JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Performance Analysis of Artificial Neural Network on Digit Recognition using MNIST Database

Rani M T: rani.m.tharapathi@gmail.com (Dr Ambedkar Institute of Technology, Bangalore)

Dr Siddaraju: siddaraju.ait@gmail.com (Dr Ambedkar Institute of Technology, Bangalore)

Abstract: The MNIST database provides huge number of opportunities to learn and work using many techniques of Machine Learning. One of those an Artificial Neural Network place an important role in analyzing the database. The MNIST database has 70000 data points, of which 60000 is train data set and 10000 is test data set. There are some work has been carried using hardware components like FPGA, and CNN which is hardware intensive and more matrix complexity. This paper explains the combination of ReLU-Sigmoid and Leaky ReLU-Sigmoid with the simulation results in achieving the performance above 96% on a software platform (Jupyter Notebook).

Keywords: ANN, ReLU, Leaky-ReLU, Sigmoid.

Introduction: The majority use of the MNIST database provides good learning for those who are interested in analyzing the different aspect of large data. Among many techniques of machine learning and on observation of previous work on MNIST data insisted me to take up one more step of achieving performance analysis using ANN. An Artificial Neural Network provides better results in the implementation phase with specific observed inputs at every preface stage. The ANNs are parallel-distributed systems since they have the capacity to receive several inputs at the same time and process and distribute these inputs in an organized manner. An Artificial Neural Network is itself an architecture. With this, the information stored and shared in all the processing units (internal connected nodes of ANN) of ANNs improves the performance and reliability of the systems implemented. The implementation considers the number of inputs and outputs, number of neurons, number of hidden layers and learning rate all together represents the performance accuracy with the involved inputs.

A. Existing system: Numerous investigations have been conducted in the field of digit recognition. The use of CNN and FPGA takes advantage of the high power consumption, and because CNN is hardware intensive, it can be partially connected in middle layers. Every iteration of input in hardware implementation at every stage consumes more power. To avoid this, the solution is briefly explained in the proposed method.
B. Proposed system: As discussed above, to do recognition of digits on MNIST, many techniques have already been implemented, like CNN, CNN with FPGA, and many others. The CNN is hardware-intensive, and implementing CNN with an FPGA requires more computational time and more matrix multiplication. Taking advantage of these, it is proposed here to implement the analysis of the performance of ANN on the MNIST database on a software platform. The parameters that are taken into account here, i.e., the most prevalent implementations, are Sigmoid, ReLU, and Leaky ReLU. The accuracy must be close to 100% or above 95%. This would ensure that the hardware implementation matched the software implementation as closely as possible.

Literature and Related work:

The literature work is a systematic study. A type of secondary research work that helps in understanding and explaining the well-defined methods and methodology. The purpose of literature work is to gather knowledge about the data set, the previous methods used and what are the concepts involved in it. In particular the accuracy factor is more focused here. The comparison of CNN, RNN, and DNN gives the information about the accuracy factor. The combination of CNN-SVM gives the accuracy about 94%, since the CNN has a drawback of more complexity and hardware intensive, whereas SVM doesn't need larger set to meet accuracy. Also the concept of activation function plays an important role in firing the input for the non-linearity model. To proceed with the implementation here MNIST data base is used. "Multi-Digit Number Classification using MNIST and ANN", In this paper they have used the ANN technique which consists of input, hidden and output layer. The have demonstrated the ANN approached method initially they have identified or recognized the digit then they tested for hand written (pen on paper). Then they evaluated for accuracy and it has given up to 93%. The concept here is recognizing the digit is already done by many researchers using MATLAB, CNN and many. But improving an accuracy with the minimum factor with a lesser gradient is influenced from this paper. So I would like to use ReLU and Sigmoid in ANN to build a three layer Neural Network to meet the accuracy higher and how ReLU-Sigmoid combination effectively works and saturate to the results.

An Overview of Neural Network concepts

1. Artificial Neural Network: ANN are usually called neural networks are interconnected systems composed of many single processing units. It has composed of input, output and hidden layers. Where each layer contains neurons called as nodes which performs various operations.

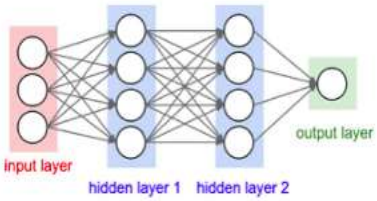

Figure 1: Artificial Neural Network

2. Neuron: With the help of biological neurons, a neuron in a neural network can be better comprehended. A biological neuron is analogous to an artificial neuron. It takes in information from other neurons, processes it, and then provides an output. The weighted average of a neuron's input is computed, and the sum is processed through a nonlinear function, sometimes referred to as an activation function, such as the sigmoid.
3. Activation function: An artificial neural network's activation function is crucial; it determines whether or not a neuron should be engaged. The activation function of a node in an artificial neural network determines its output given an input or group of inputs.
4. ReLU and Leaky ReLU: Rectified Linear Units (RLU) is a relatively new invention. It's a deceptively simple formula: maximum (0,z). It's not linear, despite its name and look, and offers the same benefits as Sigmoid (i.e., the capacity to learn nonlinear functions), but with superior performance. Whereas LeakyRelu is a variant of ReLU. Instead of being 0 when z<0z<0, a leaky ReLU allows a small, non-zero, constant gradient α (Normally, α=0.01α=0.01).

MNIST Database: MNIST database is one of the large database for digit recognition applications and different purposes. This has total of 70000 data points and of which train data is 60000 and test data is 10000. The image pixel has 28*28 which is 784 units for each digit has the representation between 0 to 255. The MNIST provides many opportunities to work and analyze the data in various techniques in the specialization of data analysis. Here it is privilege to come up with a new try and idea of expressing the concept of performance analysis of neural network using MNIST database on a software platform.

Implementation and Results: The implementation has started with initializing the both train and test data. The one hot encoding is applied here since the binary representation enables the digits from 0 to 9. The input is initiated with the 784 input units, the hidden layer and activation function value is varied according to the observation at each step. Following with the slope of the ReLU. The output unit is defined with sigmoid activation function which has 10 output value.

Implementation cases and stages involved:

Case 1: To vary the learning rate with decreased number of hidden layers and keep the accuracy greater than 95%.

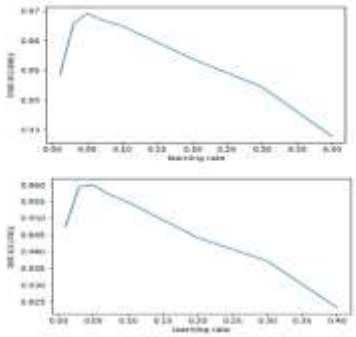

Figure 2: Accuracy with respect to HN=70(decreased) and varied LR

Deductions: The learning rate can be fixed to .05 and number of hidden nodes is further reduced to observe the change.

Case 2: To vary the hidden layers with fixed learning rate and keep the accuracy greater than 95%.

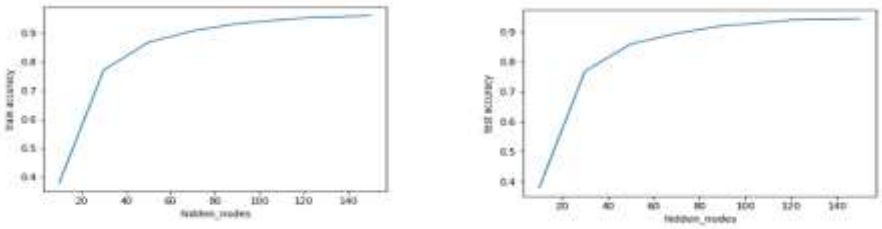

Figure 3: Accuracy with respect to varied hidden nodes and LR=0.05

Deductions: Since the curve was increasing with accuracy, next Implementation case was implemented with hidden nodes as 30(a bit lower) and learning rate was lowered and varied.

Case 3: To vary the learning rate with low hidden layers and keep the accuracy greater than 95%.

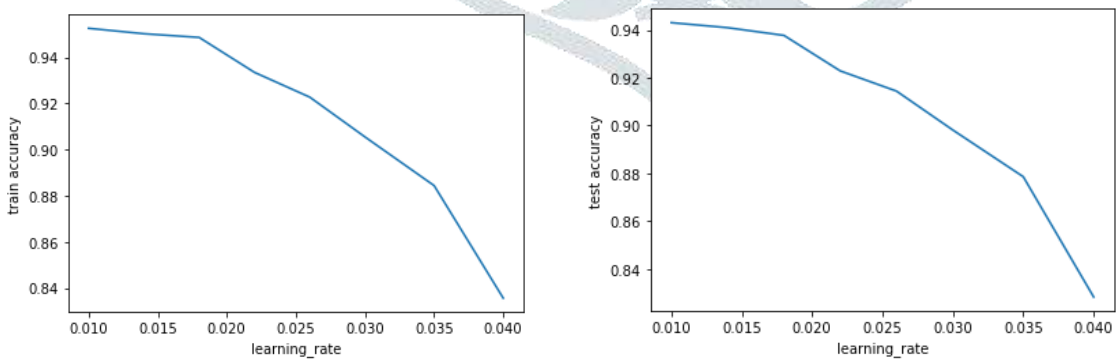

Figure 4: Accuracy with respect to hidden nodes = 30 and varied LR

Deductions: Since the curve was increasing with accuracy, next Implementation case was implemented with hidden nodes as 30(a bit lower) and learning rate was lowered and varied.

Case 4: To vary the hidden nodes with fixed learning rate .01 and keep the accuracy greater than 95%.

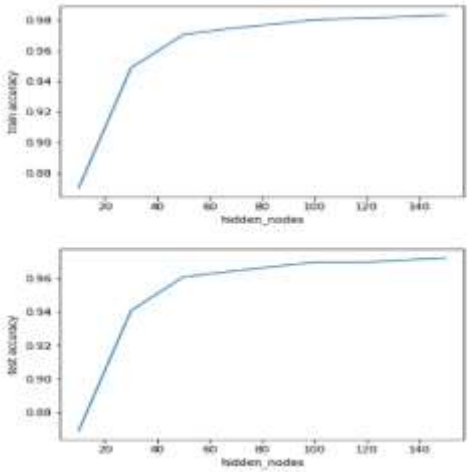

Figure 5: Accuracy with respect to varied hidden nodes and LR=0.01

Deductions: The number of hidden layers is fixed at 50.

Case 5: To vary the learning rate with fixed hidden nodes layer 1(50) and keep the accuracy greater than 95%.

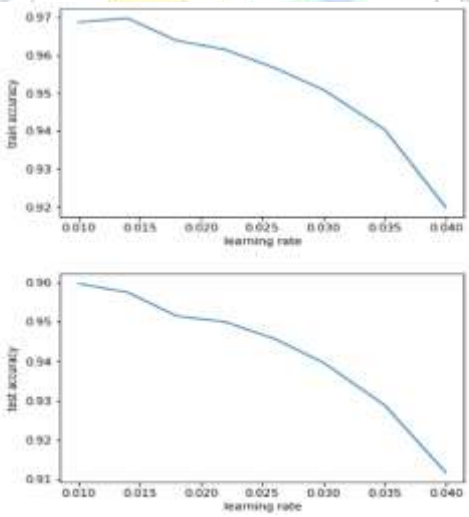

Figure 6: Accuracy with respect to hidden nodes = 50 and varied LR

Deductions: The learning rate was fixed at .01

Case 6: To see the performance of 50 hidden nodes and learning rate of .01

```
train accuracy : [0.9670847222222222]
test accuracy : [0.9597999999999999]
```

Figure 7: Accuracy with respect to hidden nodes=50 and LR=0.01

Deductions: The combination of relu and sigmoid and hidden layer of 50 with a learning rate of .01 meets the requirement >95% for test data. From Implementation case 4 and Implementation case 5, Implementation case 6 was tested with a learning rate of .01 and hidden layer 50 of nodes.

Case 7: To implement on leaky relu with varying hidden nodes.

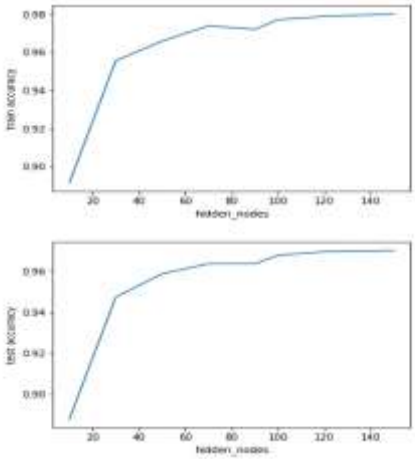

Figure 8: Leaky ReLU - Accuracy with respect to varied hidden nodes and LR=0.01

Deductions: The leaky relu gave a better performance than relu, Since it is able to use the lower slope factor in the negative side. So it can avoid vanishing the less non zero values on the negative side.

Case 8: To implement on leaky relu with varying learning rate.

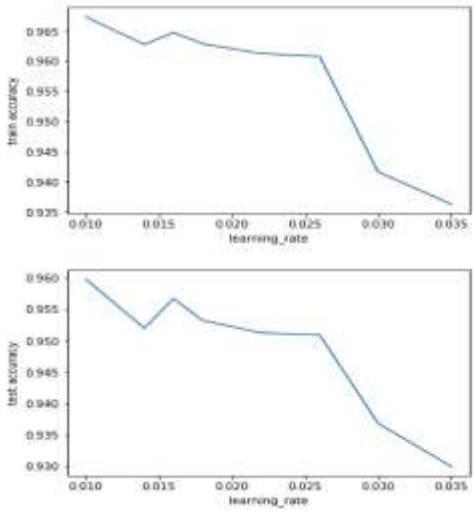

Figure 9: Accuracy with respect to hidden nodes =50 and varied LR

Deductions: Like relu , the leaky relu gave the best performance at .01 learning rate and with MSE.

Discussion: In the Implementation case 7, the accuracy is greater than 96% with the fixed learning rate at 0.01 and varying hidden nodes. It is known that, the Leaky ReLU is improved version of ReLU, it has the small slope for the negative values as compared to ReLU where it deactivates the neuron in that region if the gradient is zero or below. So to try out Leaky ReLU with the case 6, it is noticed that the Leaky ReLU gives the better performance than ReLU with accuracy greater than 96%.

In the Implementation case 8, the accuracy is greater than 96% with the varying learning rate and number of hidden nodes as 50. The Leaky ReLU is used to try out for the case 7 to understand the performance when using the varying learning rate with the cost function MSE. So even when suing the carried learning rate it is observed that the accuracy is still high.

So it is to explain that by using the Leaky ReLU on MNIST database performs well on achieving expected accuracy level above 96%.

Comparison table of all implementation stages.

Cases	Number of neurons	Sigmoid	Activation	Hidden Unit	Learning rate	Observation_train	Observation_test
Implementation case 1	784	10	ReLU	70	Varying	>96	>95
Implementation case 2	784	10	ReLU	Varying	0.05	>97	>96
Implementation case 3	784	10	ReLU	30	Varying	>95	>94
Implementation case 4	784	10	ReLU	Varying	0.01	>98	>96
Implementation case 5	784	10	ReLU	50	Varying	>97	>96
Implementation case 6	784	10	ReLU	50	0.01	>96	>95
Implementation case 7	784	10	**LeakyReLU**	Varying	0.01	>98	**>96**
Implementation case 8	784	10	**LeakyReLU**	50	Varying	>96	**>96**

Figure 10: Comparison of Accuracy observation of ANN

Conclusion: The above implementations give different accuracy based on the parameters used, like the number of neurons in the hidden layer, the learning rate, the cost functions, the activation function, etc. The Rectified Linear Unit (ReLU) activation function is used in the above implementations because it works well for ANN. The hardware implementation requires hardware components like FPGAs, i.e., Field Programmable Gate Arrays. This also considers the number of inputs and outputs, the number of neurons, the number of layers, and the number of connections to each neuron, so on. To do all these implementations, the FPGA requires high power consumption. Because each implementation on hardware consumes more power, here it is considered implementing it on a software platform using the ReLU activation function. The work is based on the accuracy of each implementation. Combinations of ReLU-Sigmoid (Leaky-ReLU) in cases 7 and 8 give the best performance based on the parameters used and perform the best among all, achieving above 96% accuracy.

Acknowledge: I would like to thank my guide Dr. Siddaraju (HoD, Dept of CSE, Dr AIT, Bangalore) for his valuable suggestions and comments that helped improving this work. He has been always there to solve any query and also guide me in the right direction regarding the paper and project.

References

1. Y. LeCun, "The MNIST database of handwritten digits," http://yann. lecun.com/exdb/mnist/, 1998.
2. *A Comparative Study on Handwriting Digit Recognition Using Neural Networks 978-1-5386-2269-8/17 IEEE DOI 10.1109/ICPET.2017.20*
3. *"FPGA Implementation of CNN for Handwritten Digit Recognition 978-1-7281-4390-3/20 978-1-7281-4390-3/20 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC 2020)"*
4. A novel hybrid CNN–SVM classifier for recognizing handwritten digits Xiao-Xiao Niu n , Ching Y. Suen Elsevier Ltd. All rights reserved. doi:10.1016/j.patcog.2011.09.021
5. "*Method of Selecting an Optimal Activation Function in Perceptron for Recognition of Simple Objects 978-1-5386-5710-2/18/ ©2018 IEEE*"

6. "Handwritten Digits Recognition Using Machine learning, International Journal of Information Sciences and Application (IJISA). ISSN 0974-2255, Vol.11, No.1, 2019, (Special Issue)"
7. Multi-Digit Number Classification using MNIST and ANN : (IJERT) ISSN: 2278-0181 Vol. 9 Issue 05, May-2020,