# MEASURING SOFTWARE COMPLEXITY USING NEURAL NETWORKS

Sibel SENAN, Selcuk SEVGEN

Department of Computer Engineering, Istanbul University, Istanbul, Turkey
{ssenan, sevgens}@istanbul.edu.tr

***Abstract:*** *Measuring the software complexity is an important task in the management of software projects. In the recent years, many researchers have paid much attention to this challenging task due to the commercial importance of software projects. In the literature, there are some software metrics and estimation models to measure the complexity of software. However, we still need to introduce novel models of software metrics to obtain more accurate results regarding software complexity. In this paper, we will show that neural networks can be used as an alternative method for estimation of software complexity metrics. We use a neural network of three layers with a single hidden layer and train this network by using distinct training algorithms to determine the accuracy of software complexity. We compare our results of software complexity obtained by using neural networks with those calculated by Halstead model. This comparison shows that the difference between our estimated results obtained by Bayesian Regularization Algorithm with 10 hidden neurons and Halstead calculated results of software complexity is less than 2%, implying the effectiveness of our proposed method of neural networks in estimating software complexity.*

***Keywords:*** *Software Complexity, Neural Networks, Halstead Metrics.*

## 1. Introduction

Software complexity analysis is a key issue essential to improve the code quality, reduce the maintenance cost, increase the robustness and meet the architecture standards. Many studies for software complexity metrics have been published in the literature [1-10]. Halstead software complexity measurement is one of the well-known and used approaches for the subject matter [11]. Halstead metrics have been developed to measure a program module's complexity directly from the source code, with the emphasis on the computational complexity. The measures have been developed for determining a quantitative measure of complexity directly from the operators and operands in the module. However, this approach is limited due to the interrelationships among all the parameters to be completely understood. Halstead complexity measurement may work appropriately for simple metrics, but when the studied models become more sophisticated, deriving metrics with equations becomes more difficult. On the other hand, neural networks approach has an ability to model a function without the need to have a knowledge of that function. The only need is to determine the endpoints (inputs and output. Unlike the traditional approach, a neural network automatically creates relationships among metric terms. In addition, neural network is a black box that needs to be well-defined for the subject matter. For instance, in [12], Halstead experiment results oscillated over the actual-calculated line indicate that the neural network was attempting to model the desired values. Thus, a precise novel model is needed to obtain more accurate results.

The objective of this paper is to evaluate the capability of neural networks in predicting software complexity and compare its prediction performance against well-known Halstead software complexity metrics in the context of randomly generated dataset. The evaluation is made by specifying the architecture of the network model to get the best results. Hence, we have compared the prediction performances of three distinct learning algorithms for two different neuron numbers of hidden layer. For this purpose, the volume (V), effort (E) and Halstead program length (H) metrics are calculated for 100 sets of operator and operand numbers of software that are randomly generated. Then, a feed-forward neural network model is designed which consists of three layers of neurons that are input, hidden, and output. The input layer consists of four nodes, and the output layer consists of three nodes. The number of hidden neurons is first chosen to be five and then to be ten for three diffeerent training algorithms (Levenberg-Marquardt (LM) Algorithm, Bayesian Regularization (BR) Algorithm, Scaled Conjugate Gradient (SCG) Algorithm) [13-19] with the purpose of obtaining the optimum result by selecting the most appropriate learning algorithm to get the best accuracy. The results of neural-network based model and Halstead model are compared to show the validity of the proposed model. This comparison shows that the difference between our estimated results obtained by Bayesian Regularization Algorithm with 10 hidden neurons and Halstead calculated

**Received on: 18.04.2017**
**Accepted on: 31.07.2017**

results of software complexity is less than 2%. The results reveal the effectiveness of neural networks in estimating software complexity.

## 2. Materials and Method

### 2.1 Halstead Complexity Measures

The Halstead measures are based on four scalar numbers derived directly from a program's source code:

| | |
|---|---|
| n1 | = the number of distinct operators |
| n2 | = the number of distinct operands |
| N1 | = the total number of operators |
| N2 | = the total number of operands |

Halstead complexity measures are derived from these numbers. Table 1. shows these Halstead metrics that are used in the study:

**Table 1.** Halstead Complexity Metrics

| Measure | Symbol | Formula |
|---|---|---|
| Program length | N | N= N1 + N2 |
| Program vocabulary | n | n= n1 + n2 |
| Volume | V | V= N *$\log_2$ (n) |
| Effort | E | E= V/2*n2 |
| Halstead Program Length | H | H = n1*$\log_2$(n1) + n2*$\log_2$(n2) |

### 2.2 Neural Networks

Neural Networks (NNs) is an eficient tool used in estimating processes. The NN structure and the training algorithm used for the application are important parameters for obtaining the optimum results. Hence, specifying these parameters for the subject of interest is a significant step in designing NN model.

In this paper, we used three distinct training algorithms (Levenberg-Marquardt(LM) Algorithm, Bayesian Regularization(BR) Algorithm, Scaled Conjugate Gradient(SCG) Algorithm) to train the network. We compared the estimating capabilities of these algorithms to get the best result. The number of hidden neurons is chosen to be 5 and 10 for each algorithm to evaluate the optimum result of the network. Input layer consists of 4 nodes, and an output layer consists of 3 nodes. Input values are n1, N1, n2 and N2 for each node. Output values are volume(V), effort(E) and Halstead program length(H) for each node. We stated the structure of the network and the appropriate training algorithm for the handled subject acording to obtained results. The architecture of the proposed NN model is shown in Fig. 1.

The NN model used in the study consists of three layers of neurons as input, hidden and output as depicted in Figure 1. The input layer of this system consists of the number images which are represented by matrices. In the training process of this type of network, the connection weights are updated to minimize the error between the correct and estimated values of the system variables [20].

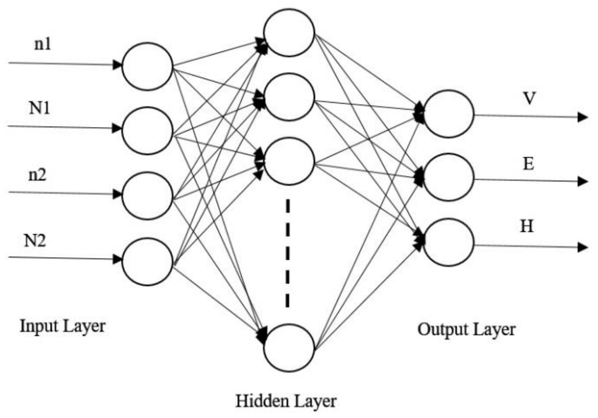

**Figure 1.** NN structure used in this study

A hidden or output unit in the NN operates as follows :

$$y_j = f\left(\sum_i w_{ji} x_i + b_j\right) \qquad (1)$$

where
*i= 1,2,3,4* and *j=1,2,3.*
$y_j$ is transformed output by the *j*th hidden or output node,
$f(\cdot)$ is an activation function,
$w_{ji}$ is a synaptic weight from the *i*th node to *j*th node,
$x_i$ is an input node,
$b_j$ is a bias at *j*th node.

### 2.3 Dataset

Dataset used in this study is generated randomly according to the following assumptions:

$$\begin{gathered} 2 \le n1 \le 50 \\ 2 * n1 \le N1 \le 5 * n1 \\ 2 \le n2 \le 50 \\ 2{,}5 * n2 \le N2 \le 6 * n2 \end{gathered} \qquad (2)$$

Table 2 shows the input values as n1, n2, N1, N2 and the calculated results as Halstead metrics V, E, H for 100 randomly generated sets.

## 3. Results

*System training:* In the designed NN, there are 4 inputs and 3 outputs for each 100 sets. It should be noted that, the proposed model can be applied to set of any length.

To obtain the best result we evaluated 3 distinct training algorithms (LM training alg., BR training alg., SCG training alg.) and compared the predicting capabilities of these algorithms for two different hidden neuron numbers. The number of neurons in the hidden layer is taken account as 5 and 10 for each algorithm to analyse the results. The input values are used as 70% for training, 15% for validation and 15% for test. Table 3 shows the %-average differences for the results of our models and the Halstead model.

**Table 3.** % Average Differences

| Result Metrics | LM 5neurons | LM 10neuron | BR 5neurons | BR 10neurons | SCG 5neurons | SCG 10neurons |
|---|---|---|---|---|---|---|
| **V** | 1,66 | 0,50 | 0,39 | **0,09** | 3,51 | 6,41 |
| **E** | 25,19 | 4,16 | 3,45 | **1,08** | 42,61 | 253,18 |
| **H** | 5,03 | 1,33 | 1,09 | **0,36** | 10,36 | 254,53 |

LM: Levenberg-Marquardt algorithm
BR: Bayesian Regularization algorithm
SCG: Scaled Conjugate Gradient algorithm

Among these models, the best solution is obtained by the network with 10 hidden neurons under BR training algorithm for each output value and the worst solution is obtained by the network with 10 hidden neurons under SCG training algorithm for each output value.

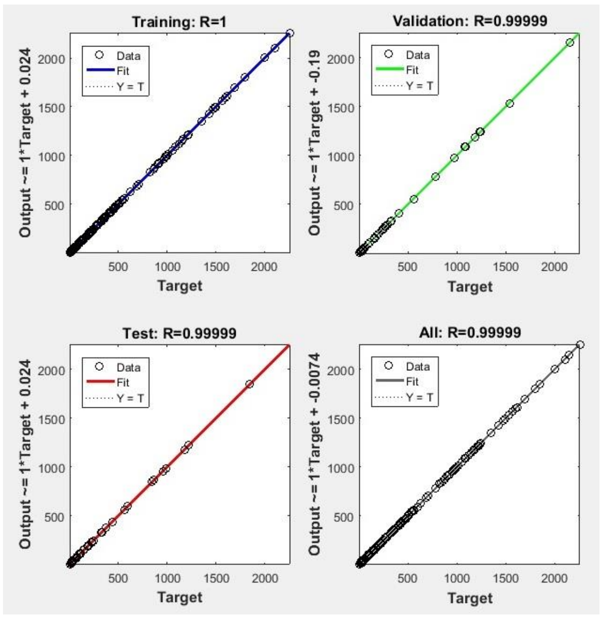

**Figure 2.** The regression coefficients of NN model with 10 hidden nodes under BR Algorithm

Figure 2. shows the regression graphics for training, validation, test and all for B-R algorithm. The regression values are obtained very close to 1 which is a desired case for modeling by NNs. Figures 3, 4 and 5 show the obtained results as bar chart histogram for V, E and H metrics with B-R algorithm and 10 neuron NN structure.

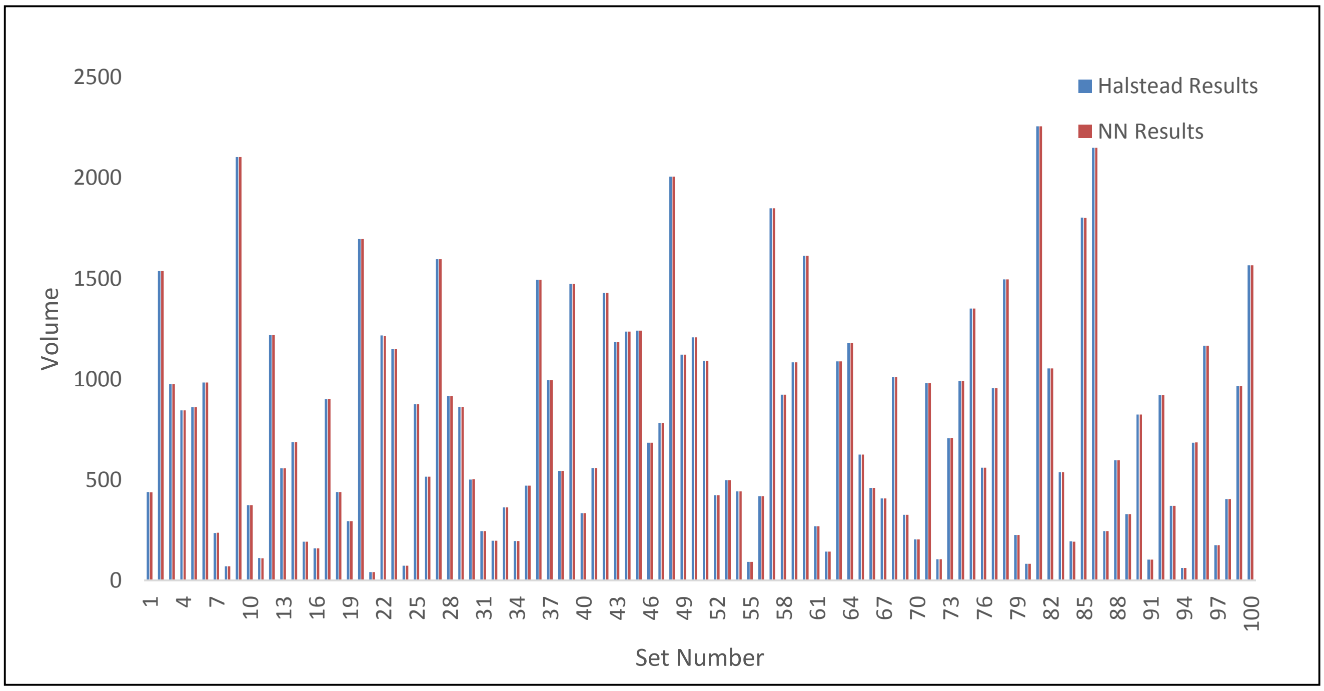

**Figure 3.** The results for Volume Metric

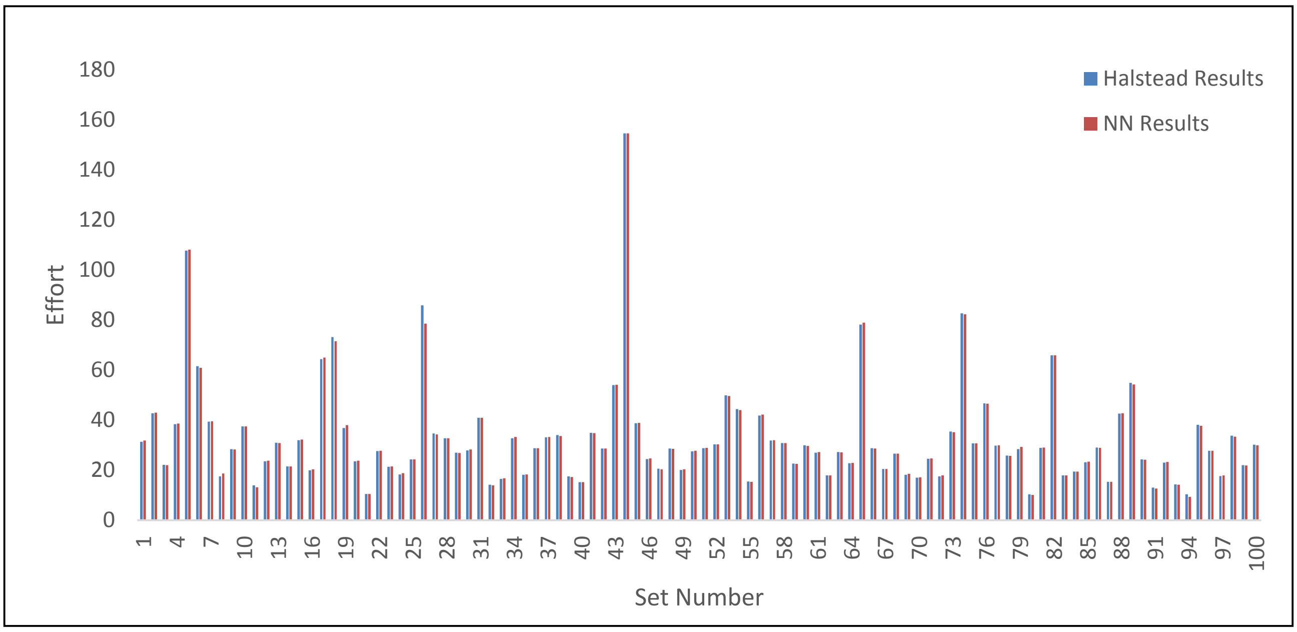

**Figure 4.** The results for Effort Metric

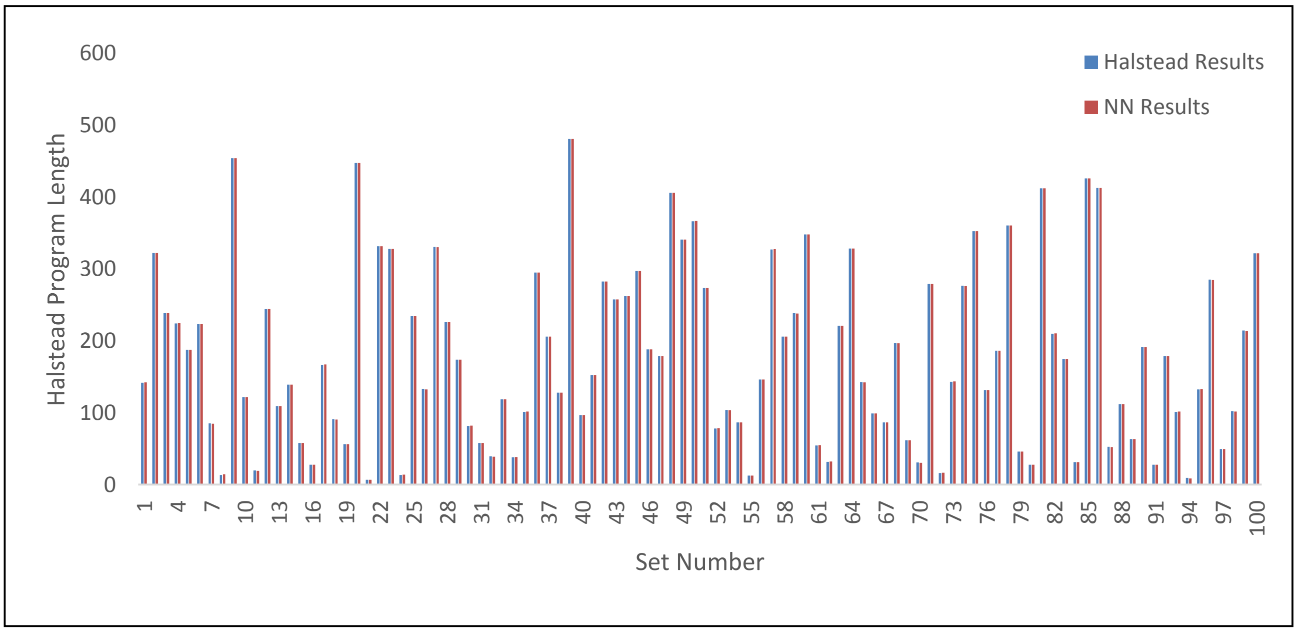

**Figure 5.** The results for Halstead Program Length Metric

**Table 2.** % Average Differences

| Inputs | | | | | | | | | | | Calculated Outputs | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **No** | **n1** | **N1** | **n2** | **N2** | **No** | **n1** | **N1** | **n2** | **N2** | | **No** | **V** | **E** | **H** | **No** | **V** | **E** | **H** |
| **1** | 26 | 59 | 7 | 28 | **51** | 37 | 92 | 19 | 96 | | **1** | 438,86 | 31,35 | 141,86 | **51** | 1091,78 | 28,73 | 273,46 |
| **2** | 45 | 164 | 18 | 93 | **52** | 15 | 68 | 7 | 27 | | **2** | 1536,16 | 42,67 | 322,19 | **52** | 423,65 | 30,26 | 78,25 |
| **3** | 29 | 75 | 22 | 97 | **53** | 21 | 84 | 5 | 22 | | **3** | 975,66 | 22,17 | 238,99 | **53** | 498,25 | 49,82 | 103,85 |
| **4** | 36 | 116 | 11 | 36 | **54** | 18 | 69 | 5 | 29 | | **4** | 844,30 | 38,38 | 224,17 | **54** | 443,31 | 44,33 | 86,67 |
| **5** | 35 | 141 | 4 | 22 | **55** | 4 | 17 | 3 | 16 | | **5** | 861,52 | 107,69 | 187,52 | **55** | 92,64 | 15,44 | 12,75 |
| **6** | 38 | 141 | 8 | 37 | **56** | 28 | 66 | 5 | 17 | | **6** | 983,19 | 61,45 | 223,42 | **56** | 418,68 | 41,87 | 146,22 |
| **7** | 19 | 39 | 3 | 14 | **57** | 36 | 140 | 29 | 167 | | **7** | 236,35 | 39,39 | 85,47 | **57** | 1848,87 | 31,88 | 327,00 |
| **8** | 5 | 19 | 2 | 6 | **58** | 30 | 101 | 15 | 67 | | **8** | 70,18 | 17,55 | 13,61 | **58** | 922,63 | 30,75 | 205,81 |
| **9** | 47 | 174 | 37 | 155 | **59** | 27 | 62 | 24 | 129 | | **9** | 2103,07 | 28,42 | 453,82 | **59** | 1083,43 | 22,57 | 238,42 |
| **10** | 24 | 55 | 5 | 22 | **60** | 41 | 189 | 27 | 76 | | **10** | 374,06 | 37,41 | 121,65 | **60** | 1613,18 | 29,87 | 348,04 |
| **11** | 5 | 20 | 4 | 15 | **61** | 12 | 49 | 5 | 17 | | **11** | 110,95 | 13,87 | 19,61 | **61** | 269,77 | 26,98 | 54,63 |
| **12** | 26 | 90 | 26 | 124 | **62** | 8 | 23 | 4 | 17 | | **12** | 1219,89 | 23,46 | 244,42 | **62** | 143,40 | 17,92 | 32,00 |
| **13** | 19 | 75 | 9 | 41 | **63** | 28 | 140 | 20 | 55 | | **13** | 557,65 | 30,98 | 109,24 | **63** | 1089,07 | 27,23 | 221,04 |
| **14** | 18 | 68 | 16 | 67 | **64** | 39 | 102 | 26 | 94 | | **14** | 686,81 | 21,46 | 139,06 | **64** | 1180,38 | 22,70 | 328,34 |
| **15** | 14 | 30 | 3 | 17 | **65** | 28 | 115 | 4 | 10 | | **15** | 192,11 | 32,02 | 58,06 | **65** | 625,00 | 78,13 | 142,61 |
| **16** | 7 | 35 | 4 | 11 | **66** | 18 | 62 | 8 | 36 | | **16** | 159,13 | 19,89 | 27,65 | **66** | 460,64 | 28,79 | 99,06 |
| **17** | 30 | 132 | 7 | 41 | **67** | 14 | 51 | 10 | 38 | | **17** | 901,24 | 64,37 | 166,86 | **67** | 408,06 | 20,40 | 86,52 |
| **18** | 20 | 87 | 3 | 10 | **68** | 25 | 107 | 19 | 78 | | **18** | 438,79 | 73,13 | 91,19 | **68** | 1009,99 | 26,58 | 196,81 |
| **19** | 13 | 61 | 4 | 11 | **69** | 10 | 31 | 9 | 46 | | **19** | 294,30 | 36,79 | 56,11 | **69** | 327,09 | 18,17 | 61,75 |
| **20** | 47 | 144 | 36 | 122 | **70** | 6 | 30 | 6 | 27 | | **20** | 1695,76 | 23,55 | 447,18 | **70** | 204,34 | 17,03 | 31,02 |
| **21** | 3 | 9 | 2 | 9 | **71** | 37 | 83 | 20 | 85 | | **21** | 41,79 | 10,45 | 6,75 | **71** | 979,93 | 24,50 | 279,19 |
| **22** | 43 | 123 | 22 | 79 | **72** | 5 | 23 | 3 | 12 | | **22** | 1216,52 | 27,65 | 331,44 | **72** | 105,00 | 17,50 | 16,36 |
| **23** | 38 | 89 | 27 | 102 | **73** | 24 | 84 | 10 | 55 | | **23** | 1150,27 | 21,30 | 327,80 | **73** | 707,16 | 35,36 | 143,26 |
| **24** | 5 | 15 | 2 | 11 | **74** | 47 | 155 | 6 | 18 | | **24** | 72,99 | 18,25 | 13,61 | **74** | 990,93 | 82,58 | 276,58 |
| **25** | 32 | 85 | 18 | 70 | **75** | 46 | 121 | 22 | 101 | | **25** | 874,80 | 24,30 | 235,06 | **75** | 1351,42 | 30,71 | 352,19 |
| **26** | 27 | 87 | 3 | 18 | **76** | 25 | 98 | 6 | 15 | | **26** | 515,22 | 85,87 | 133,14 | **76** | 559,82 | 46,65 | 131,61 |
| **27** | 42 | 185 | 23 | 80 | **77** | 26 | 104 | 16 | 73 | | **27** | 1595,93 | 34,69 | 330,52 | **77** | 954,44 | 29,83 | 186,21 |
| **28** | 34 | 127 | 14 | 37 | **78** | 41 | 101 | 29 | 143 | | **28** | 915,93 | 32,71 | 226,28 | **78** | 1495,55 | 25,79 | 360,54 |
| **29** | 24 | 101 | 16 | 61 | **79** | 11 | 48 | 4 | 10 | | **29** | 862,15 | 26,94 | 174,04 | **79** | 226,60 | 28,32 | 46,05 |
| **30** | 14 | 70 | 9 | 41 | **80** | 7 | 14 | 4 | 10 | | **30** | 502,12 | 27,90 | 81,83 | **80** | 83,03 | 10,38 | 27,65 |
| **31** | 14 | 49 | 3 | 11 | **81** | 39 | 128 | 39 | 231 | | **31** | 245,25 | 40,87 | 58,06 | **81** | 2256,46 | 28,93 | 412,26 |
| **32** | 7 | 28 | 7 | 24 | **82** | 36 | 145 | 8 | 48 | | **32** | 197,98 | 14,14 | 39,30 | **82** | 1053,67 | 65,85 | 210,12 |
| **33** | 19 | 43 | 11 | 31 | **83** | 25 | 62 | 15 | 39 | | **33** | 363,11 | 16,50 | 118,76 | **83** | 537,51 | 17,92 | 174,70 |
| **34** | 10 | 39 | 3 | 14 | **84** | 7 | 32 | 5 | 22 | | **34** | 196,12 | 32,69 | 37,97 | **84** | 193,59 | 19,36 | 31,26 |
| **35** | 14 | 42 | 13 | 57 | **85** | 41 | 154 | 39 | 131 | | **35** | 470,73 | 18,11 | 101,41 | **85** | 1801,75 | 23,10 | 425,79 |
| **36** | 34 | 122 | 26 | 131 | **86** | 41 | 123 | 37 | 219 | | **36** | 1494,44 | 28,74 | 295,19 | **86** | 2149,61 | 29,05 | 412,41 |
| **37** | 30 | 112 | 15 | 69 | **87** | 9 | 31 | 8 | 29 | | **37** | 994,03 | 33,13 | 205,81 | **87** | 245,25 | 15,33 | 52,53 |
| **38** | 23 | 72 | 8 | 38 | **88** | 21 | 104 | 7 | 20 | | **38** | 544,96 | 34,06 | 128,04 | **88** | 596,11 | 42,58 | 111,89 |
| **39** | 46 | 104 | 42 | 124 | **89** | 15 | 64 | 3 | 15 | | **39** | 1472,75 | 17,53 | 480,56 | **89** | 329,42 | 54,90 | 63,36 |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **40** | 15 | 33 | 11 | 38 | **90** | 26 | 55 | 17 | 97 | | **40** | 333,73 | 15,17 | 96,66 | **90** | 824,79 | 24,26 | 191,70 |
| **41** | 27 | 70 | 8 | 39 | **91** | 7 | 17 | 4 | 13 | | **41** | 559,09 | 34,94 | 152,38 | **91** | 103,78 | 12,97 | 27,65 |
| **42** | 33 | 112 | 25 | 132 | **92** | 21 | 61 | 20 | 111 | | **42** | 1429,35 | 28,59 | 282,56 | **92** | 921,50 | 23,04 | 178,68 |
| **43** | 41 | 164 | 11 | 44 | **93** | 14 | 31 | 13 | 47 | | **43** | 1185,69 | 53,90 | 257,71 | **93** | 370,88 | 14,26 | 101,41 |
| **44** | 46 | 195 | 4 | 24 | **94** | 3 | 15 | 3 | 9 | | **44** | 1236,00 | 154,50 | 262,08 | **94** | 62,04 | 10,34 | 9,51 |
| **45** | 43 | 169 | 16 | 42 | **95** | 23 | 87 | 9 | 50 | | **45** | 1241,24 | 38,79 | 297,33 | **95** | 685,00 | 38,06 | 132,57 |
| **46** | 28 | 56 | 14 | 71 | **96** | 37 | 125 | 21 | 74 | | **46** | 684,82 | 24,46 | 187,91 | **96** | 1165,74 | 27,76 | 284,99 |
| **47** | 22 | 89 | 19 | 57 | **97** | 11 | 30 | 5 | 14 | | **47** | 782,20 | 20,58 | 178,82 | **97** | 176,00 | 17,60 | 49,66 |
| **48** | 42 | 139 | 35 | 181 | **98** | 20 | 60 | 6 | 26 | | **48** | 2005,37 | 28,65 | 406,00 | **98** | 404,24 | 33,69 | 101,95 |
| **49** | 39 | 108 | 28 | 77 | **99** | 25 | 83 | 22 | 91 | | **49** | 1122,23 | 20,04 | 340,74 | **99** | 966,50 | 21,97 | 214,20 |
| **50** | 48 | 130 | 22 | 67 | **100** | 38 | 150 | 26 | 111 | | **50** | 1207,47 | 27,44 | 366,19 | **100** | 1566,00 | 30,12 | 321,63 |

## 4. Conclusions

In this study, we have shown that neural networks can be used as an alternative method for estimation of software complexity metrics. We have used a neural network of three layers with a single hidden layer and trained this network by using distinct training algorithms to determine the accuracy of software complexity. We have compared our results of software complexity obtained by using neural networks with those calculated by Halstead model. This comparison has shown that the difference between our estimated results obtained by Bayesian Regularization Algorithm with 10 hidden neurons and Halstead calculated results of software complexity is less than 2%, implying the effectiveness of our proposed method of neural networks in estimating software complexity.

## 5. References

[1] H. Zuse, “Software Complexity: Measures and Methods”, Walter de Gruyter, 1991.

[2] M. M. Lehmam and L. A. Belady, “Program Evolution - Processes of Software Change”, Academic Press Professional, 1985.

[3] H. F. Li and W. K. Cheung, “An Empirical Study of Software Metrics,” *IEEE Transactions on Software Engineering,* 13, 6, pp. 697-708, 1987.

[4] P. Oman and C. Cook, “The Book Paradigm for Improved Software Maintenance”, *IEEE Software*, 7, 1, pp. 39-45, 1990.

[5] H. Zuse, “A Framework of Software Measurement”, De Gruyter Publisher, 1998.

[6] C. Jones, "Software Metrics: Good, Bad, and Missing." *Computer*, *27*, *9*, pp. 98-100, 1994.

[7] J. Marciniak, “Encyclopedia of Software Engineering”, John Wiley & Sons, 1994.

[8] P. Oman, “HP-MAS: A Tool for Software Maintainability, Software Engineering”, (#91-08-TR), Moscow, ID: Test Laboratory, University of Idaho, 1991.

[9] P. Oman and J. Hagemeister, "Constructing and Testing of Polynomials Predicting Software Maintainability." *Journal of Systems and Software*, 24, 3, pp. 251-266, 1994.

[10] P: Szulewski, “Automating Software Design Metrics”, (RADC-TR-84-27), Rome, NY: Rome Air Development Center, 1984.

[11] M. H. Halstead, “Elements of Software Science, Operating, and Programming Systems Series”, 1977.

[12] G. Boetticher, K. Srinivas and D. Eichmann, “A Neural Net-Based Approach to Software Metrics”, *Proceedings of the Fifth International Conference on Software Engineering and Knowledge Engineering*, pp. 271-274, 1993.

[13] K. Levenberg, “A method for the solution of certain problems in least squares”, *Quarterly of Applied Mathematics*, 5, pp. 164-168, 1944.

[14] D. Marquardt, “An algorithm for least-squares estimation of nonlinear parameters”, *SIAM Journal on Applied Mathematics*, 11, pp. 431-441, 1963.

[15] H. Yuand, B.M. Wilamowski, “Intelligent Systems, Ch.12, Levenberg-Marquard Training”, CRC Press, 2011.

[16] M. T. Hagan and M. Menhaj, "Training feed-forward networks with the Marquardt algorithm," *IEEE Transactions on Neural Networks*, 5, 6, pp. 989–993, 1994.

[17] M. F. Møller, “A scaled conjugate gradient algorithm for fast supervised learning”, *Neural Networks*, 6, 4, pp. 525-533, 1993.

[18] B. M. Wilamowski, "Neural network architectures and learning algorithms," *Industrial Electronics Magazine, IEEE*, 3, 4, pp. 56-63, 2009.

[19] M. Avriel, “Nonlinear Programming: Analysis and Methods”, Dover Publishing, 2003.

[20] S. Haykin, “Neural Networks and Learning Machines” (3rd ed.). Prentice Hall.

**Sibel Senan** is currently an Assistant Professor at the Dept. of Computer Engineering in Istanbul University. She received her M.Sc. and Ph.D. degrees at the same department in 2005 and in 2010, respectively. Her main interests are Neural Networks, Nonlinear Systems

**Selcuk Sevgen** is currently an Assistant Professor at the Dept. of Computer Engineering in Istanbul University. He received his M.Sc. and Ph.D. degrees at the same department in 2003 and in 2009, respectively. His main interests are Neural Networks, CNNs.