, Demuth, & Beale, 1996). Neural networks solve the above problem in a much simpler way (LeCun, *et al.*, 1990; Bottou, *et al.*, 1994; Nielsen, 2018; Hegen, *et al.*, 1996; Widrow, Rumelhart, & Lehr, 1994; Mishra & Singh, 2016). The strategy is to take huge data set of black and white handwritten digits from real life people and build a neural network to train from those data sets and learn to recognize those digits (Shamim, Miah, Sarker, Rana, & Jobair, 2018; Patel, Patel, & Patel, 2011; Ganea, Brezovan, & Ganea, 2005).

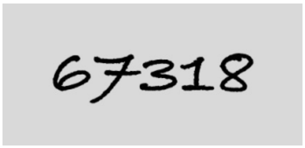

Figure 1. Sample of Handwritten Digits figure.

In Neural Networks, each node performs some simple computation on the input and conveys a signal to next node through a connection having a weight and a bias associated with it which amplifies or diminishes a signal (Nielsen, 2018; Shamim, Miah, Sarker, Rana, & Jobair, 2018). Different choices for weights and bias results in different functions evaluated by networks. An appropriate learning algorithm must be used to determine the optimal values of weights and bias (Widrow, *et al.*, 1994; Knerr, Personnaz, & Dreyfus, 1992).

All neural networks have these following common following attributes (Widrow, *et al.*, 1994; Alwzwazy, Albe–Hadili, Alwan & Islam, 2016; Matan, *et al.*, 1990).

- A set of processing units
- A set of connections
- A computing procedure
- A training procedure

The Processing Units

The processing units in a Neural Network are the smallest units just like neurons in a brain. These nodes work in a similar fashion and operate simultaneously. There is no master procedure to coordinate them all (Cardoso & Wichert, 2013). These units compute a scalar function of its input and broadcast the results to units connected to it as output. The result is called the activation value and the scalar function is called activation function (Widrow, *et al.*, 1994).

There are three types of inputs:

- Input unit, which receives data from the environment as input.
- The hidden unit, which transforms internal data of network and broadcast to the next set of units.

DOI: http://dx.doi.org/10.17993/3ctecno.2019.specialissue2.448-467

- Output unit, which represents a decision as the output of all system (Widrow, *et al.*, 1994).

The Connections

The connections are essential to determine the topology of a neural network. There are three types of topologies (Gattal, Djeddi, Chibani & Siddiqi, 2016):

- Unstructured networks for pattern completion.
- Layered networks for pattern association.
- Modular networks for building complex systems.

The topology used in this paper for the proposed system is layered networks (Widrow, *et al.*, 1994).

A Computing Procedure

Computations feed input vectors to processing units from the input layer (Sakshica & Gupta, 2015). Then the activation value of remaining units is computed synchronously or asynchronously. In a layered network, this is done by feedforward propagation method. The activation functions used are mathematical functions. The most common function is a sigmoidal function (Mishra & Singh, 2016).

A Training Procedure

Training a network implies adapting its connections according to the input environment so the network can exhibit optimized computational behaviour for all input patterns (Arel, Rose & Karnowski, 2010). The process used in this paper is modifying weights and biases with respect to the desired output. The cost of error is calculated using a mean square error method (Mishra & Singh, 2016).

2. DATA COLLECTION

Handwritten digit data set is vague in nature because they may not always be sharp straight lines of pixels (LeCun, *et al.*, 1990). The main goal in digit recognition of feature extraction is to remove the ambiguity from data (Bottou, *et al.*, 1994; Cireşan, Meier, Gambardella & Schmidhuber, 2010). It deals with extracting essential information from normalized images of isolated digits that serve as raw data in the form of vectors (Cireşan, *et al.*, 2010). The numbers in the images can be of different sizes, styles and orientation (Patel, *et al.*, 2011).In this study, a subset of MNIST dataset is used which contains ten thousands of scanned images of handwritten digits from 250 people. This data is divided into three parts, the first part contains 50,000 images to be used as training data. The second part is 10,000 images to be used as testing data. The third part contains 10,000 images for validation data. There are 28X28 pixels in size gray scale images. The training set, validation set and testing set are kept distinct to help neural network to learn from training set , validate the results from validation test and generate output from test set (LeCun, *et al.*, 1990; Liu, Nakashima, Sako & Fujisawa, 2003; LeCun, *et al.*, 1995). These are an example of digits of MNIST dataset collected in different hand writings. For example – a digit 2 can be represented in a different orientation with or without a loop at the bottom or straight line or curved line at the bottom.

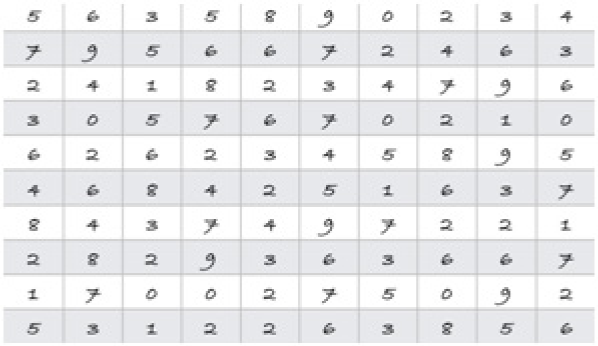

Figure 2. A sample set of MNIST dataset.

These are the 100 examples out of 60,000 used in the input of neural network in this paper.

3. PROPOSED DETECTION ALGORITHM

We now discuss this feed–forward neural network that was applied in this work to achieve highest possible accuracy in handwritten digit recognition (LeCun, *et al.*, 1989; Knerr, *et al.*, 1992).

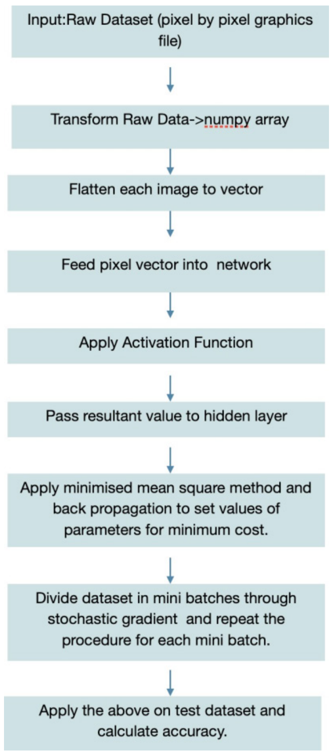
Input:Raw Dataset (pixel by pixel graphics file)
Transform Raw Data->numpy array
Flatten each image to vector
Feed pixel vector into network
Apply Activation Function
Pass resultant value to hidden layer
Apply minimised mean square method and back propagation to set values of parameters for minimum cost.
Divide dataset in mini batches through stochastic gradient and repeat the procedure for each mini batch.
Apply the above on test dataset and calculate accuracy.

Figure 3. Proposed Algorithm.

Above elaborates the steps of the proposed framework used in this study. The next section details this simulation.

4. PROPOSED METHODOLOGY

A. Neural Network

Figure 3 describes the architecture of the proposed neural network. It consists of an input layer, a hidden layer and an output layer. Each layer contains a number of neurons represented by a sigmoid function. So, the output of each neuron lies in the range of [0,1]. Every neurons output is determined by a weighted sum. jwjxj w is the weight of jthneurone having x input. The sum of the weighted sum and bias value determines the output value. The input layer consists of 784 neurons with each neuron representing each pixel value. Since each digit should be between 0and 9. So, the output layer consists of 10 neurons represented by the matrix (LeCun, *et al.*, 1990; Lauer, Suen & Bloch, 2007).

$$g(x) = \frac{1}{1+e^{-x}} \qquad (1)$$

Equation 1: Sigmoid Function.

Equation 1 defined shows the sigmoid function used. Here, g(x) represents the sigmoid function and the net value of the weighted sum is x. So, x is areal number. If the value of x is a very large positive number then the value of g(x) will be 1. And if the value of x is a very negative number then the value of g(x) is 0. Hence the graph shown by sigmoid function is

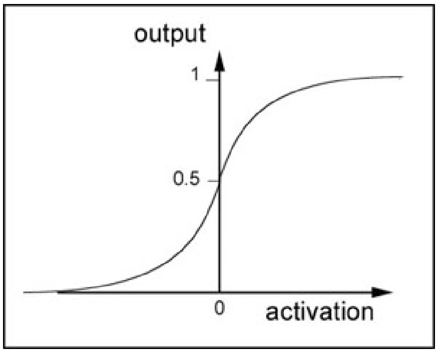

Figure 4. Graph of Sigmoid function.

But the problem with sigmoid function is that the for negative values the speed of neural networks become slow. That is in the second quadrant of the graph.

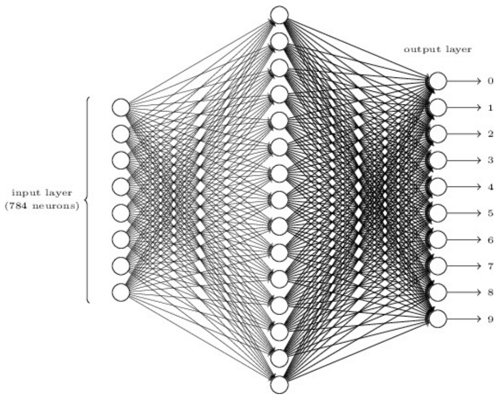

Figure 5. Neural Network Layer Architecture.

Figure 4 represents the whole multiple layered structures used in the neural network. The input layer contains 784 neurons that are each neuron for every pixel. The intensity of each pixel is the input to the neurons of the first layer. The activation function is performed on that input and the reuniting activation value is passed on to every neuron in the next layer. This same procedure is performed on the next layer. The output layer having 10 neurons So, the neuron with the highest activation value is the result. Figure 5 represents the output matrix where each row and column represents the output of the node in the output layer.

$$
\begin{array}{cccccccccc}
\multicolumn{10}{c}{\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}} \\
y_k^{(1)} & y_k^{(2)} & y_k^{(3)} & y_k^{(4)} & y_k^{(5)} & y_k^{(6)} & y_k^{(7)} & y_k^{(8)} & y_k^{(9)} & y_k^{(10)}
\end{array}
$$

Figure 6. Output Matrix for output Layer.

B. Gradient descent back propagation algorithm (**LeCun**, ***et al.*, 1990**)

This algorithm helps in finding weights and biases so that output from network approximates the ideal output for all training input x. This is done by the cost function. The cost function is a measure of how wrong the model is in terms of its ability to estimate the relationship between weights and biases (Widrow, *et al.*, 1994; Patel, *et al.*, 2011; Ganea, *et al.*, 2005; Lee, 1991). ((LeCun, *et al.*, 1990), equation 2) is also called mean squared error for the cost function.

$$C(w,b)=\frac{1}{2n}\sum_{x}\left\|y(x)-a\right\|^{2} \qquad (2)$$

Equation 2: Cost Function.

C(w,b) is non–negative since the sum of squared terms is always positive. If C(w,b) is approximately equal to 0 the ideal output is equal to obtained output. So, our algorithm is doing a good job. So, this algorithm minimizes the cost function which is represented by a graph (Shamim, Miah, Sarker, Rana & Jobair, 2018; Knerr et., 1992; LeCun, *et al.*, 1989).

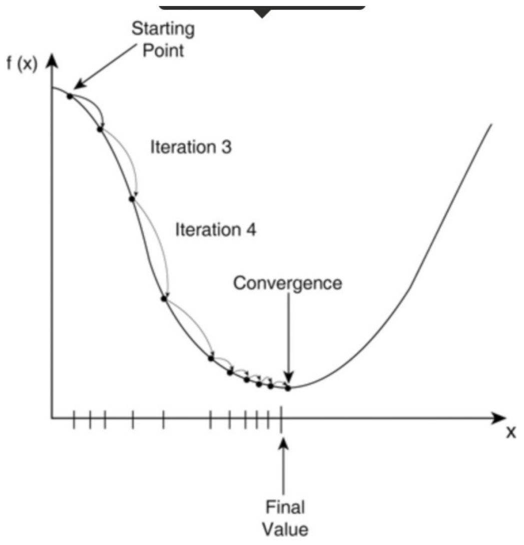

Figure 7. Curve Representing the Cost Function.

Figure 7 shows the 2D curve representing the linear equation depending on the two variables x and y same as weights and biases. Consider this curve as a valley in the mountains to reach the minimum height we have to reach the bottom of the valley. Similarly, if we roll down a ball from the above of the valley the ball will roll down and reach to the lower point in the valley. Neglecting the friction by air and ground and further physics laws of momentum.The minimum cost is the point of the lowest point in the valley. Hence the below two equations gives the minimum cost by partial differentiating the cost with respect to weight and bias. Solving this equation through this algorithm gives two ((Bottou, *et al.*, 1994), equations 3) for weight and bias as:

$$w_k \rightarrow w'_k = w_k - \eta \frac{\delta C}{\delta w_k} \quad (3)$$

$$b_l \rightarrow b'_l = b_l - \eta \frac{\delta C}{\delta w_k} \quad (4)$$

Equation 3, 4: Cost with respect to weights and biases.

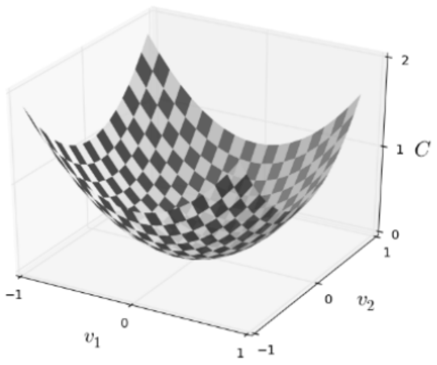

Figure 8. 2D Curve Representing the Cost Function.

C. Stochastic Gradient Descent

Stochastic Gradient Descent helps in speed up the gradient descent algorithm by dividing the training input into mini batches of each size m.

Step 1: Initialize the parameters

The cost function of this algorithm is dependent on two parameters that are weight and bias. So, random values are initialized to weight and bias vector for each input of neuron of each layer. The values of weights and biases are not initialized to 0 because that will be equivalent to the deletion of connections between the nodes.

Step 2: Feed–forward Algorithm

In this algorithm, the output from one layer is input to the next layer. That is there are no loops and information is fed forward not backward. This is achieved by the ((LeCun, *et al.*, 1990), equation 5).

$$a' = \sigma\left(wa + b\right) \qquad (5)$$

Equation 5: Feed forward equation

Here, a is the vector of activations of the second layer of neurons. To obtain a, a is multiplied by weight matrix w and vector b of biases is added. And then the function is applied element–wise to vector w.a+b. This is how inputs are fed forward. This equation gives the following results

Equation 6: Output Equation

Step 3: Calculate the gradient

The gradient for output and hidden layers are obtained by updating the weights and biases. This is done by shuffling the training data and then dividing into mini batches and then updating weights and biases for each mini batch.

Step 4: Update the weights

In the beginning, weights and biases are initialized some random values. Now, this model gives the output using those values which is far different from the ideal output. This gives a huge error. So, to reduce the error we will minimize the cost function and get the desired values of weight and biases. This is how back propagation works. ((LeCun, *et al.*, 1990), equation 7) is used.

$$w_k \rightarrow w'_k = w_k - \frac{\eta}{m}\sum_j \frac{\delta C_{X_j}}{\delta w_k} \quad (7)$$

$$b_l \rightarrow b'_l = b_l - \frac{\eta}{m}\sum_j \frac{\delta C_{X_j}}{\delta b_l} \quad (8)$$

Equation 7,8: Update weights and biases.

5. EXPERIMENTAL RESULTS

The number of neurons in hidden rate and learning rate are called hyper parameters of neural network (Gattal, *et al.*, 2016). So, changing the number of neurons and training the network again and again until the highest accuracy is achieved.

DOI: http://dx.doi.org/10.17993/3ctecno.2019.specialissue2.448-467

Table 1. Hidden Neurons Vs Accuracy.

Hidden Neurons	Accuracy(%)
10	91.46
30	95.07
50	95.86
60	95.16
65	96.30
70	87.56
80	87.33
100	78.37

Table 1 shows the analysis of data when the network is trained with a different number of neurons in the hidden layer and the accuracy they achieved. Figure 6 shows the graph of accuracy vs a number of hidden neurons. With 65 hidden neurons, mini–batch size of 10 and learning rate of 3 the highest accuracy achieved is at 96.30 Learning rate describes how quickly or slowly the network learns. Low learning rate slows down the learning process but converges smoothly. Larger learning rate speeds up the learning but may to converge. So, the desired learning rate is decaying learning rate. Table 2 shows the analysis data with different learning rate with their accuracy.Learning Rate should not be too low or too high. If it is too low then the neural network will take time to train and if it is too high the neural network will quickly forget the previous training and adapts new training faster. Both of which are of no use.

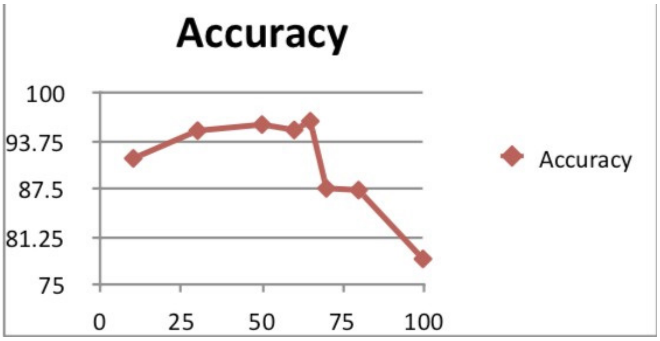

Figure 9. Accuracy Vs Hidden Neurons Graph.

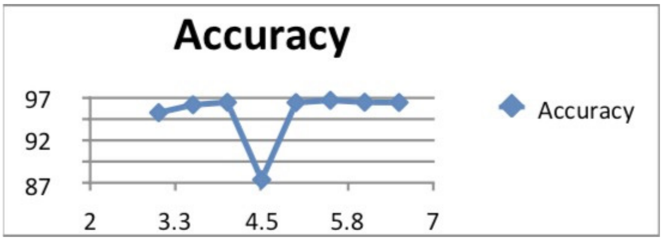

Figure 10. Accuracy Vs Learning Rate Graph.

Table 2. Learning Rate Vs Accuracy.

Learning Rate	Accuracy (%)
3	95.16
3.5	96.06
4	96.35
4.5	87.45
5	96.32
5.5	96.58
6	96.36
6.5	96.33

6. CONCLUSION

Performance of a neural network depends upon factors like the number of hidden neurons, learning rate, number of layers etc which are called hyper parameters of a network (Sakshica & Gupta, 2015; Lee, 1991). This research paper creates experiential support to showcase the effectiveness and efficiency of the neural networks to recognize handwritten digits. The main objective of this paper is to show the highest possible accuracy achieved for recognition of handwritten digit using the technique of feed forward and back propagation in neural networks. Although research is still going on in this field in many forms of LeNetarchitecture like LeNet–1, LeNet–4, Boosted LeNet–4.The main aim of the paper is to implement one of the methods of various methods. This branch of artificial intelligence is to develop a better network for all kinds of data sets with better performance.

7. FUTURE WORK

Neural networks have been applied to many applications like character recognition, signature recognition, and leaf decoding. The more training samples give more accuracy of the networks. By changing the model of the network from feed–forward to convolutional network the process can be faster and accuracy can be improved although it must be tried for different activation functions. Dropout technique which is turning off part of networks layers randomly to increase the regularization and decrease over fitting. In this case, accuracy for training data is much higher than testing data. It can be improvised for colored digits and can be used for code on post cards for sorting of letters. The proposed framework in this manuscript can also be further applied to some well–defined digit sets to accurately validate them. For example, the proposed framework in this manuscript can be applied on zip code digits, university enrollment numbers or mobile numbers of a given set in a given setting to automate their machine recognition or identification.

REFERENCES

Alwzwazy, A. H., Albe–Hadili, H. M., Alwan, Y. S. & Islam, N. E. (2016). Handwritten Digit Recognition using Convolutional Networks. *International Journal of Innovative Research in Computer and Communication Engineering, 4(2).*

Arel, I., Rose, D. C. & Karnowski, T. P. (2010). The deep machine learning new frontier in artificial intelligence research. *IEEE Computational Intelligence Magazine, vol. 5, no. 4*, pp. 13–18.

Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Jackel, L. D. … Vapnik, V. (1994). Comparison of classifier methods: a case study in handwritten digit recognition. In Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3 – Conference C: Signal Processing. doi: http://dx.doi.org/10.1109/ICPR.1994.576879

Cardoso, I. & Wichert, A. (2013). Handwritten digit recognition using biologically inspired features. *Neurocomputing, 99*(1), pp. 575–580. doi: http://dx.doi.org/10.1016/j.neucom.2012.07.027

Cireşan, D. C., Meier, U., Gambardella, L. M. & Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural computation, 22(12)*, pp. 3207–3220. doi: http://dx.doi.org/10.1162/NECO_a_00052

Ganea, E., Brezovan, M. & Ganea, S. (2005). Character Recognition using Neural Networks. *SINTES12*. Romania: IPA CIFATT.

Gattal, A., Djeddi, C., Chibani, Y. & Siddiqi, I. (2016). Isolated handwritten digit recognition using oBIFs and background features. *12th IAPR Workshop on Document Analysis Systems (DAS)*, (pp. 305–310).

Hegen, M. T., Demuth, H. B. & Beale, M. (1996). *Neural Network Design.* Boston, USA: PWS Publishing Co.

Knerr, S., Persnnaz, L. & Dreyfus, G. (1992). Handwritten Digit Recognition by Neural Networks with Single–layer Training. *IEEE Transactions on neural networks, 3(6)*, pp. 962–968.

Knerr, S., Personnaz, L. & Dreyfus, G. (1992). Handwritten digit recognition by neural networks with single–layer training. *IEEE Transactions on Neural Networks, 3(6)*, pp. 962–968. doi: http://dx.doi.org/10.1109/72.165597

Lauer, F., Suen, C. Y. & Bloch, G. (2007). A Trainable Feature Extractor for Handwritten Digit Recognition. *Pattern Recognition, 40(6)*, pp. 1816–1824. doi: http://dx.doi.org/10.1016/j.patcog.2006.10.011

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E. & Jackel, L. D. (1990). Handwritten digit recognition with a back–propagation network. *Advances in neural information processing systems*, pp. 396–404.

LeCun, Y., Jackel, L. D., Boser, B., Denker, J. S., Graf, H. p., Guyon, I., … Hubbard, W. (1989). Handwritten Digit Recognition: Applications of Neural Network Chips and Automatic Learning. *IEEE Communications Magazine, 27(11)*, pp. 41–46. doi: http://dx.doi.org/10.1109/35.41400

LeCun, Y., Jackel, L. D., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., … Vapnik, V. (1995). Learning Algorithms for Classification: A Comparison on Handwritten Digit Recognition. *Neural networks: the statistical mechanics perspective*, pp. 1–16.

Lee, Y. (1991). Handwritten Digit Recognition using k Nearest–Neighbor, Radial–basis Function, and Backpropagation Neural Networks. *Neural Computation, 3*(3), pp. 440–449. doi: http://dx.doi.org/10.1162/neco.1991.3.3.440

Liu, C. L., Nakashima, K., Sako, H. & Fujisawa, H. (2003). Handwritten Digit Recognition: Benchmarking of State–of–the–Art Techniques. *Pattern Recognition, 36(10)*, pp. 2271–2285.

Matan, O., Kiang, R. K., Stenard, C. E., Boser, B., Denker, J. S., Henderson, D., ... Lecun, Y. (1990). Handwritten Character Recognition Using Neural Network Architectures. In *Proceedings of the 4th US Postal Service Advanced Technology Conference, Washington D.C.* AT&T Bell Laboratories.

Mishra, A. & Singh, D. S. (2016). Handwritten Digit Recognition Using Neural Network Approaches and Feature Extraction Techniques: A Survey. *International Journal of Current Engineering and Scientific Research (IJCESR), 3(12)*, pp. 48–52.

Nielsen, M. A. (2018). *Neural Networks and Deep Learning.* Determination Press.

Patel, C., Patel, R. & Patel, P. (2011). Handwritten Character Recognition using Neural Network. *International Journal of Scientific and Engineering Research, 2(4)*, pp. 1–6.

Sakshica & Gupta, K. (2015). Handwritten Digit Recognition using Various Neural Network Approaches. *International Journal of AdvancedResearch in Computer and Communication Engineering, 4(2)*, pp. 78–80. doi: http://dx.doi.org/10.17148/IJARCCE.2015.4218

Shamim, M., Miah, M. B., Sarker, A., Rana, M. & Jobair, A. A. (2018). Handwritten Digit Recognition Using Machine Learning Algorithms. *Global Journal Of Computer Science And Technology, 18*(1). doi: http://dx.doi.org/10.17509/ijost.v3i1.10795

Widrow, B., Rumelhart, D. E. & Lehr, M. A. (1994). Neural networks: Applications in Industry, Business and Science. *Communications of the ACM, 37*, pp. 93–105. doi: http://dx.doi.org/10.1145/175247.175257

DOI: http://dx.doi.org/10.17993/3ctecno.2019.specialissue2.448-467