Reformation with Neural Network in Automated Software Testing

Ayush Kumar Yogi[#1]

[#1]*Faculty, MCA Department, Govt. Engineering College*
Ajmer, Rajasthan, India

***Abstract*— In the practical scenario, neural network has showed its efficient performance in diversity of applications. As the software testing is the main moving wheel in the software development, there must be adoption of new techniques to move faster the testing wheels. In this paper the impact of neural network on software testing and how we can improve and get efficient results in testing phase is described. An idea proposed to professional automated testing tools developers how they can absorb neural network technique in their testing tool. Generally after testing phase completes and actual results have obtained then at comparative phase of actual and desired test results of test cases the neural network is adopted. To absorb neural network in software testing gives the meaning to test software using neural network technique from its initial input data for test to at the end of test.**

***Keywords*— Neural network, software testing, professional automated testing tools.**

I. INTRODUCTION

Software development companies have many alternative choices to develop the softwares as available development models. Industries changed their working style from waterfall model to verification and validation (VV) model. As the development of softwares become fast as on-shelf product merging technique and emergence of software testing came into light after the ignorance of testing as a result in faulty, erroneous, bad service softwares. The time needed to test softwares depends on its functionalities. The automated testing tools such as QTP, LR, Rational testing tool have provided great efficiency to test softwares. Here is the idea proposed in this paper that how testing tool developers can adopt neural network and can design such a tool that they may provide higher efficiency than previous testing tools. Time increase in testing is the big challenging factor for project completion. The numbers of test cases increase with their inputs and we can eliminate this by using fault-based testing. Fault-based testing aims to detecting the higher priority section of code that will be utilized mostly in the software or to detect particular emphasized faults. Usually this is used in white-box testing.

Neural network is the connection of multiple neurons exchanging information or data with their calculative and limit based function. Each neuron in simple term related to testing is a module that is capable as holding input data, operating or calculations on that data and taking output from other neurons. According to the applications the structure of neural network is two layer and multilayer. Multilayer neural network is also named as multi-layer perceptron (MLP) [1] or feed forward artificial neural network model. A bias and a threshold function is used based on requirement of nature of application. Bias is used as decreasing the inputs and threshold is used to increasing and controlling inputs.

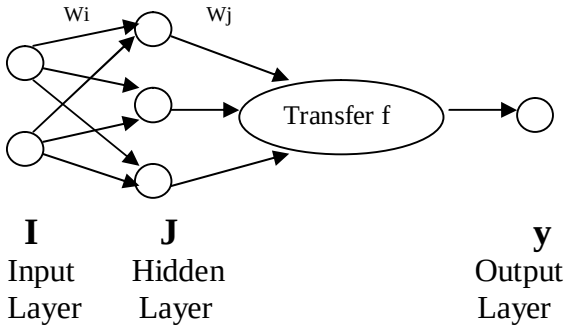

Fig. 1 Multilayer neural network architecture

Here W_i = weight associated with input layer neurons

W_j= weight associated with hidden layer neurons
I= identifier for neurons in input layer
J= identifier for neurons in hidden layer
y= output at output layer

In figure 1, input neurons take inputs; hidden layer neurons are responsible for taking inputs and processing further calculation and fetching the result to the transfer function. Transfer function takes the results from hidden layer neurons and calculates the output. As a transfer function a sigmoid function is used if the range of output signals vary from 0 to 1 that is [0, 1]. If input layer neurons vary from $I_{1, 2,3.....n}$ and weights associated with each input neuron is $W_{I1,I2,I3.....In}$ then at hidden layer each neuron perform the following calculations-

$$net\_value= \sum_{I=1}^{n} I_j * W_{IJ} \qquad (1)$$

$y_I = 1/(1+e^{-net})$ (2)

This sigmoid function outputs the values between 0.0 to 1.0 [3]. So the shape of sigmoid function is '*S*' shaped.

$$F(net) = 1/(1+e^{-net})$$

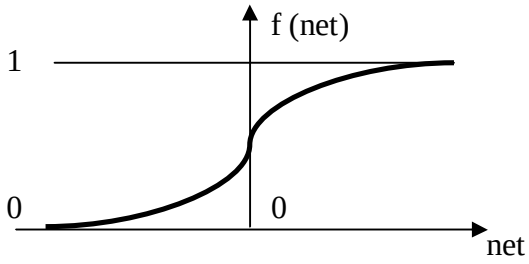

Fig.2 Sigmoid function

II. NEURAL NETWORK ADOPTION IN TOOLS

This section describe complete process an automation tool can adopt neural network. The fundamentals of testing process in which test case preparation, input values for each test case that may be a single value or in different set of values, expected output corresponding to the test case are not under this paper for tools. Here I am introducing the technique that in practical scenario automation tools can adopt that may be named as 'Neural software automation'. First of all I show the technique that how we can categorize the test cases as input neurons and how all hidden neurons can compute their calculation. Then at second step we analyse the model of this process.

A. Neural Network Model for Automation Tool

In automation tool, the developers of small and large tools can adopt this model formation to follow. This model is has adopted new mathematical transfer function that will be described in the next section. The model that is designed has its simplicity in operation and designing. Approach to design test cases and selection of input values for test cases is mentioned here as manual testing phase. The tools that will adopt or refer this idea have to be cleared that here is technique starting from set of test cases that have prepared in a softcopy sheet, it is not initializing from designing test cases. Here in the figure 3 three layer structure, that is also named multi-layered neural network (MLNN) structure is adopted that will be well suited in operation and performance. We may add more hidden layers but numbers of hidden layers are efficient for complex problems. Here in figure 3 I have explored the design for G1SSTC1 (Group 1 Sub Set's Test Cases1), same design is there for all other groups. Due to space consideration it is not fitting well there. There H1, H2...Hn are hidden layer neurons and OT refers to output layer neuron.

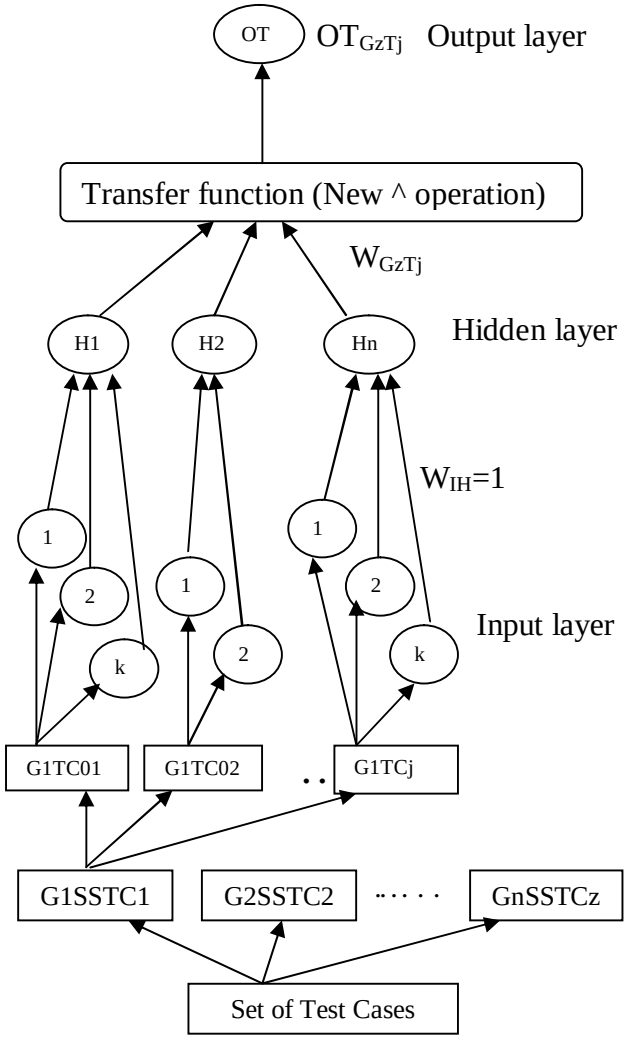

Fig. 3 Neural network model for automation testing

In the Fig. 3 it is explained that a set of all test cases is sub divided into different groups that are named with G1SSTC1, G1SSTC2..... up to G1SSTCz (Group 1 Sub Set Test Case z). Further, more normalized form of G1SSTC1 has shown as G1TC01 that is Group 1Test Case 01. After these categories the neural network is adopted and numbers of neurons are the different values supplied for a single test case. In Fig.3 W_{IH} is the weight assigned for input to hidden layer neurons. For single test case, as there are multiple inputs so hidden layer single neuron for all inputs of single test case is responsible for test case output and producing the final output for output layer. Next, there is the transfer function and I suggesting binary AND operation at there to produce output. In next section I explain how it will simplify the whole operation. In the output layer one neuron is sufficient that list the output in 0 or 1 sequentially. Weight W_{GzTj} (weight of Group z -Test case j) at hidden neuron to transfer function. The output is simply stored in order of sequence $OT_{GzTj.}$

B. Conceptual Fundamentals of Model

In this section the formulation of neural network that is described in Fig. 3 is explored. For the simplicity and error free process in this model, I have proposed the design with parallelism characteristic of the neural network. First of all the complete set of test cases is be maintained in one list. This big set of test cases with objective, input values, expected output, actual output and result are sub grouped according to the number of subgroups developer want to create. In these sub groups, test cases are in average of equal numbers. Now these subgroups have test cases and now each test case may have multiple inputs value for testing. At this point neural network adopted in automation but prior to it some preparation on whole data that will be utilized in the automation testing is must as-

- Transformation of column data.
- Desired output value will be kept as 1.
- Input values are kept as it is for test cases due to query processing at hidden layer neuron.
- The weight W_{IH}=1 is the weight desired output.

After transformation of test case data, the actual process begins. In Fig. 3 the architecture is shown for single test case of sub group 1. According to number of inputs for each test case number of neurons will be created but be careful at this initial phase, if test case has only two inputs as input type Boolean then no need to be two input neuron but at the operational process it will take less time compared to other input vales because of query processing in hidden layer. If the test case has only two input neurons then for utilization of processor create and allot input neurons to other test case. So as in its conceptual explanation there is need of flexibility in creating and allotting input neurons that is showing the connections of neurons of type self-organizing. In this model back propagation algorithm [2] will be used but updating with not to update weights but go to subsets group to follow the operation for next text case, all the weights will update only at hidden layer that are the output of hidden layer in this automation tool. As mentioned that hidden layer outputs are treated as weights of the operated current test case. The sigmoid function as the transfer function is utilized that has the limit between 0 to 1. According to the complexity and as we are not working on updating weights in training, we ignore sigmoid function because of our simple binary AND operation and fetching this output to the output layer neuron.

C. Mathematical Formulation

After selecting the test case and feeding input values to the input neurons corresponding to that test case a predefined and constant value 1 is supplied as the weight to each input to hidden link that is formulated as W_{IH}=1. This is the value that is kept as desired output for all values. In the hidden layer neuron the following operation will be performed-

- For all input values hidden layer neuron performs query operation and fetches output in 0 or 1 only.
- After getting first output the following formula is used to produce single output-

  W_{G1T1}= $W_{G1T1A1} \wedge W_{G1T1A2...} \wedge W_{G1T1Ak}$

Here this formula is for hidden neuron that is computing the output by AND operation. If any actual output becomes 0 then the output to the transfer function will be 0. Based on the number of neurons the AND operation performs calculations and produce single actual output for each test case. Now going to next level in the transfer function we use the formula-

$OT_{GzTj}= W_{GzTj} \wedge 1$
For G_z=1 to z and T_j=1 to j
OT is the final output in 0 or 1.

In the final output at the output neuron 0 will be mentioned as fail and 1 as pass. As the requirement and software testing automation tool design on small or large level one important point need to be uncover if neural network based on this given concept is designed then there are modification in back–propagation algorithm, transfer function, hidden layer neuron query processing and output in 0 or 1 only. Simplicity is best at the time of using the tool but for developers it is bit difficult to completely adopt the idea. Next section of this paper will explore more in depth about the idea. Because neural network model uses a best technique in backpropagation learning that when training on same data for testing outcomes compared to target values overwrites same results to output it automatically stops training. But we have ignored it because weights are mentioned in the algorithm to be updated to obtain target data in each training and our model not require such calculation we just want to configure test cases from groups that are unprocessed as explained earlier.

D. Important Issues

The neural network approach has been used in variety of application areas. In the practical scenario it is efficient than traditional approaches no doubt. I have shown the idea to completely neural network based software testing automation tool. This complete idea is named as INNSTA (Idea of Neural Network for Software Testing Automation). The INNSTA based tool has characteristic that selection of input neurons does not require user interaction because from transformed test cases in the sub-groups will be configured by tool at its initial process at each operational process. The Backward direction in back-propagation is just for making busy all the neurons for processor's best utilization by going back to the unprocessed test cases group and selecting for operation. The dynamic behaviour of INNSTA at starting to end of the one complete process from input to output may provide complexity in development. Idea for flexibility of creating input neurons and modified back-propagation for INNSTA based tools will be proposed later. The best use of parallelism is proposed by INNSTA for developers.

III. CONCLUSIONS

As the new function is created as transfer function it will provide flexibility to fast operation. There is another function called hard-limit transfer function used in neural networks as transfer function between hidden to output layer but there is condition that hard-limit function throws 0 if the calculated vales from hidden neuron is less than 0 or throws 1 if values are equal or greater than 0. As in this case we have 0 and 1 values, so all values will become 1 as result. In software automation testing such a tool based on neural network will reduce much of the time effort spent in. As the parallel approach of neural network, selected number of inputs simultaneously processed so speed up in efficiency for industries for testing the software will provide a beneficial boom.

E. References

[1] Dr. Hameed Nameti, Introduction to Data Mining Using Artificial Neural Networks, ISM 611.

[2] Meenakshi Vanmali, Mark Last, Abraham Kandel,"Using a Neural Network in the Software Testing Process", INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 17, 45–62 (2002), © 2002 John Wiley & Sons, Inc.

[3] Manoj Kumar Panda , "Empirical Testing Of The Neural Network Application Using Feedforward Testing Method", International Journal of Engineering Research & Technology (IJERT),IJERT Vol. 2 Issue 6, June – 2013