COMPARATIVE ANALYSIS OF PROCESSORS PERFORMANCE USING ARTIFICIAL NEURAL NETWORK

Vijay Maurya[1], Anuj Verma[2], Akanksha Singh[3]

[1]*Assistant Prof., Electronics Engineering Department, Institute of Engineering & Technology, U.P., India*
[2]*Assistant Prof., Electrical Engineering Department, International Maritime Institute, U.P., India*
[2]*Asst. Prof., Computer Science Dept., Rameshwaram Institute of Technology & Management, U.P., India*

ABSTRACT

In the present era of plurality and diversity in electronic world taking decision for choosing the best one from a pool of products is a prudent approach. Machine learning helps us a lot in deciding the optimal choice. Artificial Neural Network is a family of statistical learning algorithm inspired by biological neural network. It can be incorporated to take a decision at par with smart human perception.

The purpose of this paper is to compare performance of CPU processors using some characteristics that are used as input, and subsequently comparing that result with existing performance that is published.

Keyword: - *Machine Learning; ANN algorithm; data objects; neural networks; Training*

1. INTRODUCTION

The Artificial Neural Network (or ANN for short) is used to estimate or approximate function that can be dependent on a large number of inputs which are generally unknown. Artificial neural networks are generally presented as systems of interconnected "neurons" which can compute values from inputs, and are capable of machine learning as well as pattern recognition. There is no single formal definition of what an artificial neural network is. However, a class of statistical models may commonly be called "Neural" if they possess the following characteristics:

1. Consist of sets of adaptive weights, i.e. numerical parameters that are tuned by a learning algorithm, &
2. are capable of approximating non-linear functions of their inputs.

The adaptive weights are conceptually connection strengths between neurons, which are activated during training and prediction. The objective of this paper is to train a neural network so as to compare relative performance of CPU processors using some characteristics that are used as input, and subsequently comparing that result with existing performance that is published. The data set contains 209 instances with the total of 10 attributes. The first two attributes are vendor name and model name of the processor, and the rest of the attributes are as follows:

1. Machine cycle time in nanoseconds (integer)
2. Minimum main memory in kilobytes (integer)
3. Maximum main memory in kilobytes (integer)
4. Cache memory in kilobytes (integer)
5. Minimum channels in units (integer)
6. Maximum channels in units (integer)
7. Published relative performance (integer)
8. Estimated relative performance (integer)

2. STATE OF THE ART

There have been numerous works done in the area of prediction of performance. Lee et al. [1] use regression models to predict performance and power usage of the applications found in the SPECjbb and SPEC2000 benchmarks. As in the previous reference, the data points are created using simulations. Kahn et al. [2] uses predictive modeling, a

machine learning technique to tackle the problem of accurately predicting the behavior of unseen configurations in CMP environment. Dubach et al. [3] has used a combination of linear regression models in conjunction with neural networks to a model that can predict the performance of programs on any micro-architectural configuration with only using 32 further simulations.

3. SIMULATION SEUP

Neural networks can be used for prediction with various levels of success. The advantage of then includes automatic learning of dependencies only from measured data without any need to add further information (such as type of dependency like with the regression), and that is the reason this approach will be used here. In order to setup a neural network and training (automatic learning) it, there are following five steps to be made:
Procedure to train a Neural Network:

3.1. Normalization of data

For this experiment the data that is prepared cannot be used in its original form because of the difference in units and values between attributes. For instance machine cycle time ranges between 17 and 1500 nanoseconds but the minimum channels ranges only between 0 and 52 units. That is the reason the data must first be normalized. From the ten attributes only seven will be normalized. The first two have no significant interest in this experiment and are also not an integer values and the last attribute (estimated relative performance) will be compared with our results which will be later converted using the same formula but in the other direction.

3.2. Creating training set

Next we will add a training set to our project. A training set is a list of data that represents inputs and outputs that will be used in neural network in order to train it. Training simply means finding weights that correspond to neurons in our artificial networks so that the error between expected and known results is minimal or below certain threshold.

3.3. Creating a neural network

We do this by using "nntool" command in MATLAB. A window should appear where we must choose a type of a neural network to use. Then create a neural network and choose a network function. We used three types of neural networks.

Network1: A **cascade forward back-propagation** network function based neural network: (fig-1)

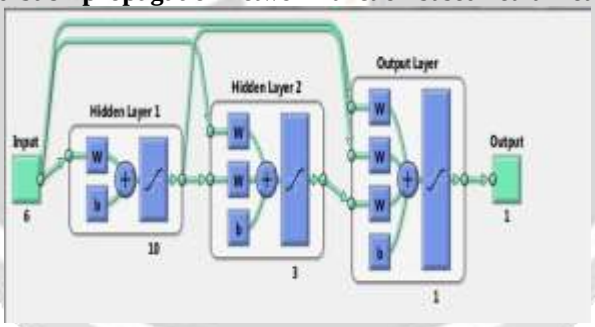

Fig-1 cascade forward back-propagation network

Network2: Another neural network using **feed forward back-propagation** network function: (fig: 2)

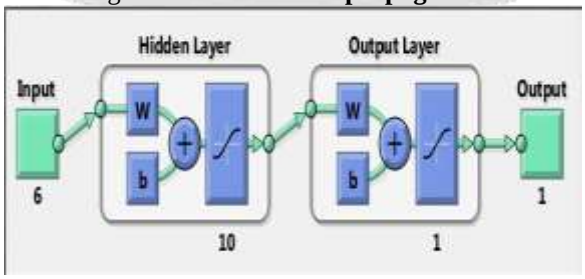

Fig-2 feed forward back-propagation network

Network3: Another neural network using **feed forward back-propagation with 3 layers** network function: (fig: 3)

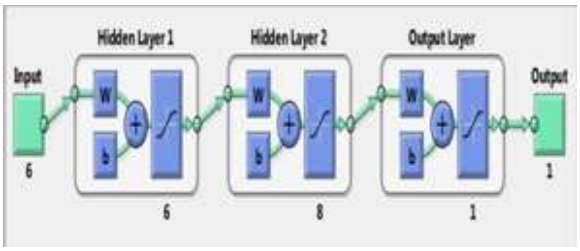

Fig-3 feed forward back-propagation with 3 layers network

3.4. Training

First thing we need to do is connect our training set to our network by simply clicking at the training set name whilst our neural network is selected. The method involves supplementing the current weight adjustments with a fraction of the most recent weight adjustment.

To train it we need to click on the Train button.

Training Network1:

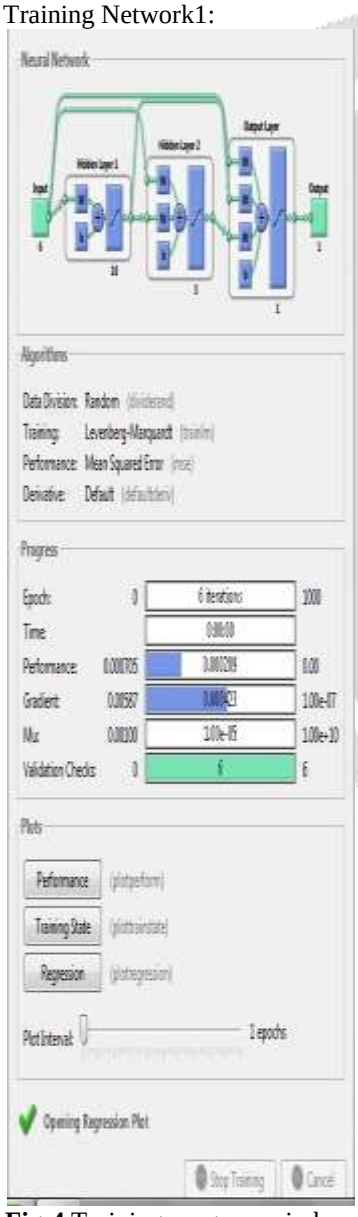

Fig-4 Training progress window of network1

Training Network2:

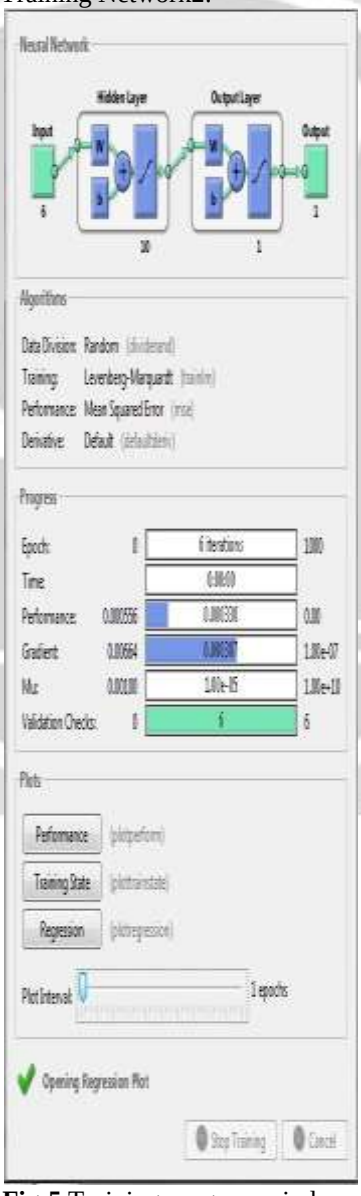

Fig-5 Training progress window of network2

Training Network3:

Fig-6 Training progress window of network3

3.5. Testing

In order to test how our network behaves with novel data, we will divide our data set in two parts – training set and test set.

Usually, when dividing dataset like this there must be enough data to be use in order to train the network but not too much. When the network is over trained any minor deviation from expected data can be considered as a bigger error that it is. That is the reason why those networks are not good in prediction which is exactly what we don't need.

In this research we will try two different proportions, 70/30 and 80/20, that is for example in first case, about 70% of data will be our training set and the rest we will use to test our network.

The data will be chosen randomly because if we chose similar data the network would not be able to perform very well with the data that differs significantly from our original.

4. RESULTS

The results are displayed below. So as to have a comparative analysis results are displayed parallel based on mean square error for each of the network created.

4.1. for network1:

Cascade forward back-propagation network function based neural network

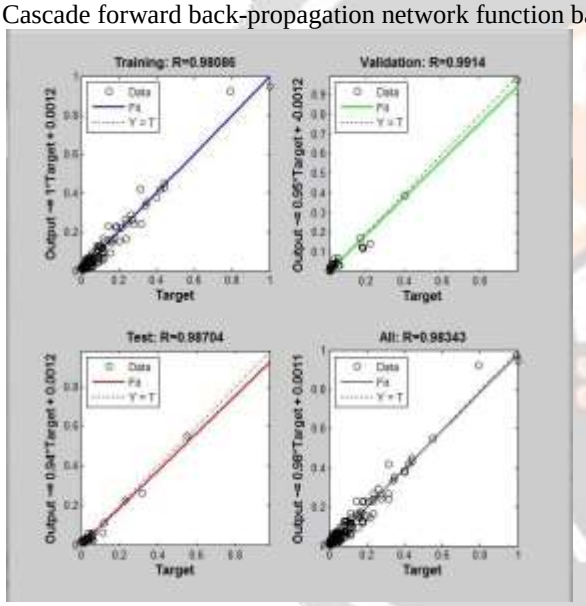

Fig- 7 TRAINING, VALIDATION & TEST PLOTS:

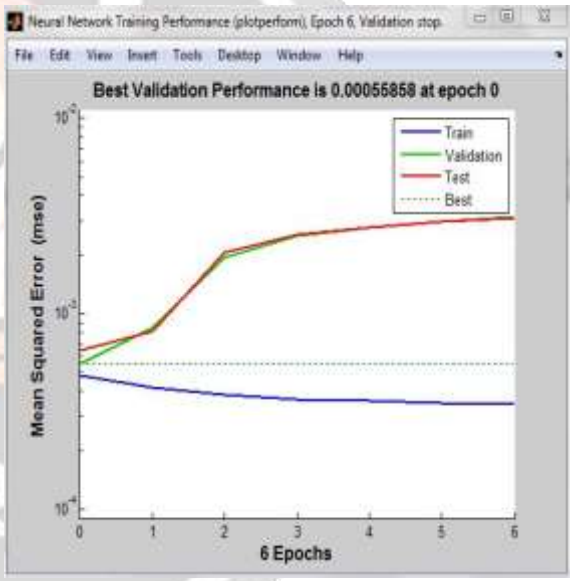

Fig- 8 MEAN SQUARE ERROR PLOT:

4.2. For network2:

Feed forward back-propagation network function

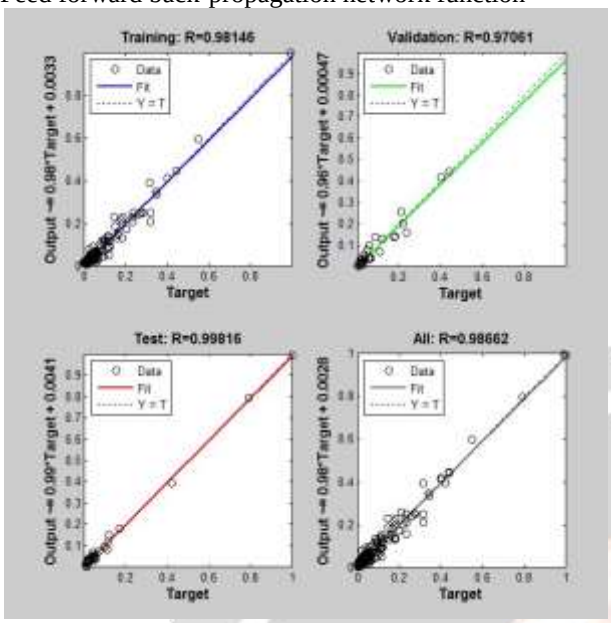

Fig- 9 TRAINING, VALIDATION & TEST PLOTS:

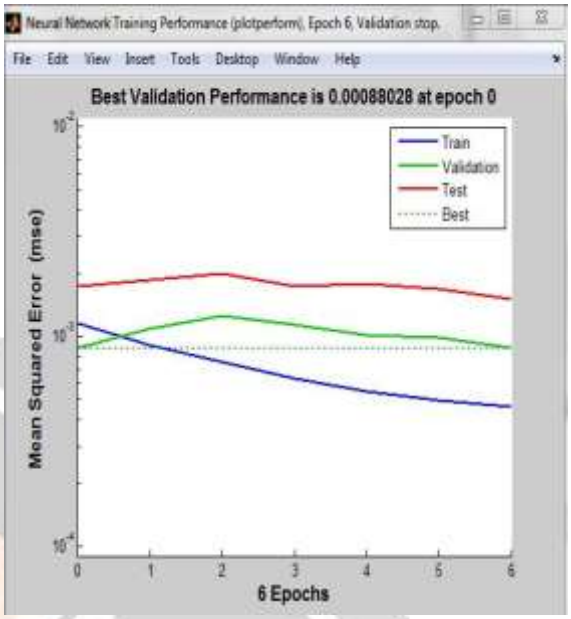

Fig- 10 MEAN SQUARE ERROR PLOT:

Clearly from the above two network's mean square plots network2 is performing better. Network2 train curve (fig.9) follow validation curve well than in Network1 (fig.7) and also the test curve is following the validation curve very well as compared to Network1 over all the Epochs as it should do being a feed forward back-propagation network function used in the Network2. Best result is indicated by dotted lines which held once.

4.3. For network3:

Feed forward back-propagation with 3 layers network function

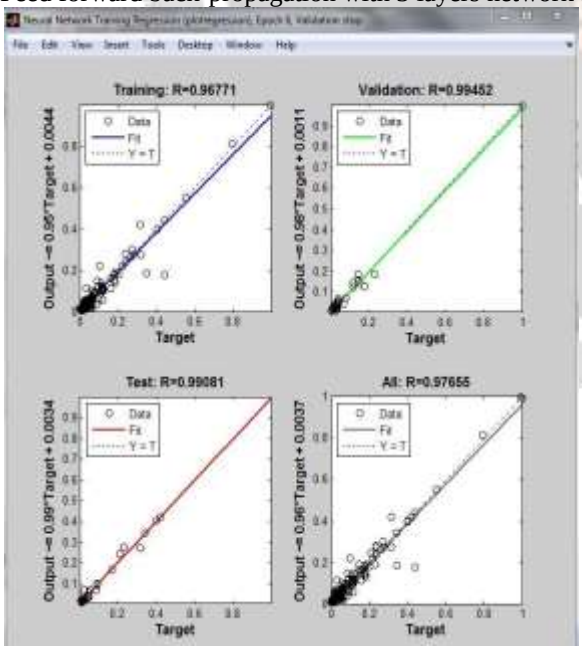

Fig- 11 TRAINING, VALIDATION & TEST PLOTS:

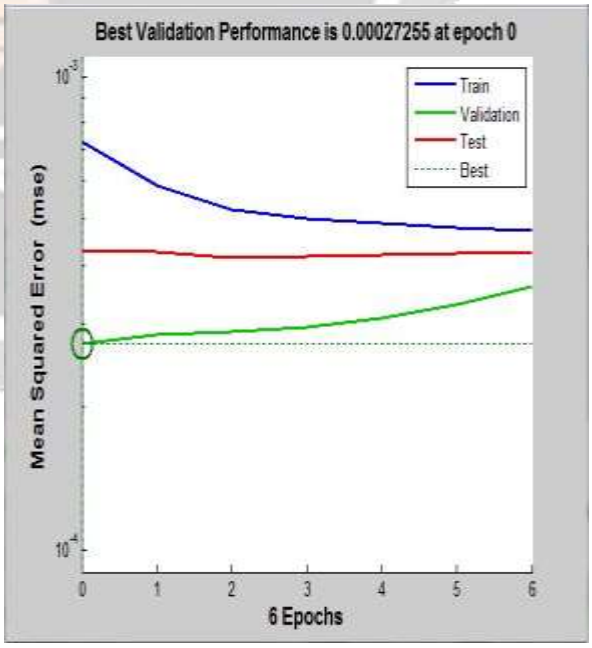

Fig- 12 MEAN SQUARE ERROR PLOT:

Now the Network3 here is providing the best results among both the networks Network1 and Network2. In Network3 train curve follow validation curve (fig.11) more properly than the Network1 and Network2 follows over

all Epochs as it should do because of being feed forward back-propagation network function used with 3 layers in Network3 and also the test curve is lying here in between the train and validation curves which clarifies about the results outcome to be more accurate.

5. CONCLUSION

We created three different neural network architectures in this experiment. We concluded that with recognizing data using neural network the best way to go is to use 8 neurons in a hidden layer. The results that we got worked better in all of our cases where compared to linear regression method that is located in our dataset, and should be used to estimate relative performance of CPU-s.

6. REFERENCES

[1] Lee B. C. and Brooks D. M. Accurate and Efficient Regression Modeling for Micro-architectural Performance and Power Prediction. In Proc. of the ASPLOS, Oct 2006, San Jose, CA

[2] Khan S., Xekalakis P., Cavazos J. and Cintra M. Using Predictive Modeling for Cross-Program Design Space Exploration in Multicore Systems. In Proc. of the Int. Conf. on PACT, Sep 2007, Brasov, Romania

[3] Dubach C., Jones T. M. and O'Boyle M. F. P. Micro-architectural Design Space Exploration Using An Architecture-Centric Approach. In Proc. of the Int. Symp. on Micro-architecture, Dec 2007, Chicago, IL

[4] Russell, Ingrid. "Neural Networks Module".Retrieved 2012

[5] http://archive.ics.uci.edu/ml/datasets/Computer+Hardware

[6] Kibler,D. & Aha,D. (1988). Instance-Based Prediction of Real-Valued Attributes. In Proceedings of the CSCSI (Canadian AI) Conference

[7] Machine Learning Models to Predict Performance of Computer System Design Alternatives by Berkin Ozisikyilmaz, Gokhan Memik, Alok Choudhary, Northwestern University, Evan