INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

**ISSN: 2454-132X**
**Impact factor: 4.295**
***(Volume 4, Issue 3)***
Available online at: www.ijariit.com

# Analysis of multi-layered perceptron, radial basis function and convolutional neural networks in recognizing handwritten digits

*Karishma Dasgaonkar*
*karishmadasgaonkar@gmail.com*
*Veermata Jijabai Technological Institute, Mumbai, Maharashtra*

*Swati Chopade*
*schopade@vjti.org.in*
*Veermata Jijabai Technological Institute, Mumbai, Maharashtra*

## ABSTRACT

***Identification of handwritten digits is one of the major areas of research in the field of character recognition. Artificial Neural Networks helps in computer vision that deals with how a computer could achieve high-level of understanding from digital images or videos. Thus, neural networks prove to be a boon in recognizing handwritten digits that are scanned as images. However, this paper aims at studying the working of specifically three neural networks - Multi-Layered Perceptron (MLP), Radial Basis Function (RBF) and Convolutional Neural Network (CNN). In order to focus majorly on the implementation of these three neural networks rather than the complexity of the dataset being used, we have used MNIST (Modified National Institute of Standard and Technology) dataset from keras library. The MNIST dataset contains 70,000 black and white images of handwritten English digits (60,000 training images and 10,000 testing images). In our study of the above three mentioned neural networks, we have used relu as activation function in the hidden layers and softmax as activation function in the final layer of neural network, Adam as an optimizer and cross-entropy as loss function. We have observed that all three networks give accuracy above 95%, however, the major difference is in its training time and error rate.***

***Keywords***— *Neural networks, Multi-Layered perceptron, MLP, Radial basis function, RBF, Convolutional neural network, CNN, MNIST, ANN*

## 1. INTRODUCTION

Identification of handwritten digits is one of the major areas of research in the field of character recognition and pattern recognition. Artificial Neural Networks (ANN) helps in computer vision or machine vision - a field that deals with how a computer could achieve high-level of understanding from digital images or videos. Thus, ANN proves to be a boon in recognizing handwritten digits that are scanned as images. ANN is basically a rough model of human brain structure that mimics the functioning of the human brain in order to do specific things such as classifying objects, etc. ANN is made up of multiple layers and each layer is made up of one or more nodes called a neuron. A basic neural network structure can be broken down into three sections. The first section consists of a layer commonly known as input layer. The input layer takes the input from the dataset, processes it and passes the generated output to the next section. The second section consists of at least one layer commonly known as a hidden layer. The hidden layer accepts the output generated by the input layer and applies an activation function over each input neurons in order to recognize the pattern of the input. The final section consists of the output layer. In our case, the output layer contains 10 neurons and if fired gives any of the output between 0 - 9.

## 2. METHODOLOGY

### A. MNIST Dataset

The MNIST (Modified National Institute of Standard and Technology) is one of the first datasets that proves the effectiveness of neural networks. Classifying MNIST dataset is considered as 'hello world' program of Machine Learning (ML) and is widely used for classification and image recognition task. The MNIST dataset is also often used to compare algorithm performances in research and hence, we have chosen this classic dataset primarily to focus more on the working and understanding of neural networks topologies rather than the complexity of the dataset being used. An MNIST dataset of handwritten digits contains a training set of 60,000 examples and a test set of 10, 000 examples. It is a subset of a larger set obtainable from NIST. The training set is used to teach the algorithm to predict the correct label, i. e., the integer 0 to 9, while the test set is used to check how accurate the trained network can make guesses. In ML, this is called supervised learning as we have the correct answers of the images we are making guesses about. The training set acts as a supervisor that corrects the neural network when it guesses wrong. Each MNIST image is 28 x 28 pixels, which means our input data is 28 rows x 28 columns matrix and moreover, MNIST contains 10 possible outputs, i.e., labels numbered as 0 to 9.

### B. Keras Library

Keras is an open source Deep Learning (DL) library developed in python [1]. We have used keras mainly for three reasons –

First, keras is a wrapper that allows the use of either Theano or TensorFlow backend. Theano is a low-level library that focuses inefficient computation whereas TensorFlow is another low-level library that is less mature than Theono, though, supported by Google and offers out-of-the-box distributed computing. That means, using keras we can easily switch between the two, depending on our needs. Second, keras's guiding principles such as modularity, minimalism, extensibility, and python-nativeness makes working in keras simple and enjoyable. And finally, keras has out-of-the-box implementations of common neural network structures. Thus, keras is the ideal library for our research.

### C. Activation function: ReLu and Softmax

A neuron in the artificial neural network calculates the weighted sum of its input, add a bias and then decide whether it should be 'fired' (activated) or not. The activation function helps to take this decision. There are various activation functions such as step function, sigmoid, tanH, ReLU, etc. Recent DL neural networks use Rectified Linear Unit (ReLU) activation function for the hidden layers. In fact, in the study of various activation functions (such as, sigmoid, tanH, eLu, ReLU, softSign, and softPlus), 98.43% classification accuracy is obtained with the ReLU activation function [4] which is the highest percentage of accuracy as compared to others. ReLU function gives an output x if x is positive and 0 otherwise. The main advantage of ReLU function over other activation functions is that it does not activate all the neurons at the same time unnecessarily, i. e., if the input to the neuron is negative it will convert it to zero and then the neuron does not get activated.

The output classes (0-9) of our problem are mutually exclusive and therefore we have used softmax classifier in the final layer of our neural networks. The softmax function compresses the output of each unit to be between 0 and 1, just like a sigmoid function. But unlike sigmoid, it also divides each output such that the sum total of the output is always equal to 1. The output of the softmax function is equal to a categorical probability distribution, i. e. it tells the probability if any of the classes are true. Mathematically, the softmax function is shown below, where z is a vector of inputs to the output layer (in our case, there are 10 elements in z as we have 10 output classes) and j indexes the output classes, so j = 1, 2, 3K (in our case, K = 10).

### D. Optimization function: Adam

The choice of optimization function in neural network model can mean the difference between good results (accuracy) within minutes, hours and days. Gradient descent is so far the most commonly used optimization procedure to optimize neural networks. Gradient descent is used to find the values of parameters of a function that minimizes a cost/loss function. Adam (Adaptive Moment Estimation) is one of the gradient descent algorithms that are an extension to Stochastic Gradient Descent (SGD). SGD upholds a single learning rate for all weight updates and the learning rate does not change during training whereas Adam computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. The bias-correction in Adam helps it to outperform other gradient descent algorithms such as RMSProp, Adadelta, Adagrad, etc. and therefore, Adam might be the best overall choice [2] as it achieves good results fast.

### E. Loss function: Cross-Entropy

Most learning neural networks calculate error as the difference between the actual output and the predicted output. The function that is used to calculate this error is known as Loss or Error or Cost function. There are various loss functions that give different errors for the same prediction and thus have a significant effect on the performance of the model. Loss functions can be categorized into three major categories:

1. Regressive loss functions: These functions are used in a regressive problem that is when the target variable is continuous. Most commonly used regressive loss function is Mean Square Error
2. Embedding loss function: These functions are used to measure whether two inputs are similar or dissimilar. Examples of this category are Hinge Error and Cosine Error
3. Classification loss functions: These functions are used in classification problem that is when the target variable is a probability value f(x) called the score for the input x. Since our research case problem of recognizing handwritten digits is a classification problem, we have used cross-entropy which is a classification loss function. There are two types of cross-entropy functions such as binary cross-entropy that is used to classify inputs into either of two classes (0 or 1) and categorical cross-entropy that is used to classify inputs into more than two classes. In our case, we have 10 classes or labels (0 to 9) and so we have used categorical cross-entropy as loss function for our neural networks.

## 3. IMPLEMENTATION

We have followed the basic framework of the artificial neural network while implementing the three algorithms – Multi-layered Perceptron (MLP), Radial Basis Function (RBF) and Convolutional Neural Network (CNN) – in order to analyze their functioning and performance in recognizing handwritten digits using MNIST dataset. The basic framework followed for our research is as follows:

1. Load MNIST dataset from keras library.
2. The MNIST dataset contains images of 28 x 28 pixels (2-dimensional array), flattened the images into a vector of 784-pixel numbers.
3. Normalize the pixel values ranging between 0 and 255 to fit into the scale of 0 and 1.
4. A number of nodes in the input layer is 784 and that in the output layer is 10 (for labels 0 to 9).
5. Assign random weights to all linkages (links from one neuron of one layer to the neuron in another layer) to start the algorithm.
6. Find the activation rates of hidden nodes by ReLU activation function using the inputs in the input layer and the linkages (input nodes --> hidden nodes).
7. Find the error rate at the output node by categorical cross-entropy loss function and recalibrate all the linkages between hidden nodes and output nodes.
8. Cascade down the error to hidden nodes using the weights and error found at output nodes by Adam optimizer.
9. Recalibrate the weights between the hidden node and the input nodes.
10. Repeat the process until the convergence criterion is met
11. Score the activation rate of the output nodes using the final linkage weights.

MLP, RBF and CNN - all three follow the above basic framework that we have defined for our research, however, RBF and CNN also provide an extension of this framework. The detailed functioning of these three neural networks algorithms are given below:

### F. Multi-layered Perceptron (MLP)

MLP strictly follows the basic framework defined above. MLP has at least one hidden layer and is referred to as "vanilla" neural network specifically when it has only one hidden layer.

### G. Radial Basis Function (RBF)

RBF neural network is structurally same as MLP. While both MLP and RBF could solve non-linear classification problems, RBF neural networks transform the input signal into another form which can then be fed into the network, unlike MLP. RBF neural networks are strictly limited to have just one hidden layer. This hidden layer is called feature vector. RBF neural networks increase the dimension of the feature vector; this, in turn, increases the linear separability of the feature vector. Gaussian activation functions are generally used for RBF networks. Only the hidden layer nodes perform the radian basis transformation (Gaussian activation) function and output layer performs the linear combination of the generated outputs in the hidden layer in order to give a final probabilistic value. Thus, in RBF networks, the classification is done only at hidden layer to output layer.

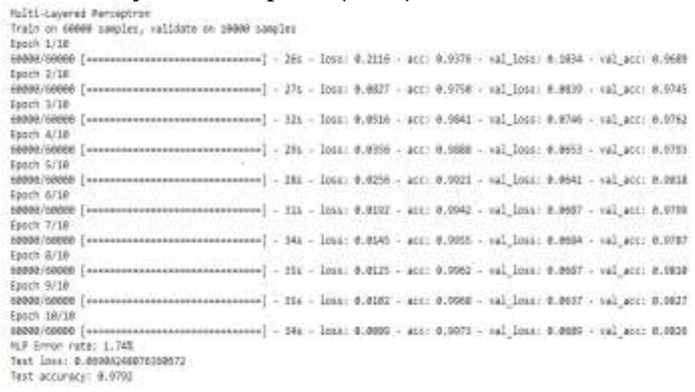
### H. Convolutional Neural Networks (CNN)

CNN is an extension to MLP. The first hidden layer is the convolutional layer. The convolutional layer has feature maps each with the size of 5 x 5 matrix and a rectifier activation function. The output of the convolutional layer is smaller than the original image. The next layer is the pooling layer that takes the maximum or average (in our case, max) of the output from the first hidden layer and is configured with a pool size of 2 x 2. The next layer is the activation layer that converts the 2D matrix to a vector and is passed to be processed by fully connected layers (standard MLP).

## 4. RESULTS

In our study of the above three mentioned neural networks, we have used relu as activation function in the hidden layer of MLP, CNN and Gaussian as activation function for RBF, softmax as activation function in the final layer of neural network, adam as an optimizer, cross-entropy as loss function, batch size (total number of training examples present in a single batch) of 64 and 1 hidden layer. Following are the output screenshots of our implementation of the three mentioned neural networks:

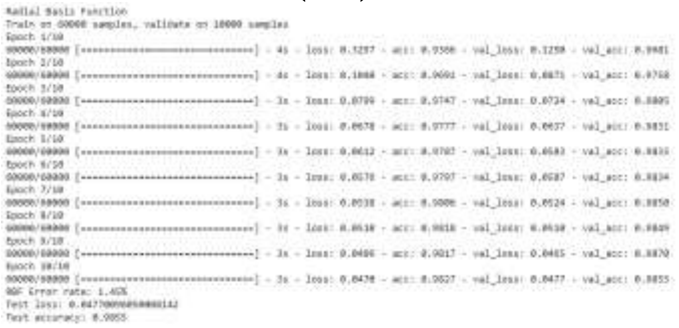
### I. Multi-layered Perceptron (MLP)



```
Multi-Layered Perceptron
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 26s - loss: 0.2116 - acc: 0.9376 - val_loss: 0.1034 - val_acc: 0.9688
Epoch 2/10
60000/60000 [==============================] - 27s - loss: 0.0827 - acc: 0.9750 - val_loss: 0.0839 - val_acc: 0.9745
Epoch 3/10
60000/60000 [==============================] - 32s - loss: 0.0516 - acc: 0.9841 - val_loss: 0.0746 - val_acc: 0.9762
Epoch 4/10
60000/60000 [==============================] - 28s - loss: 0.0356 - acc: 0.9888 - val_loss: 0.0653 - val_acc: 0.9793
Epoch 5/10
60000/60000 [==============================] - 28s - loss: 0.0256 - acc: 0.9921 - val_loss: 0.0641 - val_acc: 0.9818
Epoch 6/10
60000/60000 [==============================] - 31s - loss: 0.0192 - acc: 0.9942 - val_loss: 0.0687 - val_acc: 0.9799
Epoch 7/10
60000/60000 [==============================] - 34s - loss: 0.0145 - acc: 0.9955 - val_loss: 0.0684 - val_acc: 0.9787
Epoch 8/10
60000/60000 [==============================] - 31s - loss: 0.0125 - acc: 0.9962 - val_loss: 0.0687 - val_acc: 0.9818
Epoch 9/10
60000/60000 [==============================] - 31s - loss: 0.0102 - acc: 0.9968 - val_loss: 0.0637 - val_acc: 0.9827
Epoch 10/10
60000/60000 [==============================] - 34s - loss: 0.0099 - acc: 0.9971 - val_loss: 0.0689 - val_acc: 0.9828
MLP Error rate: 1.74%
Test loss: 0.0890424807830672
Test accuracy: 0.9792
```

In MLP, the error rate is 1.74% and test accuracy is 97.92% after 10 epochs (one epoch is when an entire dataset is passed forward and backward through the neural network only once). The time is taken by each epoch to complete increases by few seconds as the number of epochs increases. With the increase in the number of epochs, the current loss (loss) of the training set decreases and the accuracy (acc) of the training set increases gradually. Same goes for the validation set loss and accuracy (val_loss and val_acc).

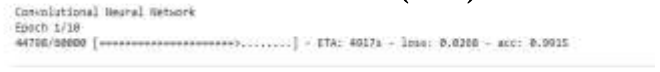
### J. Radial Basis Function (RBF)



```
Radial Basis Function
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 4s - loss: 0.3257 - acc: 0.9366 - val_loss: 0.1258 - val_acc: 0.9681
Epoch 2/10
60000/60000 [==============================] - 4s - loss: 0.1048 - acc: 0.9691 - val_loss: 0.0871 - val_acc: 0.9768
Epoch 3/10
60000/60000 [==============================] - 3s - loss: 0.0799 - acc: 0.9747 - val_loss: 0.0724 - val_acc: 0.9805
Epoch 4/10
60000/60000 [==============================] - 3s - loss: 0.0678 - acc: 0.9777 - val_loss: 0.0637 - val_acc: 0.9831
Epoch 5/10
60000/60000 [==============================] - 3s - loss: 0.0612 - acc: 0.9792 - val_loss: 0.0583 - val_acc: 0.9835
Epoch 6/10
60000/60000 [==============================] - 3s - loss: 0.0578 - acc: 0.9797 - val_loss: 0.0587 - val_acc: 0.9834
Epoch 7/10
60000/60000 [==============================] - 3s - loss: 0.0538 - acc: 0.9806 - val_loss: 0.0524 - val_acc: 0.9850
Epoch 8/10
60000/60000 [==============================] - 3s - loss: 0.0518 - acc: 0.9818 - val_loss: 0.0538 - val_acc: 0.9849
Epoch 9/10
60000/60000 [==============================] - 3s - loss: 0.0486 - acc: 0.9817 - val_loss: 0.0485 - val_acc: 0.9870
Epoch 10/10
60000/60000 [==============================] - 3s - loss: 0.0478 - acc: 0.9827 - val_loss: 0.0477 - val_acc: 0.9855
RBF Error rate: 1.45%
Test loss: 0.0477000005000142
Test accuracy: 0.9855
```

In RBF, the error rate is 1.45% and test accuracy is 98.55% after 10 epochs. The time taken by each epoch to complete is much less than that in MLP. With the increase in the number of epochs, the current loss (loss) of the training set decreases and the accuracy (acc) of the training set increases gradually. Same goes for the validation set loss and accuracy (val_loss and val_acc). However, the losses are more in the initial epochs as compared to MLP

### K. Convolutional Neural Networks (CNN)



```
Convolutional Neural Network
Epoch 1/10
44700/60000 [=====================>........] - ETA: 4017s - loss: 0.0200 - acc: 0.9915
```

## 5. CONCLUSIONS

RBF trains model faster as compared to MLP and CNN while CNN takes a longer time to train than the other two. MLP CNN gives the highest accuracy of 99%. However, if the dataset is not complex such as MNIST, it is better to use MLP or RBF or combination of both instead of CNN as CNN takes longer to train. Therefore, the selection of a neural network for classification problem highly depends upon the dataset used.

## 6. REFERENCES

[1] Ali Shatnawi, Ghadeer Al-Bdour, Raffi Al-Qurran and Mahmoud Al-Ayyoub, “A Comparative Study of Open Source Deep Learning Frameworks,” 9th International Conference on Information and Communication Systems (ICICS), 2018

[2] Sebastian Ruder, “An overview of gradient descent optimization algorithms,” Insight Centre for Data Analytics, NUI Galway, Aylien Ltd., Dublin, arXiv: 1609.04747v2 [cs.LG] 15 June 2017.

[3] Jyoti and Anil Kumar Rawat, “Scale Conjugate Gradient-based learning applied to Handwritten Digit Classification,” Conference on Information and Communication Technology (CICT17), 2017

[4] Fatih Ertam, Galip Aydin, “Data Classification with Deep Learning using Tensorflow,” 978-1-5386-0930-9/17 IEEE.