Quest Journals
Journal of Software Engineering and Simulation
Volume 8 ~ Issue 7 (2022) pp: 23-38
ISSN(Online) :2321-3795 ISSN (Print):2321-3809
www.questjournals.org

Research Paper

A Comparative Analysis Of Adaptive Neuro-Fuzzy Inference System Back Propagation And Hybrid Learning For Software Development Effort Evaluation.

Edward N. Udo[1*], Okure U. Obot[1] and Peter G. Obike[2]

[1]*Department of Computer Science, Faculty of Science, University of Uyo, Uyo, Nigeria*
[2] *Department of Computer Science, College of Physical and Applied Sciences, Michael Okpara University of Agriculture, Umudike, Nigeria.*
**Corresponding Author: Edward N. Udo*

ABSTRACT: *Finding accurately the amount of effort needed to develop any software is quite essential in software project management because inaccurate estimates affect both the software resources and the deliverables. A number of models have been developed for the estimation of the amount of effort required to develop a software. One of such models is the Constructive Cost Model (COCOMO), which was found to contain uncertain and imprecise inputs. To improve the COCOMO, an Adaptive Neuro-Fuzzy Inference System (ANFIS) was built by training the data acquired from PROMISE repository – NASA COCOMO with ANFIS back propagation and hybrid algorithm using MATLAB R 2017. This was done after Principal Component Analysis (PCA) was employed to reduce the dimensionality of the datasets from 23 to 6. With 6 attributes, the datasets were trained on the back propagation learning rule and the hybrid learning rule. It was revealed that the hybrid learns faster from epoch 70 and above. The performance of the outputs of the ANFIS algorithms was evaluated using Mean Magnitude Relative Error (MMRE) and Root Mean Square Error (RMSE). The results show that the Hybrid model has the least MMRE and RMSE of 31.4829 and 0.00000187 respectively, which is a confirmation that ANFIS hybrid learning performs better than ANFIS Back Propagation methods in terms of training convergence, reduced error and performance.*

KEYWORDS*: Software Development Effort, ANFIS Back Propagation, Hybrid Learning Algorithm, COCOMO*

R*eceived 25 June, 2022;* ***R****evised 05 July, 2022;* ***A****ccepted 07 July, 2022 © The author(s) 2022.*
P*ublished with open access at* ***www.questjournals.org***

I. INTRODUCTION

In software project management, cost estimation is used to measure the success of any software project [1]. Software development cost estimation guides in the prediction of the likely amount of effort, time and staffing required in building a software system [2] and is one of the most important activities, with a critical role, in software project management process [3].

Software development effort estimation is defined as a set of tasks that should be performed in order to derive some estimates, which are usually expressed in terms of hours and money [4]. It is the process of finding out the exact effort required to develop or maintain a software [5].

Estimating the cost or the effort in person-months required in developing any software project accurately is very essential at the early stages of the software development life cycle (SDLC) [6]. This accurate prediction helps investors in software projects and their customers to know the total investment needed for the project as well as the project schedule. This is the reason many organizations that are involved in software projects usually estimate the resources, effort and time needed to satisfactorily complete the project and hand it over to the client as scheduled [7].

With inaccurate estimation figures, it becomes almost impossible to thoroughly plan, monitor and control a software project [8] and this can have adverse consequences on project resources [6]. According to the result of the research conducted by the International Society of Parametric Analysis (ISPA) [9] and the Standish Group International [10]**,** two thirds of software projects are not delivered on time and within budget; this is due

to incorrect estimation (in terms of project size, cost, needed staff etc) and inappropriate software and system requirements [6].

An accurate estimate of the software effort, cost and schedule is very important in managing financial issues and monitoring all the developmental activities and on-time delivery [1]. The ability to predict the cost or effort of a software project has a direct impact on the management decision as either to accept or reject the proposal to venture into such a project [3]. Accurate estimation of software resources is very challenging and as such many techniques have been investigated and proposed in the past decades with the aim of improving the accuracy of software estimation models [11], support project managers to elaborate budget, forecast iteration and define project plans [12] and also support software developers in performing several of their software development tasks.

The techniques used in software effort estimation are categorized into three main groups: expert judgment, algorithmic models, and machine learning models [13]. In expert judgment group, the software project effort estimation depends on the expertise and experience of the estimator [14]. The expertise of the estimator is based on the problem domain the estimator is familiar with as well as the similarity index of the software projects in question with the estimator's familiar historical projects.

Algorithmic (parametric) models group use mathematical equations to predict software cost. These models are derived from statistical or numerical analysis of historical software project data [8]. Constructive Cost Model (COCOMO) is a parametric model; software size is the main input of these models. Linear and non-linear regression equations can be used in parametric models [6]. The algorithmic estimation models are very simple and reliable but not so accurate. The categorical datasets cannot be estimated using parametric models [5]. Algorithmic models require inputs that are estimated accurately and with specific attributes. These inputs are not easy to get at the early stages of SDLC. Algorithmic models also have issues in predicting accurately the development cost and effort; the accuracy of such estimates is always very low due to availability of limited information at the early stages of software development. These limitations of the algorithmic models led to the exploration of the non algorithmic techniques, visualized through soft computing approaches [2].

Machine learning (nonparametric and non algorithmic) models group are based on nonlinear characteristics [11]; set of artificial intelligence techniques, such as artificial neural networks, genetic algorithms, regression trees, rule-based induction, analogy-based or case-based reasoning (CBR), and fuzzy logic etc [8]. Machine learning techniques have gained popularity in cost estimation of predictive models [3]. These models can be standalone models or models that work in conjunction with algorithmic models. Some non-standalone works include: [15 -16] predicted software effort using Naïve Bayes, Logistic Regression and Random Forests; [17] investigated software development effort estimation using polynomial linear regression, ridge regression, decision trees, support vector regression, and Multilayer Perceptron.

Unfortunately, the industry is plagued with unreliable estimates, and no effort estimation model has proven to be consistently successful at predicting software project effort in all situations [8]. Even with these several software development effort estimation models in place and their crucial impact on budgeting and project planning in the industry, software development effort estimation is still an open question in the field of software engineering [12]. This means that practitioners and researchers are yet to have an effective and widely acceptable approach for estimation and evaluation of software development effort; they still need to choose any approach that matches their domain and researches needs respectively. The adoption of an approach that does not fit their needs becomes questionable in realistic scenarios [12].

This work therefore compares software development effort models: COCOMO, Adaptive Neuro Fuzzy Inference System (ANFIS) Back Propagation and ANFIS Hybrid with 6 input parameters instead of traditional 23 inputs used in literature. This comparison is yet to be done in literature. Datasets for COCOMO model was collected from Promise Repository and the number of input parameters in the dataset reduced using Principal Component Analysis (PCA). Fuzzy inference system was generated and trained using ANFIS Back Propagation methods and hybrid learning algorithms. The performance was evaluated using MMRE and RMSE. In Section 2, related work, fuzzy logic model and ANFIS pseudo-code are presented. Section 3 presents the attributes used for software development effort prediction while section 4 discusses dimension reduction using Principal Component Analysis (PCA). Section 5 and 6 considers ANFIS training and performance evaluation respectively. The work is concluded in section 7.

II. RELATED WORK

COCOMO proposed a nonlinear relationship between the size of the project and estimated effort measured in person months [18]. With the advent of component based development and service oriented architecture, COCOMO model was improved to COCOMO II to include changes evidenced around code reuse, function points and other related changes. COCOMO II was also formulated with a nonlinear relationship between the size of the project and estimated effort.

ANFIS is a framework of a neuro-fuzzy model which adapts itself through learning. It is used to identify systems based on available data and its result is significant when used in modeling non-linear functions [19 – 22]. ANFIS model has the advantage of having both numerical and linguistic knowledge [23].

There are several COCOMO and ANFIS based software effort development models using NASA datasets in literature. [24] proposed the use of ANFIS for software effort estimation using COCOMO II datasets. The ANFIS was modeled for several type of membership functions like Gaussian curve, Difference of sigmoidal membership, Gaussian combination membership, Trapezoidal membership, Triangular membership functions etc. [25] designed an efficient software effort estimation model using ANFIS on COCOMO datasets and tested it to see how it performed. [1] used a Neuro-fuzzy model optimized with PSO to derive an improved software development effort estimation model using NASA dataset software project. The results of the optimization were trained using ANFIS to get an effort prediction. [26] carried out neuro-fuzzy computing through ANFIS using COCOMO and COCOMO II datasets by comparing the expected and the actual data. The results showed that ANFIS model can be efficiently used for estimating software development effort.

ANFIS have been combined with other algorithms/models to estimate software development effort. Such models include Decision Tree and Artificial Neural Network [27], Multilayer Perceptron [28], Hybrid Particle Swarm Optimization [1], Naïve Bayes, Logistic Regression and Random Forests [16], Differential Evolution Algorithm [29], Functional Point Analysis [30]. It is also important to compare the two variants of ANFIS; back propagation and hybrid, which is the aim of this work.

a. Fuzzy Logic Model for Software Development Effort Evaluation

The fuzzy logic model for software development effort evaluation, shown in Figure 1, contains a knowledge base and processing stage. The knowledge base provides membership functions and the fuzzy rules needed for the process. Numerical crisp variables are the input of the system in the processing stage. These variables are passed through a fuzzification stage where they are transformed into linguistic variables, which become the fuzzy input of the inference engine. This fuzzy input is transformed by rules of the inference engine to fuzzy output. These linguistic results are then changed by a defuzzification stage into numerical values that becomes the output of the system.

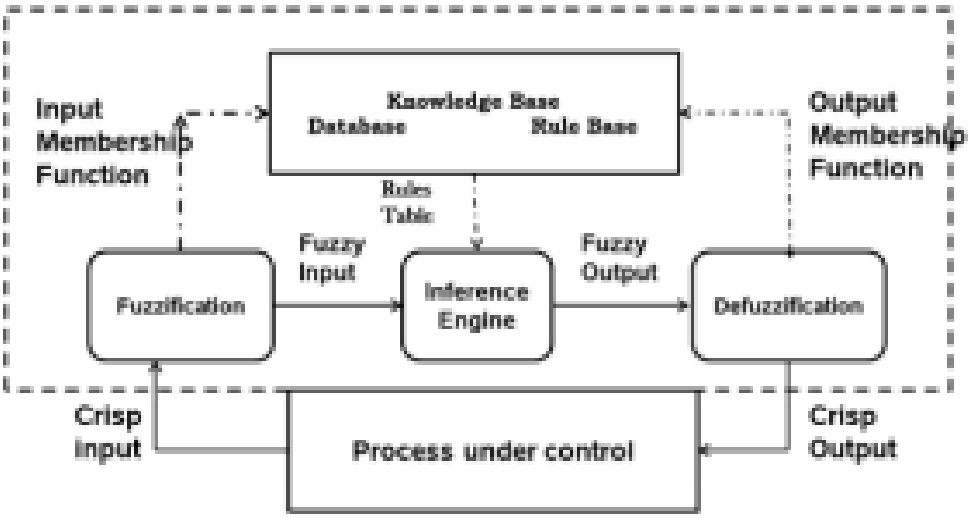

Figure 1: Fuzzy Logic Model for Software Development Effort Evaluation

b. Pseudo-Code of ANFIS Evaluation

Begin:

Step I: Determine the inputs of the model;

Collect datasets;
Divide the datasets into: Training and Testing (for evaluating the validity of the estimated model).
Step II: Generate ANFIS model;

[Define Number of Membership functions] numMFs;

[define type of Membership functions] mfType;
[define Number of epoch] epoch_n;
[Generate Fuzzy Inference System structure from data using grid partition] in_fis=genfis1 (trnData, numMFs, mfType);
[Training routine for Sugeno-type Fuzzy Inference System (uses a hybrid learning algorithm)] out_fis= anfis (trnData,in_fis,epoch_n);

Step III: Evaluate the value of Development Time;

For each individual test data
For i=1 to total test data
[Evaluate the value of Development Time]
dt(i)= evalfis(inpData,fis);
Next i;

Step IV: Evaluate the Value of MRE and MSE from result obtained in step III;

For MRE and MSE of each individual test data
For i = 1: to total test data
mre(1,i)=abs((Actual dt(i)-dt(i))/Actual
dt(i));
Next i;

Step V: Evaluate the Value of MMRE and PRED from result obtained in step IV;

For MMRE and PRED of each individual test data
Initialize mmre =0, pred =0;
For i = 1: to total test data
mmre = mmre+mre(i);
IF (mre (i) <=.25)
pred =pred+1;
EndIF
Next i;
MMRE= (mmre/ (total test data))*100;
PRED=pred/ (total test data);

End:

III. ATTRIBUTES USED FOR SOFTWARE DEVELOPMENT EFFORT PREDICTION

The attributes are divided into NASA, COCOMO and modes of development constraints. They are described in Table 1 – 3.

Table 1: NASA based attributes

NASA ATTRIBUTES	DESCRIPTION
Uniqueid	Defined as numbers to identify each unique project. The numbers are not continuous since the records are a subset of another NASA database.
Forg	States whether it is a flight or ground system
Center	Used to define the NASA center. Examples include 1, 2,3, 4, 5, 6
year	Defines the year of development
Mode	Defines the development mode, whether it is embedded, organic or semidetached

Table 2: COCOMO attributes

COCOMO ATTRIBUTES	DESCRIPTIONS
RELY	Defined as the reliability of the system
DATA	Defined as the database size
CPLX	Measures Process complexity
TIME	Measures Time constraint for CPU
STOR	Measures Memory constraint
VIRT	Measures Machine volatility

TURN	Measures Turnaround time
ACAP	Measures analysts capability
AEXP	Measures application experience
PCAP	Defines programmer's capability
VEXP	Measures virtual machine experience of the developers
LEXP	Measures language experience of the programmers
MODP	Measures modern programing practices of the programmer
TOOL	Measures the programmer's use of software tools
SCED	Measures schedule constraint

Table 3: Modes of Development Constants

Mode	A	B	C	D
Organic	2.4	1.05	2.5	0.38
Semi-detached	3	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Table 4 shows the ratings (numeric values) of the COCOMO effort attributes (multipliers) and their correlation with more effort.

Table 4: COCOMO Attributes: Ratings and Correlation

Effort Attributes	**Ratings(Numeric Values)**						**Correlation with more Effort**
	Very Low	**Low**	**Nominal**	**High**	**Very High**	**Extra High**	
ACAP	1.46	1.19	1.00	0.86	0.71		Positive
PCAP	1.42	1.17	1.00	0.86	0.70		Positive
AEXP	1.29	1.13	1.00	0.91	0.82		Positive
MODP	1.24	1.10	1.00	0.91	0.82		Positive
TOOL	1.24	1.10	1.00	0.91	0.83		Positive
VEXP	1.21	1.10	1.00	0.90			Positive
LEXP	1.14	1.07	1.00	0.95			Positive
SCED	1.23	1.08	1.00	1,04	1.10		No Correlation
STOR			1.00	1.06	1.21	1.56	Negative
DATA		0.94	1.00	1.08	1.16		Negative
TIME			1.00	1.11	1.30	1.66	Negative
TURN		0.87	1.00	1.07	1.15		Negative
VIRT		0.87	1.00	1.15	1.30		Negative
RELY	0.75	0.88	1.00	1.15	1.40		Negative
CPLX	0.70	0.85	1.00	1.15	1.30	1.65	Negative

The COCOMO software cost model measures effort in calendar months of 152 hours (development and management hours included). COCOMO assumes that the effort grows more than linearly on software size; that is:

$$person\ per\ months\ = A * Size^{\sum ScaleFactor} * \sum Effort\ Multiplier \quad (1)$$

Size is the estimated size in KLOC, scale factor is the combined process factors, and Effort multiplier is the combined effort factors. The constant and the scale factor are domain-specific parameters and Effort Multiplier is the product of over a dozen effort multipliers.

IV. DIMENSION REDUCTION

Principal Component Analysis (PCA) was used to reduce the dimension of COCOMO 11 NASA dataset from 23-input feature vector to a 6-input feature vector. For design's sake, the first two data points are used for reduction with only 15 inputs considered to aid in flawless computation. The mathematical principle behind dimension reduction is to find a vector that defines the surface to which we wish to project the data.
The first eleven data point in the dataset used to illustrate the workings behind PCA is given in Table 5.

Table 5: 15 COCOMO II sample for PCA

1	1.15	0.94	1.15	1.00	1.00	0.87	0.87	1.00	1.00	1.00	1.00	0.95	0.91	1.00	1.08
2	1.15	0.94	1.15	1.00	1.00	0.87	0.87	1.00	1.00	1.00	1.00	0.95	0.91	1.00	1.08
3	1.15	0.94	1.15	1.00	1.00	0.87	0.87	1.00	1.00	1.00	1.00	0.95	0.91	1.00	1.08
4	1.15	0.94	1.15	1.00	1.00	0.87	0.87	1.00	1.00	1.00	1.00	0.95	0.91	1.00	1.08
5	1.15	0.94	1.15	1.00	1.00	0.87	0.87	1.00	1.00	1.00	1.00	0.95	0.91	1.00	1.08
6	1.15	0.94	1.15	1.00	1.00	0.87	0.87	1.00	1.00	1.00	1.00	0.95	0.91	1.00	1.08
7	1.15	0.94	1.15	1.00	1.00	0.87	0.87	1.00	1.00	1.00	1.00	0.95	0.91	1.00	1.08

8	1.15	0.94	1.15	1.00	1.00	0.87	0.87	1.00	1.00	1.00	1.00	0.95	0.91	1.00	1.08
9	1.15	0.94	1.15	1.66	1.56	0.87	1.07	0.86	0.91	0.86	1.00	0.95	0.91	0.91	1.00
10	1.00	0.94	1.15	1.00	1.00	0.87	0.87	0.86	0.82	0.70	1.00	0.95	1.00	1.00	1.00
11	1.00	0.94	1.15	1.00	1.00	0.87	0.87	0.86	0.82	0.86	1.00	0.95	1.00	1.00	1.00

After the calculation of the mean of the 15 cost drivers, the transpose matrix of the dataset, the estimation of the covariance and the correlation set matrix, the Eigen vector is shown in Table 6.

Table 6: Eigen Vector of the Correlation Matrix

0.42	-0.14	0.02	-0.12	0.15	-0.03	-0.49	0.13	0.00	-0.34	0.41	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.71	-0.46	0.37	0.71	-0.71	-0.71	0.71
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.71	-0.46	0.37	0.71	-0.71	-0.71	0.71
0.37	0.25	0.17	0.11	0.12	-0.09	0.27	0.02	0.00	0.02	0.04	0.00	0.00	0.00	0.00
0.26	0.29	0.07	-0.17	-0.71	0.54	-0.10	-0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00
-0.11	-0.10	0.54	-0.38	0.11	0.04	0.03	-0.17	0.00	0.08	0.07	0.00	0.00	0.00	0.00
0.37	0.25	0.17	0.11	0.12	-0.09	0.27	0.02	0.00	-0.38	0.33	0.00	0.00	0.00	0.00
0.20	-0.44	-0.17	-0.23	-0.04	0.12	0.37	0.16	0.00	-0.16	-0.27	0.00	0.00	0.00	0.00
0.17	-0.38	0.15	0.25	-0.52	-0.57	-0.02	-0.39	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.09	-0.34	0.14	0.56	0.26	0.57	-0.02	-0.39	0.00	0.00	0.00	0.00	0.00	0.00	0.00
-0.11	-0.10	0.54	-0.38	0.11	0.04	0.03	-0.17	0.00	-0.08	-0.07	0.00	0.00	0.00	0.00
-0.15	-0.17	0.47	0.36	-0.22	0.05	-0.03	0.73	0.00	0.00	0.00	0.00	0.00	0.00	0.00
-0.42	0.14	-0.02	0.12	-0.15	0.03	0.49	-0.13	0.00	-0.34	0.41	0.00	0.00	0.00	0.00
-0.37	-0.25	-0.17	-0.11	-0.12	0.09	-0.27	-0.02	0.00	-0.36	0.36	0.00	0.00	0.00	0.00
0.20	-0.44	-0.17	-0.23	-0.04	0.12	0.37	0.16	0.00	0.16	0.27	0.00	0.00	0.00	0.00

The coefficients of the eigenvectors serve as the regression coefficients of the 15 principal components (PC). For example, the first PC can be expressed by:

$$PC_1 = 0.42RELY + 0DATA + 0CPLX + 0.37TIME + .. - 0.37TOOL + 0.20SCED \quad (2)$$

For dimension reduction, the eigen vector can be rearranged in their descending order to see how each variance was accounted for by the eigen vector; this and some other important information is shown in Table 7.

Table 7: Eigen value summary

Eigen Value	**Percentage Composition**	**Cumulative Percentage**
5.1849	34.57%	34.57%
3.4028	22.69%	57.26%
2.5140	16.76%	74.02%
1.1960	7.97%	81.99%
0.4576	3.05%	85.04%
0.2837	1.89%	86.93%
-0.0997	-0.67%	86.26%
0.0606	0.40%	86.66%
2.0000	13.33%	99.99%
6.04E-16	4.03e-15%	99.99%
-2.43E-16	-1.62e-15%	99.99%
-3.14E-17	-2.09e-16%	99.99%
-1.11E-17	-7.4e-17%	99.99%
3.85E-18	2.57e-17%	99.99%
-1.21E-32	-8.07e-32	99.99%
Total = 15		

The corresponding bar chart for the data in Table 7 is depicted in Figure 2.

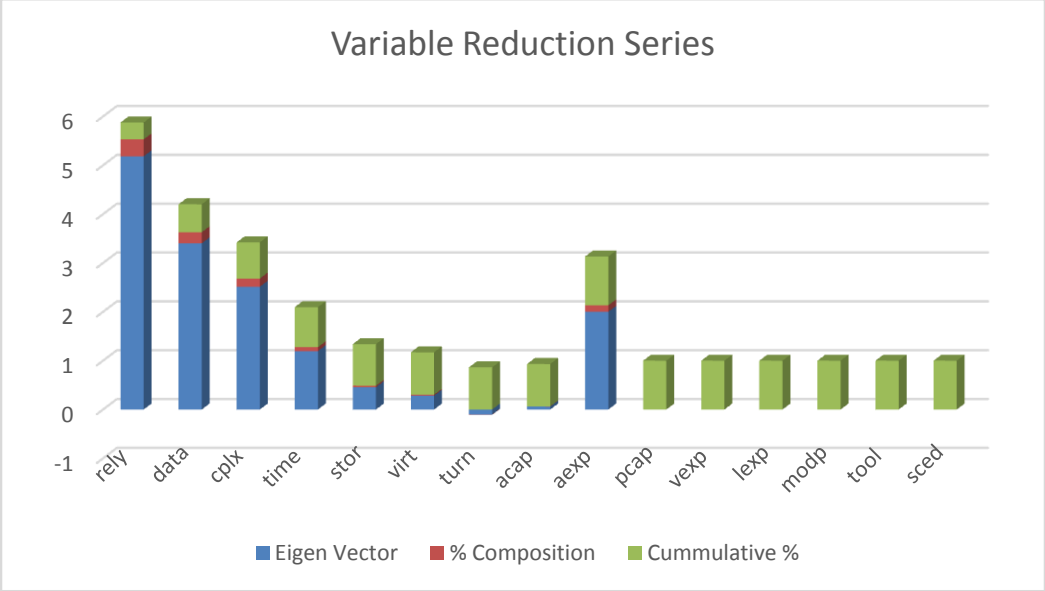

Figure 2: Variable Reduction Series from Eigen Values

From the above chart, it can seen that the principal component determinant, which is the Eigen vector favors only RELY, DATA, CPLX, TIME and AEXP based on their percentage composition. For design's sake, PCA is used to reduce 15 dimensions in the original COCOMO II file to 5 dimensions for flawless computations; however, in the implementation using MATLAB, the 23 inputs is reduced to 6.

V. ANFIS TRAINING

The model was trained with 70% of the entire dataset and tested with 30% of the dataset. Figure 3 shows the distribution of training dataset.

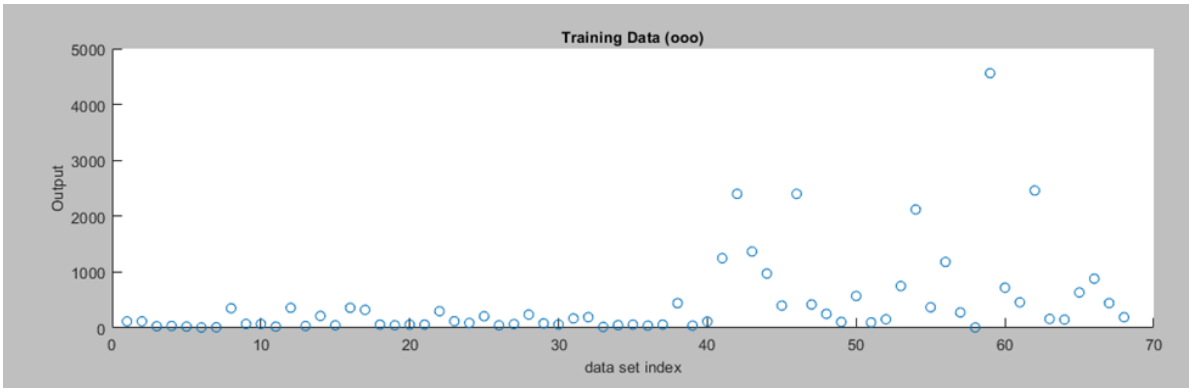

Figure 3: ANFIS training dataset

The model training was carried out on different number of epochs: 10 epoch, 20 epoch, 25 epoch, 30 epoch, 50 epoch, 70 epoch and 100 epoch. The ANFIS training result with different epoch values are presented in Figures 4 – 9. Figure 4 shows the ANFIS training error result at 10 epoch with a training error of 0.5732 and 0.0079 for Back Propagation and Hybrid algorithm respectively.

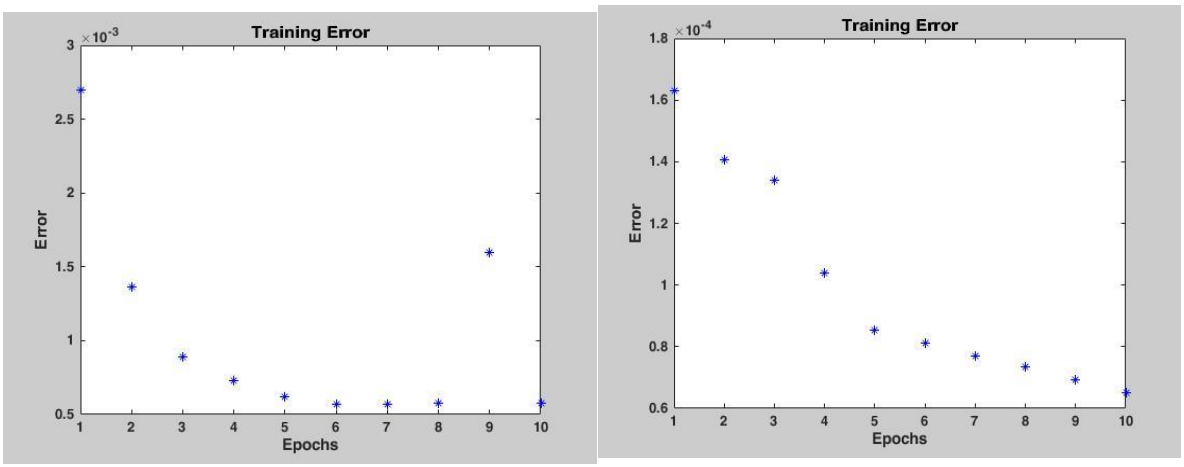

Figure 4: Training Results for Epoch 10 (Back Propagation vs Hybrid)

Figure 5 shows the ANFIS training with 20 epoch giving a training error of 0.5305 and 0.0063for Back Propagation and Hybrid algorithm respectively.

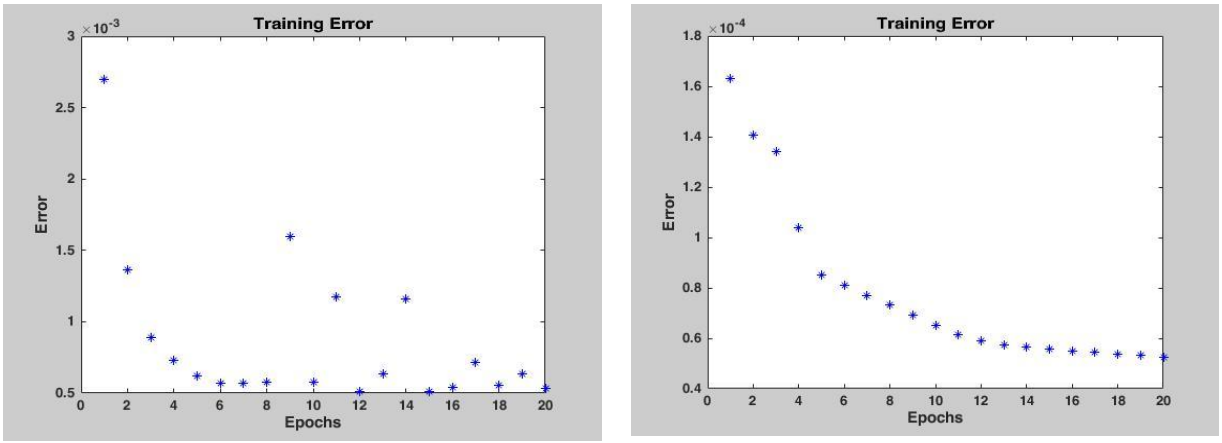

Figure 5: Training result with 20 Epochs (Back Propagation vs Hybrid)

Figure 6 is the ANFIS training result with 50 epochs giving a training error of 0.4966 and 0.0050 for Back Propagation and Hybrid algorithm respectively.

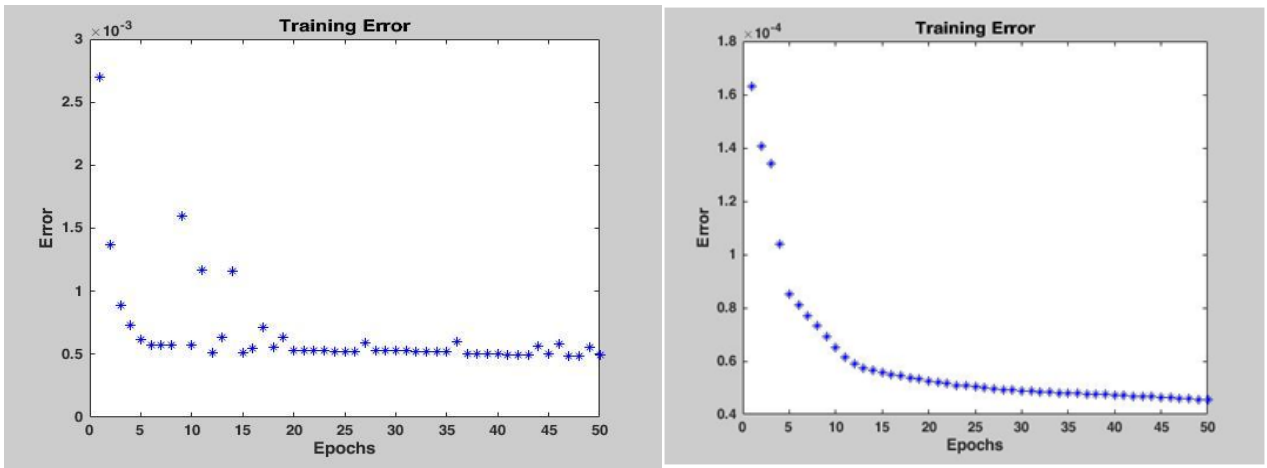

Figure 6: Training result with 50 Epoch (Back Propagation vs Hybrid)

Figure 7 presents the training results at epoch of 60 with a training error of 0.4867 and 0.0048 for Back propagation and Hybrid algorithm respectively. At Epoch 60, BP algorithm converges while the Hybrid experienced a premature convergence.

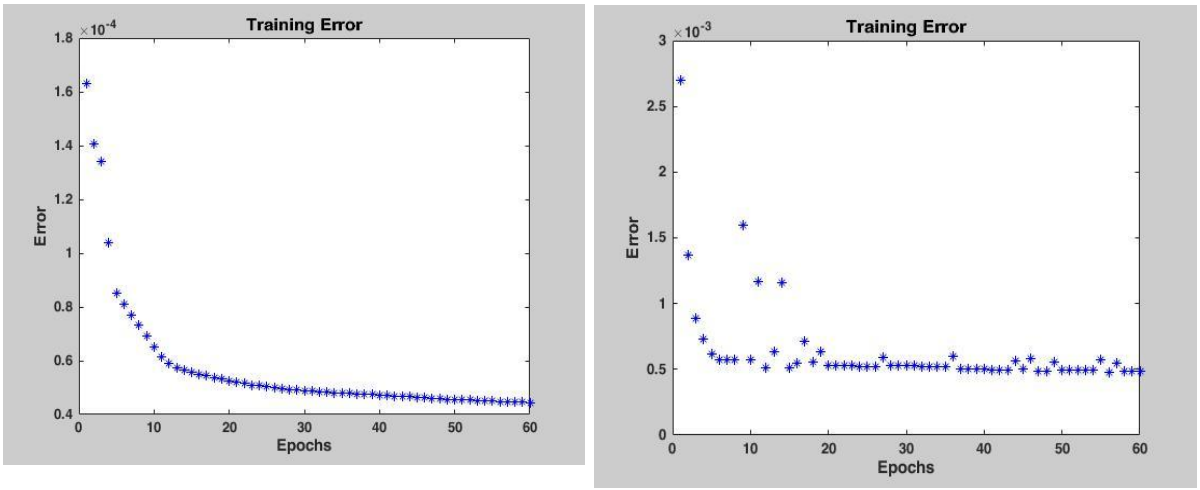

Figure 7: Training result with 60 Epoch (Back Propagation vs Hybrid)

Figure 8 presents the training results at epoch of 70 with a training error of 0.4773 and 0.0045 for Back propagation and Hybrid algorithm respectively. Figure 9 shows the training results at Epoch 100. At Epoch 70, BP algorithm had converged already with some little divergence difference of 0.0094 but the Hybrid experienced a proper convergence from Epoch 70.

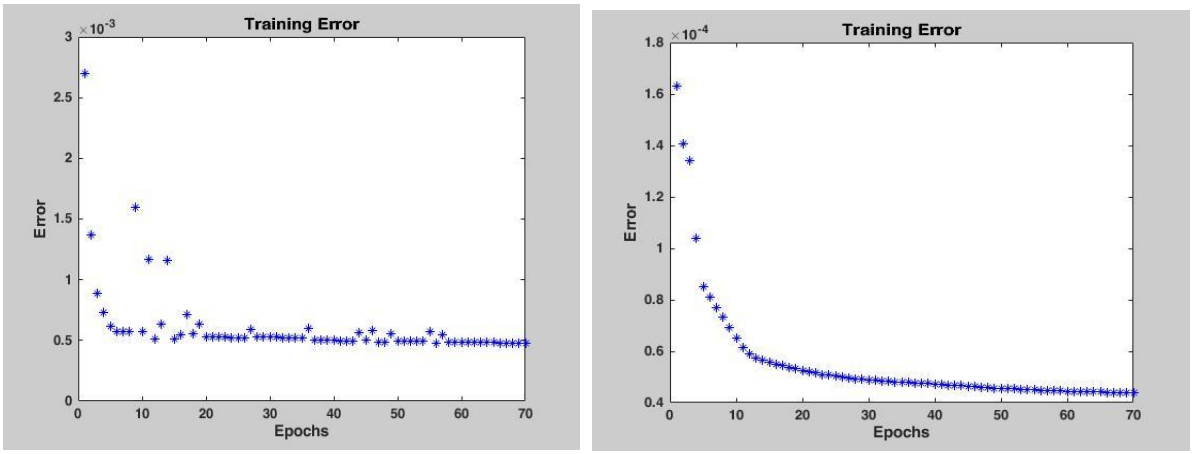

Figure 8: Training result with 70 epochs (Back Propagation vs Hybrid)

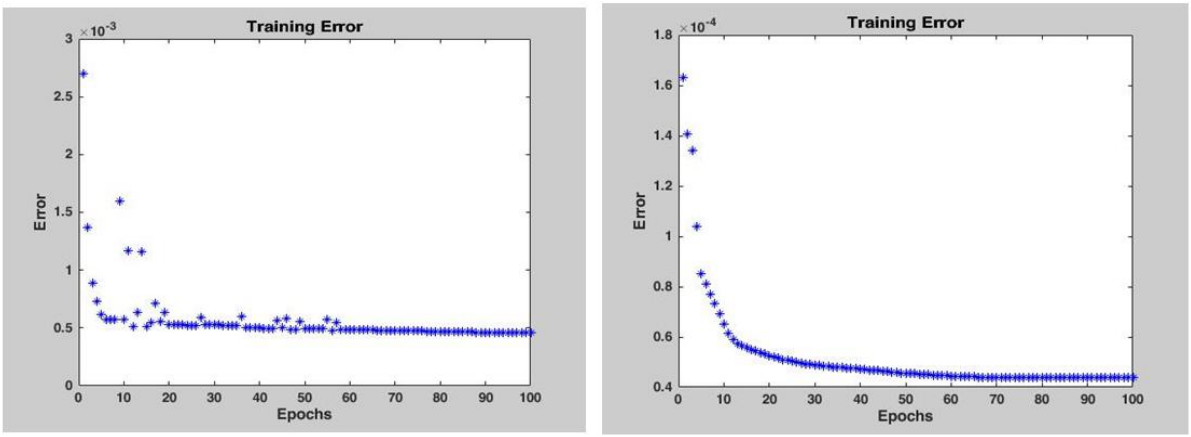

Figure 9: Training result with 100 epochs (Back Propagation (BP) vs Hybrid (HB))

The effect of epoch on the training error is presented in Table 8. This suggests also that Hybrid performs better at Epoch 70.

Table 8: ANFIS Training Errors

Metric	**Values**					
Epoch	10	20	50	60	70	100
BP Training Error	0.5732	0.5305	0.4966	0.4867	0.4773	0.4559
HB Training Error	0.0079	0.0063	0.0050	0.0048	0.0045	0.0045

The ANFIS training error plot is shown in Figure 10.

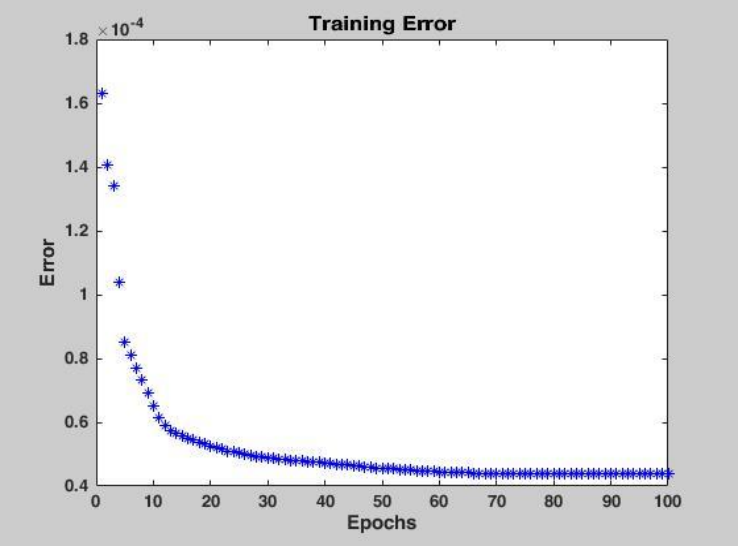

Figure 10: ANFIS Training Error Plot

In order to test the accuracy of the ANFIS model, 30% of the reduced dataset was used for model testing. The training error and validation error for each of the epoch is shown in Figure 11.

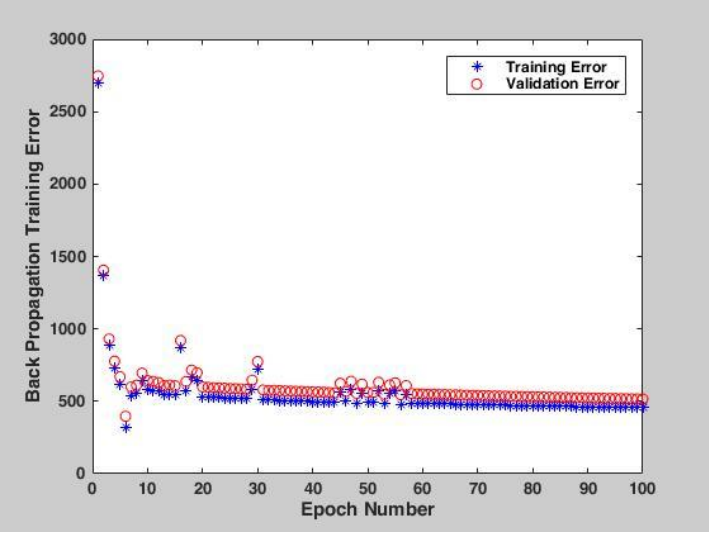

Figure 11: Training Error and Validation Error Plots

The effect of the epoch on the model testing is shown in Table 9 which presents the ANFIS testing error on the 2 different ANFIS training models trained at different epochs.

Table 9: ANFIS Testing Error

Metric	**Values**					
Epoch	10	20	50	60	70	100
Testing Error BP	0.64217	0.5991	0.56104	0.5491	0.5403	0.5160
Testing Error HB	0.5735	0.2167	0.1892	0.1889	0.1877	0.12156

The graph in Figure 12 compares the ANFIS model output and COCOMO output which is the actual effort (number of person-hours) required to develop software.

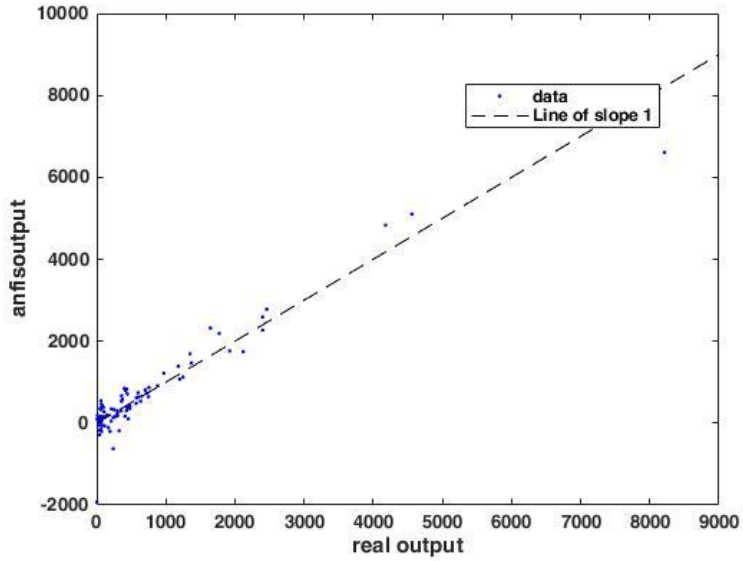

Figure 12: ANFIS Output and Real Output

Table 10 is the result of prediction comparisons between COCOMO, HB ANFIS and BP ANFIS Models. The COCOMO Model is the actual software development effort. The Rel. DIFF column is the Relative Difference between the COCOMO Effort and ANFIS effort. The result shows HB ANFIS prediction with less than 1% difference from the COCOMO effort, relative to other predictions.

Table 10: Comparison of Predictions and COCOMO Model

COCOMO Model	**HB ANFIS Model**	**Rel. DIFF**	**BP ANFIS Model**
2	2.02	0.07%	-1929.282
2	2.02	0.07%	-1929.282
2	2.02	0.07%	-1929.282
8	0.33	7.45%	99.827
8	6.02	0.67%	98.329
11	11.74	0.08%	101.436
12	132.81	705.99%	192.026
18	20.62	0.38%	104.128
24	23.91	0%	-50.582
25	39.52	7.58%	110.623
31	30.47	0.02%	107.388

36	32.73	0.29%	108.173
36	47.59	3.52%	42.990
38	69.03	21.79%	121.155
42	36.19	0.78%	-286.710
42	28.45	4.09%	-174.849
48	79.63	17.87%	357.137
48	47.25	0.01%	19.203
48	61.54	3.57%	58.849
50	42.71	1.03%	111.816
60	59.22	0.01%	118.441
60	48.31	2.15%	130.590
60	90.62	13.46%	-142.057
60	85.82	9.8%	162.348
60	66.44	0.67%	327.549
62	62.58	0.01%	541.649
70	76.50	0.59%	456.794
72	58.94	2.22%	-200.064
72	72.83	0.01%	46.129
72	19.67	29.61%	-54.742
82	81.31	0.01%	427.410
90	119.13	8.18%	160.002
97	191.20	59.26%	378.720
99	118.95	3.72%	156.979
107	88.52	2.91%	273.722
114	112.82	0.01%	-65.386
118	115.30	0.04%	146.066
118	109.08	0.59%	142.629
120	132.64	1.25%	156.110
155	168.91	1.17%	197.978
170	166.78	0.06%	-105.737
192	181.54	0.54%	-200.767
210	128.47	21.64%	47.986
215	321.99	32.69%	345.479
239	245.30	0.16%	-627.327
240	285.05	6.81%	140.486
252	247.24	0.09%	326.412
278	275.66	0.02%	167.714
300	336.55	3.73%	302.057
300	380.94	14.97%	245.872
300	316.02	0.79%	176.975
324	316.68	0.16%	-186.518
353	316.02	3.21%	302.057
360	343.26	0.72%	535.238
360	364.58	0.06%	670.311
370	359.86	0.26%	577.728

400	315.35	12.08%	847.459
409	429.74	0.95%	169.447
420	422.01	0.01%	789.459
430	345.35	11.24%	303.532
432	516.45	11.15%	830.974
444	441.56	0.01%	713.286
444	445.69	0.01%	380.968
458	436.90	0.88%	107.264
480	488.49	0.14%	414.503
480	333.92	23.37%	355.971
571	509.58	4.99%	480.314
576	360.61	33.06%	614.676
599	676.05	6.91%	745.243
600	512.74	8.47%	635.196
636	633.75	0.01%	532.145
648	645.17	0.01%	655.504
703	697.72	0.04%	807.428
720	719.95	0%	726.577
750	795.34	2.2%	644.248
756	795.25	1.69%	868.471
882	882.10	0%	908.801
973	969.17	0.01%	1216.975
1181	1178.06	0.01%	1385.985
1200	1201.80	0%	1071.178
1248	1254.66	0.03%	1125.642
1350	1350.62	0%	1689.326
1368	1372.02	0.01%	1469.094
1646	1676.70	0.5%	2320.574
1773	1741.64	0.46%	2187.276
1925	1924.47	0%	1761.842
2120	2107.53	0.07%	1744.086
2400	2400.42	0%	2591.337
2400	2399.25	0%	2270.225
2460	2461.58	0%	2777.290
4178	4179.06	0%	4834.174
4560	4558.77	0%	5099.861
8211	8210.53	0%	6608.818

VI. PERFORMANCE EVALUATION

Based on the conducted experiments, ANFIS model using Hybrid Training Model with six (6) inputs reduced by PCA gave better estimate than the Back Propagation Model with six (6) inputs reduced by PCA. The result of performance measures using MMRE and MSE for each of the training is shown in Table 11.

Table 11: Performance Evaluation Results

Measure	**BP ANFIS Model**	**HB ANFIS Model**
MMRE	388.2579	31.4829
MSE	0.000082	0.00000187

From Table 11, it is seen that Hybrid Model gives a better result than Back Propagation model.

The MRE Performance Measure of Hybrid and Back Propagation were plotted. The plot shows that the Hybrid training guaranteed convergence more than BP training. In Figure 13, the high shoots between Epoch 10 and Epoch 20, Epoch 60 and 90 were responsible for negative predictions. In some cases, the predictions appeared to be meaningful why others were not. This is where the least square method complements the back propagation algorithm to prevent large learning rate that may have occurred at the early epochs.

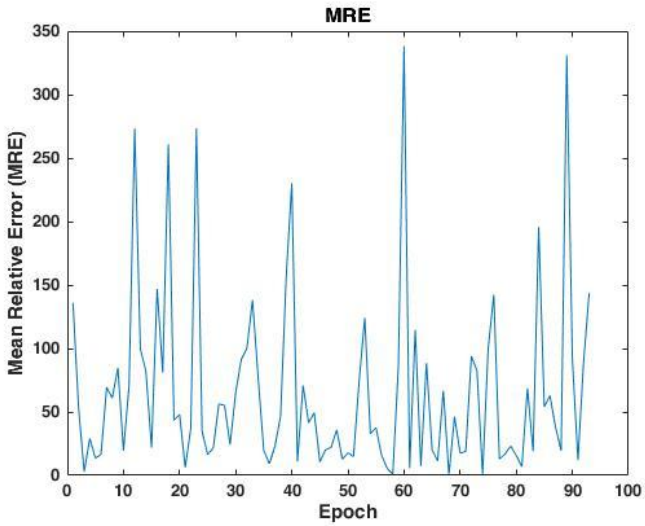

Figure 13: MRE Plot ANFIS BP

In Figure 14, the least square method complements the large learning rate that occurred between epoch 10 and 20. These learning rates, if large, affect the weights of neuron, so the least square updates parameters by minimizing the squared difference between observed data and desired data. At Epoch 20 the large learning rate was reduced by least-square hence ensuring convergence at Epoch 70.

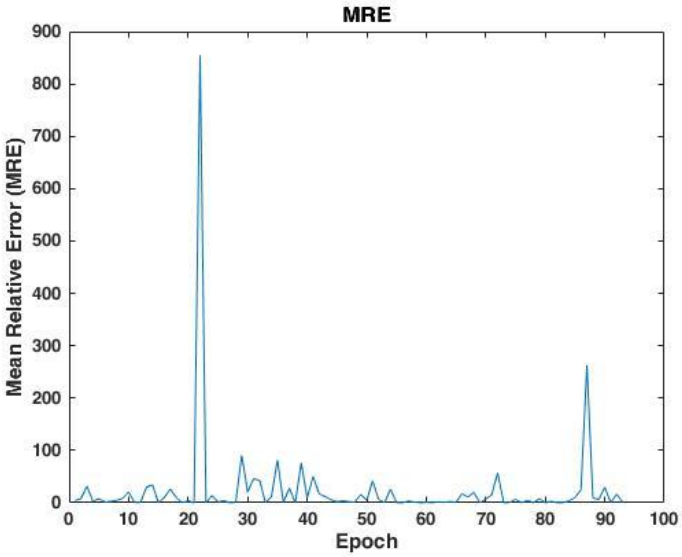

Figure 14 - MRE Plot ANFIS HB

VII. CONCLUSION

Software development effort estimation is very essential in software project management because proper effort estimation aids in the prediction of required resources, time and personnel costs. As at present, no effort estimation model has proven to be consistently successful in the prediction of software development effort, employing other approaches and/or comparing the performance of existing approaches on different situations is still open to research.

This work compares software development effort models: COCOMO, Adaptive Neuro Fuzzy Inference System (ANFIS) Back Propagation and ANFIS Hybrid. The result obtained shows that using the hybrid algorithm of ANFIS with a reduced input of 6 perform better than the COCOMO and the back propagation model while using both 23 and 6 inputs. The performance evaluation reveals that hybrid training model is more efficient and stable in terms of reduced error.

To avoid underestimation or overestimation of software development effort, which can result in catastrophic effect during software planning, many models and approaches have been proposed; with results suggesting that these models are all estimating software development effort accurately. It is quite obvious that these models may not be very successful in predicting software effort in certain situations given some other kinds of data. It is therefore pertinent to continue to compare different models using different datasets and situations all in an attempt to arrive at a generic software development effort estimation model.

For further research, the COCOMO dataset used by many authors to implement software development effort estimates possesses some level of imprecision and therefore certain factors such as productivity of team size may be introduced into the model and compared with the two ANFIS algorithms (back propagation and hybrid) to see the one which performs best.

REFERENCES

[1]. Suharjito, S., Nanda, S. and Soewito, B. (2016). Modeling Software Effort Estimation Using Hybrid PSO- ANFIS, In Proceedings of 2016 International Seminar on Intelligent Technology and Its Application, Lombok, Indonesia, 219 – 224

[2]. Chawla, R., Ahlawat, D. and Kumar, M. (2014). Software Development Effort Estimation Techniques: A Review. International Journal of Electronics Communication and Computer Engineering, 5(5), 2278 – 4209

[3]. Nassif, A. B., Azzeh, M., Idri, A. and Abran, A. (2019). Software Development Effort Estimation Using Regression Fuzzy Models. Hindawi Computational Intelligence and Neuroscience, Volume 2019, Article ID 8367214, https://doi.org/10.1155/2019/8367214

[4]. Chatzipetrou, P., Papatheocharous, E., Angelis, L., *et al.*, (2015). A multivariate statistical framework for the analysis of software effort phase distribution, *Inf. Softw. Technol.*, 2015, **59**, pp. 149–169

[5]. Shivakumar, N., Balaji, N. and Ananthakumar, K (2016): A Neuro Fuzzy Algorithm to Compute Software Effort Estimation, Global Journal of Computer Science and Technology: C - Software and Data Engineering, 16(1), 23 – 28.

[6]. Nassif, A. B., Azzeh, M., Capretz, L. F. and Ho, D. (2016). Neural Network Models for Software Development Effort Estimation: A Comparative Study. Neural Computing and Applications, 27(8), 2369-2381, doi: 10.1007/s00521-015-2127-1

[7]. Aljohani, M. and Qureshi, R. (2017). Comparative Study of Software Estimation Techniques, International Journal of Software Engineering and Applications, 8(6), 39 – 53. DOI: 10.5121/ijsea.2017.8603

[8]. Amazal, F. and Idri, A (2014). Software Development Effort Estimation using Classical and Fuzzy Analogy: A Cross-validation Comparative Study, International Journal of Computational Intelligence and Applications, 13(3), 1450013, doi: 10.1142/S1469026814500138

[9]. Eck, D., Brundick, B., Fettig, T., Dechoretz, J. and Ugljesa, J. (2009). Parametric estimating handbook, Fourth Edition, The International Society of Parametric Analysts (ISPA), Parametric estimating handbook, Fourth Edition, Vienna, VA, USA, pp 488

[10]. Lynch, J. (2009). Chaos manifesto, The Standish Group, Boston. Available Online: http://www.standishgroup.com/newsroom/chaos_2009.php.

[11]. Silhavy, R., Silhavy, P. and Prokopova, Z. (2017). Analysis and selection of a regression model for the use case points method using a stepwise approach, Journal of Systems and Software, vol. 125, pp. 1–14.

[12]. Carbonera, C. E., Farias, K. and Bischoff, V. (2020). Software development effort estimation: a systematic mapping study. Institution of Engineering and Technology Software, 14(4), 328-344

[13]. Lopez-Martin, C., Y´añez-M´arquez, C. and Gutierrez-Tornes, A. (2006). A fuzzy logic model for software development effort estimation at personal level, in Lecture Notes in Computer Science, Springer, Berlin, Germany, pp. 122–133

[14]. Jorgensen, M. (2007). Forecasting of software development work effort: Evidence on expert judgment and formal models. International Journal of Forecasting, 23(3), 449-462

[15]. Mustapha, A. (2018). Predicting Software Effort Estimation Using Machine Learning Techniques, In Proceedings of 8th International Conference on Computer Science and Information Technology, Amman, 249- 256

[16]. Marapelli, B. and Peddi, P. (2020). Effort Estimation Methods in Software Development using Machine Learning Algorithms, Parishodh Journal. Vol. IX, Issue 1, 824 – 829.

[17]. Rehmana, I., Alib, Z. and Jana, Z (2021). An Empirical Analysis on Software Development Efforts Estimation in Machine Learning Perspective, Advances in Distributed Computing and Artificial Intelligence Journal, 10(3), 227-240

[18]. Singal, P., Kumari, A. and Sharma, P. (2020). Estimation of Software Development Effort: A Differential Evolution Approach. In Proceedings of International Conference on Computational Intelligence and Data Science, Procedia Computer Science, 167, 2643 – 2652.

[19]. Loganathan, C. and Girija, K. (2014): Investigations on Hybrid Learning in ANFIS. International Journal of Engineering Research and Applications, 4(10), 31 – 37.

[20]. Srisaeng, P., Baxter, G.S., Wild, G. (2015). An adaptive neuro-fuzzy inference system for forecasting Australia's domestic low cost carrier passenger demand. Aviation, 19, 150–163.

[21]. Kamari, A., Mohammadi, A. H.., Lee, M. and Bahadori, A. (2017). Decline curve based models for predicting natural gas well performance. Petroleum, 3, 242–248
[22]. Sonmez, A. Y., Kale, S., Ozdemir, R. C. and Kadak, A. E (2018): An Adaptive Neuro-Fuzzy Inference System (ANFIS) to Predict of Cadmium (Cd) Concentrations in the Filyos River, Turkey. Turkish Journal of Fisheries and Aquatic Sciences 18:1333-1343. DOI: 10.4194/1303-2712-v18_12_01
[23]. Sahin, M. and Erol, R. (2017): A Comparative Study of Neural Networks and ANFIS for Forecasting Attendance Rate of Soccer Games. Mathematical and Computing Applications, MDPI, 22(43), DOI: 10.3390/mca22040043.
[24]. Praynlin, E. and Latha, P. (2012). Estimating Development effort on Software Projects using ANFIS. In Proceedings of International conference on Recent Trends in Computational Methods, Communication and Controls, 15 – 20
[25]. Mewada, K., Sinhal, A. and Verma, B. (2013). Adaptive Neuro-Fuzzy Inference System (ANFIS) Based Software Evaluation. International Journal of Computer Science Issues, 10(5), 244 – 250.
[26]. Sharma, S. and Vijayvargiya (2020). Enhancing Software Project Effort Estimation using Neuro-Fuzzy System. Solid State Technology, 63(6), 2986 – 2998
[27]. Prabhakar, K. and Dutta, M. (2013). Application of machine learning techniques for predicting software effort. Elixir Computer Science and Engineering, 56, 13677 – 13682
[28]. Seref, B. and Barisci, N. (2014). Software Effort Estimation Using Multilayer Perceptron and Adaptive Neuro Fuzzy Inference System. International Journal of Innovation, Management and Technology, 5(5), 374 – 377
[29]. Karimi, A. and Gandomani, T. (2021). Software development effort estimation modeling using a combination of fuzzy-neural network and differential evolution algorithm International Journal of Electrical and Computer Engineering 11(1):707-715
[30]. Le, T., Quyet, T., Nguyen, T. and Thi, M, (2021). A New Method for Enhancing Software Effort Estimation by Using ANFIS-Based Approach. A chapter in Industrial Networks and Intelligent Systems, DOI: 10.1007/978-3-030-77424-0_16