Long-term Software Fault Prediction with Robust Prediction Interval Analysis via Refined Artificial Neural Network (RANN) Approach

Momotaz Begum, *Member, IAENG,* Md. Samzid Bin Hafiz, Md. Jakirul Islam, and Mohammad Jakir Hossain

***Abstract*—Predicting software faults is one of the most challenging issues in software engineering but has not been reached yet to obtain satisfactory results. Among various methods of software fault prediction, non-homogeneous Poisson process (NHPP) based software reliability models (SRMs) and artificial neural network (ANN) are used vastly. But, the suitable model selection with parameter estimation of SRMs and appropriate architecture selection of ANN complicate the task of software fault prediction. The purpose of this paper is to predict the long-term software faults from the software fault count data using a refined artificial neural network approach (RANN). In order to pre-process software fault count data, we have used five data transformation methods to transform Poisson count data to Gaussian data. The long-term behavior of the software faults is then predicted by means of point and interval predictions. The point prediction of RANN is compared with the conventional SRMs for the case of our newly made synthetic data and eight real data in terms of predictive performance (average relative error). To ensure software reliability, we have constructed prediction intervals (*PIs*) of the predicted fault points using our proposed simulation based method (*PI_simulation*) and compared them with those of existing delta method (*PI_delta*) in terms of coverage rate and mean prediction interval width. The *PI_simulation* method covers both the real and predicted fault point within narrower interval width than the *PI_delta* method. Thus, the RANN can afford a cost effective prediction device from the viewpoint of predictability in the early phase of software testing.**

***Index Terms*—non-homogeneous Poisson process, artificial neural network, long-term prediction, software reliability, fault count data, prediction interval.**

I. Introduction

WITH the increase of the software applications in versatile fields, the mismatch between expected results and actual results, which is known as software fault, has become a major concern. Determination of software fault prediction is a crucial process of software testing [1]. The performance of a computer program without experiencing any software fault is known as software reliability. To assess the software reliability, prediction interval analysis of software faults is necessary [2], which represents the uncertainty of the prediction of software faults within a specified range. In the ref. [3], they proposed a new prediction interval method based on fuzzy numbers to improve the quality of uncertainty modelling for a higher prediction horizon. To adjust the prediction intervals for short-term load forecasting, a probabilistic method is proposed based on Association Rules and ANNs, which shows closer prediction interval maintaining accuracy [4]. Khosravi *et al.* [5] introduced the combination of prediction interval methods (Delta, Bayesian, bootstrap, and mean-variance estimation) and examined with 12 synthetic and real-world case studies to show its better quality compared to each methods.

Nowadays, the use of software reliability models (SRMs) for software fault prediction is increasing significantly [6, 7, 8]. Stochastic models like SRMs must be developed under several mathematical assumptions, and their parameter estimation is a complicated task in software testing. Another problem with SRMs is the selection of a suitable model from a huge number of SRM models, as no SRM can fit every software fault count data [9].

At present, the use of artificial neural network (ANN) with back propagation (BP) learning algorithm is very common in various classification and prediction problems [10, 11]. Karunanithi *et al.* [12] first proposed the software fault prediction using ANN. Since then, many other researchers have applied the ANN approach [13, 14]. Sitte [15] compared some ANN approaches with a set of SRMs from the viewpoint of prediction. In [16, 17, 18, 19], they utilized some evolutionary computing techniques and machine learning techniques to improve the predictive performance of the classical ANN models. Hu *et al.* [20] used the ANN approach to predict the software fault as well as to correct it.

There is no straightforward way to select a suitable architecture of neural network for each application as the number of hidden layers and neurons must be determined through a trial-and-error heuristic approach. Unlike the familiar SRMs, the software reliability can not be quantified as a probability by using the conventional ANN. Multilayer perceptron (MLP) with one-stage look-ahead prediction method was used for software release decision [21] and software fault prediction, where delta method was used for constructing the prediction intervals [2]. Recently, the same authors investigated the applicability of Box-Cox power transformation to the neuro-based long-term software fault prediction using multi-stage look-ahead prediction method, where the optimal number of hidden neurons and transformation parameter λ were determined [22].

Meanwhile, it should be noted that the conventional ANN has some limitations for the case of long-term software fault

Manuscript received February 20, 2021

Momotaz Begum is an Associate Professor in the Department of Computer Science and Engineering, Dhaka University of Engineering & Technology, Gazipur, Bangladesh (corresponding e-mail of corresponding author: drmomotaz@duet.ac.bd).

Md. Samzid Bin Hafiz is an Assistant Professor in the Department of Electrical and Electronic Engineering, Dhaka University of Engineering & Technology, Gazipur, Bangladesh (e-mail: samzid.eee@duet.ac.bd).

Md. Jakirul Islam is an Assistant Professor in the Department of Computer Science and Engineering, Dhaka University of Engineering & Technology, Gazipur, Bangladesh (e-mail: jakirduet@duet.ac.bd).

Mohammad Jakir Hossain is a Professor in the Department of Electrical and Electronic Engineering, Dhaka University of Engineering & Technology, Gazipur, Bangladesh (e-mail: jakir@duet.ac.bd).

prediction. To overcome the limitation, Begum and Dohi [23, 24] introduced a refined ANN (RANN) approach for optimal software release problem. In this paper, we have used the RANN approach for the problem of long-term software fault prediction with proposed prediction interval method. Here, we have examined the software fault count data, whereas the conventional ANNs deal with the software fault-detection time data that are not readily available for software testing. We have applied five data transformation techniques to transform the Poisson data into the Gaussian data. Then, we have made the long-term software fault prediction by using RANN with multiple inputs multiple outputs. It is worth noting that the ANN approaches in early works [12, 15, 16, 20, 17, 18, 19] have considered only one-stage look ahead prediction instead of the long-term prediction. In addition, the common SRMs also fail to consider it [25]. Through numerical experiments with eight real software fault count data, we have investigated the predictive performance of the RANN and compared it with the NHPP based SRMs. Also, we have generated Monte Carlo simulation data (*synthetic data*) and compared the performance of the transformation and non-transformation based RANNs for four datasets. Additionally, we have proposed a simulation-based method to construct the prediction interval (*PI_simulation*), which is compared with the existing delta method (*PI_delta*) [26, 27] for real datasets and *synthetic datasets*.

The rest of the paper is organised as follows: the section II describes the theoretical formulation of NHPP based SRMs, RANN, and *PI_delta* method. The proposed *PI_simulation* method along with synthetic data generation is explained in section III. In section IV, the point prediction and prediction interval analysis of RANN are explained for real and synthetic datasets. The prediction interval analysis includes the comparison between *PI_delta* and proposed *PI_simulation*.

II. Related Theory

A. Non-homogeneous Poisson process-based SRM

Suppose $Z(t)$ denotes the cumulative number of software faults detected at time t. Let (i) $Z(0) = 0$, (ii) $Z(t)$ has independent increments, (iii) $\Pr\{Z(t+q) - Z(t) \geq 2\}$=$o(q)$, (iv) $\Pr\{Z(t+q) - Z(t) = 1\}$= $\lambda(t;\boldsymbol{\theta})q$+$o(q)$, where $\lambda(t;\boldsymbol{\theta})$ is the intensity function of non-homogeneous Poisson process (NHPP), $o(q)$ is the higher term of infinitesimal time q, and $\boldsymbol{\theta}$ is the model parameter. Hence, the probability that $Z(t)$ equals y is evaluated by

$$\Pr\{Z(t) = y\} = \frac{\{\Lambda(t;\boldsymbol{\theta})\}^y}{y!} \exp\{-\Lambda(t;\boldsymbol{\theta})\}, \tag{1}$$

where

$$\Lambda(t;\boldsymbol{\theta}) = \int_0^t \lambda(y;\boldsymbol{\theta})dy \tag{2}$$

is called the mean value function. By specifying $\Lambda(t;\boldsymbol{\theta})$ or $\lambda(t;\boldsymbol{\theta})$, the identification problem of the NHPP can be diminished to a statistical problem of estimating $\boldsymbol{\theta}$. The NHPP-based SRMs are regarded as parametric SRMs due to its dependence on the model parameter $\boldsymbol{\theta}$. Table I shows NHPP-based SRMs as well as their mean value functions. Okumara and Dohi [28] developed a parameter estimation tool named SRATS. By using this tool, the model parameter $\boldsymbol{\theta}$ can be estimated.

Let t_i $(i = 1, 2, \ldots, n)$ represents the testing day, and t $(\geq t_n)$ is the observation point. The maximum likelihood method is used to evaluate the model parameter $\boldsymbol{\theta}$. Hence, the log likelihood function for the fault count data (t_i, y_i) is expressed by

$$LLF(\boldsymbol{\theta}) = \sum_{i=1}^{n}\Big((y_i - y_{i-1})\log\{\Lambda(t_i;\boldsymbol{\theta}) - \Lambda(t_{i-1};\boldsymbol{\theta})\} - \log\{(y_i - y_{i-1})!\}\Big) - \Lambda(t_n;\boldsymbol{\theta}), \tag{3}$$

where $\Lambda(0;\boldsymbol{\theta}) = 0$, $y_0 = 0$ and $t = t_n$ is used for simplicity [8]. By maximizing Eq. (3) with respect to $\hat{\boldsymbol{\theta}}$, the maximum likelihood estimate of $\hat{\boldsymbol{\theta}}$ can be attained [24, 37, 36, 38]. After evaluating the model parameter, the mean value function at an arbitrary time t_{n+l} can be represented as follows:

$$\Lambda(t_{n+l};\hat{\boldsymbol{\theta}}) = \int_0^{t_{n+l}} \lambda(y;\hat{\boldsymbol{\theta}})dy, \tag{4}$$

$$\Lambda(t_{n+l}|Z(t_n) = y_n;\hat{\boldsymbol{\theta}}) = y_n + \int_{t_n}^{t_{n+l}} \lambda(y;\hat{\boldsymbol{\theta}})dy = y_n + \Lambda(t_{n+l};\hat{\boldsymbol{\theta}}) - \Lambda(t_n;\hat{\boldsymbol{\theta}}) \tag{5}$$

where l stands for the length of prediction. In the next section, a RANN approach will be used for the long-term software fault prediction, which is regarded as a non-parametric model.

B. Refined Artificial Neural Network (RANN) Approach

ANN consists of an input layer, multiple hidden layers, and one output layer. The input layer and the output layer interact with the outside world while the hidden layers don't communicate with the outside world. Here, we have considered a multiple-inputs multiple outputs ANN with a single hidden layer as a refined ANN [23, 24].

*1) **Pre-data Processing**:* Since in the software fault prediction, the Poisson count data [7, 8] is used as integer values, the original data needs to be transformed into the Gaussian data at first [22]. In this paper, we have applied five data transformation techniques to transform the Poisson data into the Gaussian data such as *Anscombe transform* (AT1) [39], *Asymptotically unbiased Anscombe transform* (AT2) [40], *Bartlett transform* (BT) [41], *Fisz transform* (FT) [42], and *Box-Cox power transform* (BCT) [43]. The formulae of data transformation and their inverse transformation are described in Table II. Here, we have pre-processed the data $\tilde{y}_i$ by any of the five data transformation methods.

*2) **Training Phase**:* We have assumed that the uniformly distributed pseudo-random variates give nk connection weights from input to hidden layer and kl connection weights from hidden to output layer. Since the common BP algorithm does not support the training of all the weights, including $k(n + l)$ unknown patterns, it is necessary to develop a long-term point prediction method, as shown in Fig. 1. In order to get the prediction of the number of software faults for l testing days from the observation point

TABLE I: Non-homogeneous Poisson process-based SRMs.

Model (Abbr.)	Mean value function
Log extreme value minimum: (lxvmin) [29, 30]	$\Lambda(t;\boldsymbol{\theta}) = a(1-F(-\log t)), \boldsymbol{\theta} \in (a,b,c),\ F(t) = \exp(-\exp\{(-\frac{t-c}{b})\})$
Truncated extreme value minimum: (txvmin) [30]	$\Lambda(t;\boldsymbol{\theta}) = a\frac{F(0)-F(-t)}{F(0)}, \boldsymbol{\theta} \in (a,b,c),\ F(t) = \exp(-\exp\{(-\frac{t-c}{b})\})$
Log extreme value maximum: (lxvmax) [30]	$\Lambda(t;\boldsymbol{\theta}) = aF(\log t), \boldsymbol{\theta} \in (a,b,c),\ F(t) = \exp(-\exp\{(-\frac{t-c}{b})\})$
Truncated extreme value maximum: (txvmax) [30]	$\Lambda(t;\boldsymbol{\theta}) = a\frac{F(t)-F(0)}{1-F(0)}, \boldsymbol{\theta} \in (a,b,c),\ F(t) = \exp(-\exp\{(-\frac{t-c}{b})\})$
Log logistic: (llogist) [31]	$\Lambda(t;\boldsymbol{\theta}) = aF(\log t), \boldsymbol{\theta} \in (a,b,c)\ ,\ F(t) = \frac{1}{1+\exp(-\frac{t-c}{b})}$
Truncared logistic: (tlogist) [32]	$\Lambda(t;\boldsymbol{\theta}) = a\frac{F(t)-F(0)}{1-F(0)}, \boldsymbol{\theta} \in (a,b,c),\ F(t) = \frac{1}{1+\exp(-\frac{t-c}{b})}$
Log normal: (lnorm) [33, 28]	$\Lambda(t;\boldsymbol{\theta}) = aF(\log t), \boldsymbol{\theta} \in (a,b,c),\ F(t) = \frac{1}{\sqrt{2\pi}b}\int_{-\infty}^{t}\exp(-\frac{(s-c)^2}{2b^2})ds$
Truncated normal: (tnorm) [28]	$\Lambda(t;\boldsymbol{\theta}) = a\frac{F(t)-F(0)}{1-F(0)}, \boldsymbol{\theta} \in (a,b,c),\ F(t) = \frac{1}{\sqrt{2\pi}b}\int_{-\infty}^{t}\exp(-\frac{(s-c)^2}{2b^2})ds$
Pareto: (pareto) [34, 35]	$\Lambda(t;\boldsymbol{\theta}) = aF(t), \boldsymbol{\theta} \in (a,b,c),\ F(t) = 1-(\frac{c}{t+c})^b$
Gamma: (gamma) [36, 37]	$\Lambda(t;\boldsymbol{\theta}) = aF(t), \boldsymbol{\theta} \in (a,b,c),\ F(t) = \int_0^t \frac{c^b s^{b-1}\exp(-cs)}{\Gamma(b)}ds$
Exponential: (exp) [38]	$\Lambda(t;\boldsymbol{\theta}) = aF(t), \boldsymbol{\theta} \in (a,b),\ F(t) = 1-\exp(-bt)$

TABLE II: Formulae of Data transformation.

Name	Data transformation	Inverse transformation
AT1 [39]	$\tilde{y}_i = 2\sqrt{y_i+3/8}$	$\frac{\tilde{y}_{n+l}^2-3/2}{4}$
AT2 [40]	$\tilde{y}_i = 2\sqrt{y_i+1/8}$	$\frac{2\tilde{y}_{n+l}^2-1}{8}$
BT [41]	$\tilde{y}_i = 2\sqrt{y_i+1/2}$	$\frac{\tilde{y}_{n+l}^2-2}{4}$
FT [42]	$\tilde{y}_i = \sqrt{y_i+1}+\sqrt{y_i}$	$\frac{\tilde{y}_{n+l}^2+\tilde{y}_{m+l}^{-2}-2}{4}$
BCT [43]	when $\lambda=0,\quad \tilde{y}_i = \log(y_i)$	$\exp(\tilde{y}_{n+l})$
	when $\lambda\neq 0,\quad \tilde{y}_i = \frac{y_i^\lambda-1}{\lambda}$	$(\lambda\tilde{y}_{n+l}+1)^{1/\lambda}$

t_n, the prediction needs to be made at the point t_{n-l}. In that case, there are $(n-l)k+kl = nl$ estimable weights and $k(n+l)-nl$ non-estimable weights for the period $(t_{n-l}, t_n]$. In this paper, we have used $(\tilde{y}_1,\ldots,\tilde{y}_{n-l})$ and $(\tilde{y}_{n-l+1},\ldots,\tilde{y}_n)$ in the training period as the input and the teaching signals, respectively. By updating the connection weights, the squared error between teaching signals and the network outputs can be minimized. The common input bias with unit values is connected to the hidden and output neurons.

Suppose $w_{ij} \in [-1,1]$ are the connection weights from i-th ($i=0,1,\ldots,n-l$) input neuron to j-th ($j=0,1,\ldots,k$ hidden neuron, where w_{0j} and w'_{0s} denote the bias weights for j-th hidden neuron and s-th ($s=n-l+1, n-l+2,\ldots,n$) output neuron, respectively. The hidden neuron h_j is represented by:

$$h_j = \sum_{i=1}^{n-l}\tilde{y}_{ij}w_{ij} + w_{0j}. \tag{6}$$

We choose a proper value of k in the pre-experiments to evaluate the number of hidden neurons [2]. Here, we have introduced a threshold function in the RANN named sigmoid function $f(h_j) = 1/\exp(-h_j)$. Each output neuron y_s is defined by:

$$\tilde{y}_s = \sum_{j=1}^{k} f(h_j)w'_{js} + w'_{0s}. \tag{7}$$

where w'_{js} is the connection weight from j-th hidden neuron to s-th output neuron, and the final output is $f(\tilde{y}_s) = 1/\exp(-\tilde{y}_s)$. Here, the sum of square error is represented by

$$\text{Error} = \frac{\sum_{s=n-l+1}^{n}(\tilde{y}_s^o - \tilde{y}_s)^2}{(l-1)}, \tag{8}$$

where $\tilde{y}_s^o$ is teaching signal, and $\tilde{y}_s$ is point prediction for the period $(t_{n-l+1}, t_n]$.

In order to adapt the convergence speed and weights in the BP algorithm, the main tuning parameters such as the learning rate η and momentum α have been controlled and pre-experimented [22]. We have updated connection weights

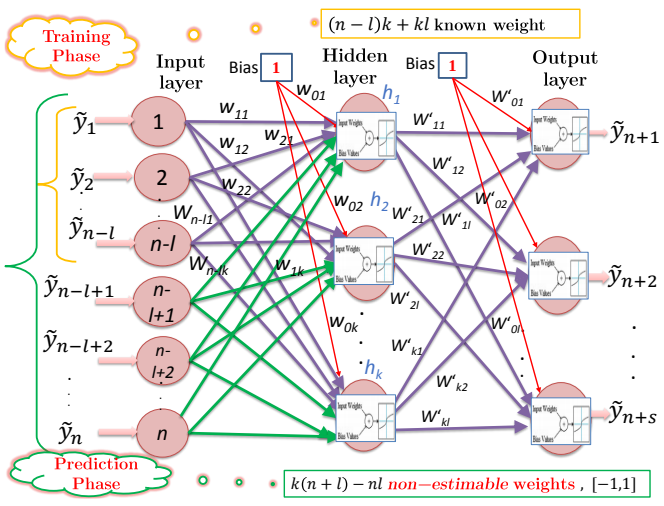


Fig. 1: Architecture of RANN.

as follows:

$$w_{ij(new)} = w_{ij} + \alpha w_{ij} + \eta \delta h_j \tilde{y}_i, \quad (9)$$

$$w'_{js(new)} = w'_{js} + \alpha w'_{js} + \eta \delta \tilde{y}_s \tilde{y}_s. \quad (10)$$

Here $\delta\tilde{y}_s$ and δh_j are the output gradient of the output layer and hidden layer, respectively, which are represented by

$$\delta \tilde{y}_s = \tilde{y}_s(1 - \tilde{y}_s)(\tilde{y}_s^o - \tilde{y}_s), \quad (11)$$

$$\delta h_j = f(h_j)(1 - f(h_j)). \quad (12)$$

The updated the bias weights for hidden and output neurons are as follows:

$$w_{0j(new)} = w_{0j} + \alpha w_{0j} + \eta \delta h_j, \quad (13)$$

$$w'_{0s(new)} = w'_{0s} + \alpha w'_{0s} + \eta \delta \tilde{y}_s. \quad (14)$$

In order to obtain the desired output, the outlined procedure is repeated.

3) ***Prediction Phase*** *:* To get the $k(n+l) - nl$ non-estimable weights, we have generated these weights by the uniform pseudo-random variates ranged in $[-1, 1]$. By using these arbitrary connection weights, the predicted point of software fault $(\tilde{y}_{n+1}, \ldots, \tilde{y}_{n+l})$ is evaluated by substituting Eqs. (6) and (7) by

$$h_{j(new)} = \sum_{i=1}^{n} \tilde{y}_{ij} w_{ij(new)} + w_{0j(new)}, \quad (15)$$

$$\tilde{y}_{n+s} = \sum_{j=1}^{k} f(h_{j(new)}) w'_{js(new)} + w'_{0s(new)}, \quad (16)$$

respectively, for $s = 1, 2, \ldots, l$. It should be noted that we have got a single output from a single sample. To get the point prediction, we have created m sets of random variates and used the arithmetic mean of the m predictions of $(\tilde{y}_{n+1}, \ldots, \tilde{y}_{n+l})$, where our preliminary experiments showed that $m = 1,000$ provides satisfactory results. Since the BP algorithm is used with combination of Monte Carlo simulation, we can call the prediction scheme as the refined artificial neural network.

C. *Prediction Interval by Delta (PI_delta) Method*

Delta method is a commonly used approach in many situations such as an approximation of the variance for an arbitrary function of random variables based on Taylor series expansions [44, 45].

Let δ_r^T be the output gradient vector for the output and hidden neurons:

$$\delta_r^T = [\delta\tilde{y}_1, \delta\tilde{y}_2, \ldots, \delta\tilde{y}_s, \delta h_1, \delta h_2, \ldots, \delta h_j] \quad (17)$$

where δh_j and $\delta\tilde{y}_s$ are defined in Eq. (11 and 12). Generally, the ANN parameters, specifically weights, are adjusted by minimizing the $Error^2$. Here Δw_r is the Jacobian matrix and defined as:

$$\Delta w_r = \begin{bmatrix} w'_{1s(new)} & w_{11(new)} & w_{21(new)} & \cdots & w_{i1(new)} \\ w'_{2s(new)} & w_{12(new)} & w_{22(new)} & \cdots & w_{i2(new)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w'_{js(new)} & w_{1j(new)} & w_{2j(new)} & \cdots & w_{ij(new)} \end{bmatrix}. \quad (18)$$

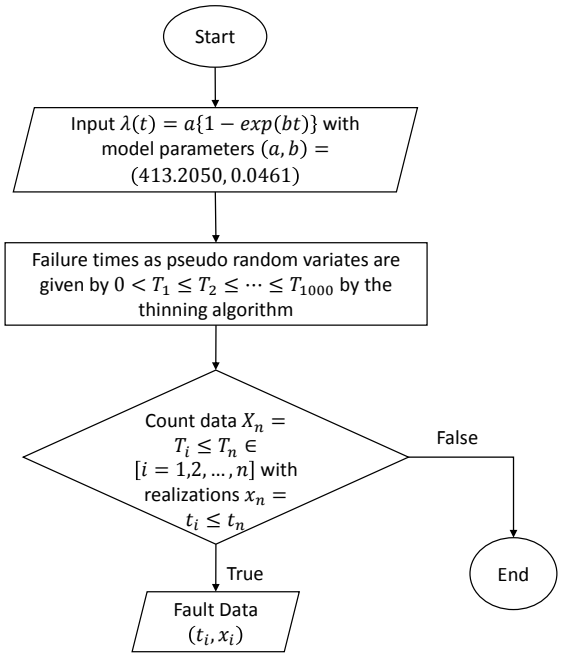


Fig. 2: Flowchart for synthetic data generation.

In the above equation, the weights $w'_{js(new)}$ and $w_{ij(new)}$ are given in Eq. (9 and 10), respectively. The prediction intervals (PIs) of the number of software faults are defined by $[PI_{low}, PI^{up}]$ in the RANN. Then, the lower limit and upper limit of PI are defined by

$$PI_{low} = \tilde{y}_{n+s} - t_{n-l}^{1-\alpha/2} \sqrt{1 + \delta_r^T (\Delta w_r^T \Delta w_r)^{-1} \delta_r}, \quad (19)$$

$$PI^{up} = \tilde{y}_{n+s} + t_{n-l}^{1-\alpha/2} \sqrt{1 + \delta_r^T (\Delta w_r^T \Delta w_r)^{-1} \delta_r}, \quad (20)$$

respectively. Here $\tilde{y}_{n+s}$ is calculated from Eq. 16, and $t_{n-l}^{1-\alpha/2}$is the $(\alpha/2)$ - quantile of the student t-distribution function with $(n-l)$ degree of freedom [44]. Here Δw_r and δ_r^T are calculated at off-line, even though they can be the potential sources of computational error for forming PIs.

III. Proposed Methods

A. *Simulation-based Prediction Interval (PI_simulation) Method*

In Section II(B), we have discussed about the RANN approach for the long-term point prediction. In Subsection II(B.2), we have estimated the weights for the period $(0, t_n]$ of the training data. To get the $k(n+l) - nl$ non-estimable weights, we need to apply the uniform pseudo-random variates ranged in $[-1, 1]$ to get the non-estimable weights. Then the predicted points of the software faults, $(\tilde{y}_{n+1}, \ldots, \tilde{y}_{n+l})$, are calculated by Eqs. (15) and (16). In order to get the PIs of the software fault count data, we have generated m sets of random variates where our preliminary experiments show that $m = 1,000$. After getting the predicted points, we have sorted these values. According to the 95% significance level, 25^{th} value acts as the lower limit, and 975^{th} value acts as the upper limit.

B. Synthetic Data Generation

The synthetic data are generated by the popular numerical computation method, named the Monte Carlo simulation. We have assumed that the software fault count process follows an exponential NHPP model, $\lambda(t) = a\{1-\exp(-bt)\}$, with model parameters $(a, b) = (413.2050, 0.0461)$ shown in Fig. 2. We have created the primary failure time data as the pseudo-random variates, which are given by $0 < T_1 \leq T_2 \leq \cdots \leq T_{1000}$ with realizations $0 < t_1 \leq t_2 \leq \cdots \leq t_{1000}$, by the *thinning algorithm* [46]. Without losing generality, we can create the count data y_i at time t_i $(i = 1, 2, \ldots, n)$. We have considered the pair (t_i, y_i) as a software fault count data of the underlying NHPP. From 1000 samples we have selected 4 types of cases: Case1 (fitted), Case2 (under fitted), Case3 (overfitted), and Case4 (s-shaped). Figure 6 shows all the datasets.

IV. Numerical illustrations

A. Real Data Analysis

*1) **Point Prediction**:* We have estimated the point prediction using our RANN described in Eq. (16). For the case of SRMs, the model parameter $\boldsymbol{\theta}$ has been estimated by using SRATS tool, and then the point prediction at any arbitrary time has been calculated by using Eq. (5). We have used eight real project data sets DS1$\sim$ DS8 [7], where the testing days and the total number of software faults are given as (62, 41, 46, 109, 111, 73, 81, 114) and (133, 266, 144, 553, 481, 367, 461, 144), respectively. In the process of getting the expected output via the BP algorithm, the calculation of the gradient descent requires high computational cost, while the convergence criteria of the minimum error and the total number of iterations are same [22]. In our experiments, the search range of λ for Box-Cox transform is $[-3, +2]$.

The prediction model's capability is estimated by the average relative error (AE),

$$AE_l = \frac{\sum_{s=1}^{l} RE_s}{l}, \tag{21}$$

where the relative error (RE_s) for the future time $t = n + s$ is defined by

$$RE_s = \left| \frac{(\tilde{y}^o_{n+s} - \tilde{y}_{n+s})}{\tilde{y}^o_{n+s}} \right|. \tag{22}$$

The prediction model with smaller AE is considered as the best one. The supplemental material [47] summarizes the analysis results for data sets DS1$\sim$ DS8 at 50% $\sim$ 90% observation points of the whole data, where the prediction lengths are l=5, 10, 15, and 20 days.

Table III shows the best model in terms of AE for dataset DS1$\sim$ DS8, where l means that prediction length, "Middle Testing" implies 50% $\sim$ 70%, and "Late Testing" represents 80% $\sim$ 90% observation points. From these results, it can be perceived that (i) our refined neural network with MIMO (FT) could predict the software faults in the early testing phase, and (ii) SRM (txvmax) provides better result than RANN in the late testing phase.

In figure 3, we depict the time-dependent behavior of RE and compare SRMs with all datasets for 20 output length. From this result, it is obvious that from the early testing to the late testing, AT2 provides less error than others. The case of non-transformation gives the worst result. Though we omit to show all the results for conciseness, it can be observed that the SRM (txvmax) leads to better results than other models.

*2) **Prediction Interval (PI) Analysis**:* Here, we consider prediction interval for software fault data by our RANN approach. To construct the prediction interval, we have used the *PI_delta* and proposed *PI_simulation* method.

There are two quality assessment criteria of PIs: the mean prediction interval width (MPIW) and PI coverage rate (PICP) [44]. The significance level is assumed as 95%. The PCIP is defined by

$$PICP = \frac{\sum_{s=1}^{l} CP_s}{l}, \quad (s = 1, 2, \ldots, l). \tag{23}$$

where

$$CP_s = \begin{cases} 1 & \tilde{y}_{n+s} \in [PI_{low}, PI^{up}] \\ 0 & \tilde{y}_{n+s} \ni [PI_{low}, PI^{up}]. \end{cases} \tag{24}$$

Here, PI_{up} and PI^{low} are the upper and lower predictive limits, respectively. The MPIW is defined by

$$MPIW = \frac{\sum_{s=1}^{l} PI^{up} - PI_{low}}{l}. \tag{25}$$

Additionally, PI-normalized averaged width (PINAW) which quantifies the PI length is given by:

$$PINAW = MPIW/l, \tag{26}$$

where l is prediction length.

Table IV shows the comparison between two prediction interval methods for DS1$\sim$DS8 where l = 20. In the table, *PI_delta* and *PI_simulation* mean the prediction intervals by delta and proposed simulation methods, respectively. From these results, it is observed that the delta method provides wider PIs than the simulation-based method although its coverage probability is over 95% significance level. The RANN with BCT and non-transformation (NT) do not cover the coverage probability for all cases.

Figure 4$\sim$5 shows the two-sided 95% prediction intervals of software faults in case of l = 20 via two prediction interval methods. The length of each rectangle in the Box plot indicates the two-sided 50% prediction intervals with DS1. Firstly, we observe that the PIs of the delta method are wider than those of the simulation-based method as the delta method is prone to much more uncertainty. The non-transformation RANN does not cover the point prediction and the real value because of its tighter prediction regions with low coverage probability. Secondly, it is seen that the RANN with AT1, AT2, BT, FT, and BCT includes the point prediction and the real value within the coverage range.

B. Synthetic Data Analysis

*1) **Point Prediction**:* For synthetic data Case1$\sim$Case4, we observe 50% past data as our input to the ANN for the next 15 days. At the observation point t_n, we have utilized the $(n-l)$ software fault count data for training the RANN. After pre-processing the synthetic datasets, we have made the long-term point prediction by five data transformation methods

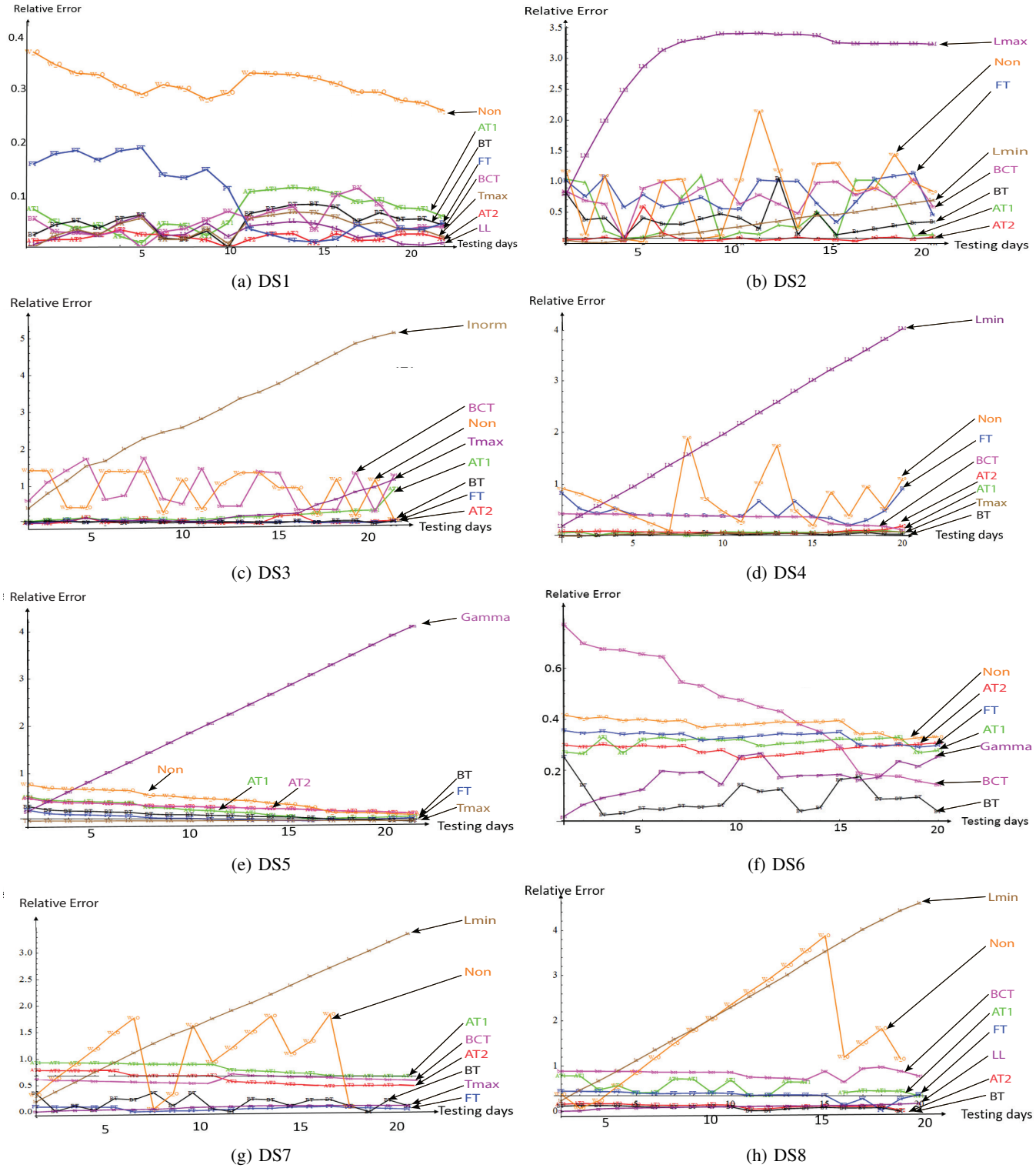


Fig. 3: Inter-temporal behavior of relative error in real data analysis.

using Eq. (16) for the next 15 days. The point prediction has been justified by the average error (AE) in Eq. (17).

Table V outlines the results of AE for Case1∼Case4 at the 50% observation points for the prediction length $l = 15$ days, where "Best λ" stands for the optimal transformation parameter, and the bold number denotes the best prediction model for the case of minimum AE. Here, the number of hidden neurons changes from $k = 10 \sim 50$. In our RANN, we have compared the non-transformed method with the five data transform methods. We observe that in almost all cases our approach gives smaller AEs than the non-transformation method. For Case1∼Case4, the RANN with FT offers less error than the non-transformation method. However, for the synthetic datasets, the data transformation with RANN method can predict accurately for long-term prediction.

In Figure 7, we delineate the time-dependent behavior of RE and compare five data transformation methods (including non-transformation method) for all cases. From this result, it is obvious that FT delivers fewer error. The common

TABLE III: Best prediction models for varying prediction point and prediction length

(a) DS1.

l	Middle Phase			Late Phase	
	50%	60%	70%	80%	90%
5	MIMO (BCT)	**MIMO (FT)**	MIMO (BCT)	SRM (txvmin)	**SRM (txv-max)**
10	MIMO (AT2)	SRM (tlo-gist)	MIMO (AT2)	SRM (lxv-max)	-
15	MIMO (AT2)	MIMO (AT2)	SRM (txv-max)	-	-
20	MIMO (BCT)	MIMO (AT2)	-	-	-

(b) DS2.

l	Middle Phase			Late Phase	
	50%	60%	70%	80%	90%
5	**MIMO (FT)**	**MIMO (FT)**	SRM (lxv-max)	SRM (lxvmin)	**SRM (txv-max)**
10	MIMO (AT2)	MIMO (AT2)	SRM (lxv-max)	-	-
15	**MIMO (FT)**	MIMO (BCT)	-	-	-
20	MIMO (AT2)	-	-	-	-

(c) DS3.

l	Middle Phase			Late Phase	
	50%	60%	70%	80%	90%
5	MIMO (AT2)	MIMO (BT)	MIMO (BT)	MIMO (FT)	**SRM (txv-max)**
10	MIMO (BT)	SRM (txv-max)	SRM (txv-max)	-	-
15	MIMO (AT1)	SRM (txv-max)	-	-	-
20	**MIMO (FT)**	-	-	-	-

(d) DS4.

l	Middle Phase			Late Phase	
	50%	60%	70%	80%	90%
5	**MIMO (FT)**	SRM (txv-max)	SRM (txv-max)	SRM (lxv-max)	SRM (lxv-max)
10	MIMO (BCT)	MIMO (BT)	SRM (lxv-max)	MIMO (AT2)	-
15	SRM (txv-max)	MIMO (AT1)	SRM (txv-max)	-	-
20	MIMO (BT)	MIMO (AT1)	-	-	-

(e) DS5.

l	Middle Phase			Late Phase	
	50%	60%	70%	80%	90%
5	SRM (gamma)	SRM (txv-max)	SRM (txv-max)	SRM (lxv-max)	SRM (lxv-max)
10	**MIMO (FT)**	SRM (lxv-max)	SRM (txv-max)	SRM (lxv-max)	**SRM (txv-max)**
15	MIMO (BT)	SRM (txv-max)	SRM (txv-max)	**SRM (txv-max)**	-
20	MIMO (BT)	SRM (txv-max)	MIMO (BT)	-	-

(f) DS6.

l	Middle Phase			Late Phase	
	50%	60%	70%	80%	90%
5	MIMO (BT)	**MIMO (FT)**	SRM (txv-max)	MIMO (FT)	SRM (lxv-max)
10	**MIMO (FT)**	MIMO (BT)	MIMO (FT)	**SRM (txv-max)**	-
15	MIMO (BT)	MIMO (AT2)	MIMO (FT)	**SRM (txv-max)**	-
20	MIMO (AT1)	MIMO (BT)	MIMO (AT1)	-	-

(g) DS7.

l	Middle Phase			Late Phase	
	50%	60%	70%	80%	90%
5	**MIMO (FT)**	SRM (txv-max)	SRM (txv-max)	MIMO (BT)	**SRM (txv-max)**
10	MIMO (BT)	MIMO (AT1)	**MIMO (FT)**	MIMO (FT)	-
15	MIMO (BT)	SRM (txv-max)	MIMO (AT1)	**SRM (txv-max)**	-
20	MIMO (BT)	**MIMO (FT)**	MIMO (BT)	-	-

(h) DS8.

l	Middle Phase			Late Phase	
	50%	60%	70%	80%	90%
5	**MIMO (FT)**	SRM (lxv-max)	SRM (lxv-max)	MIMO (BT)	MIMO (BT)
10	MIMO (BCT)	MIMO (BT)	**MIMO (FT)**	SRM (lxv-max)	SRM (exp)
15	**MIMO (FT)**	MIMO (AT2)	MIMO (BT)	**SRM (txv-max)**	-
20	MIMO (AT2)	MIMO (BT)	MIMO (AT2)	MIMO (BT)	-

non-transformation method gives the worst result in the full testing phase in comparison to others. On the other hand, the most standard BT, AT2, and BCT provide worse results and show the increasing trend.

*2) **Prediction Interval Analysis**:* We have constructed PIs for synthetic data (Case1∼Case4) by using two methods: proposed *PI_simulation* and *PI_delta* method. Here, we have used the point prediction of four simulation data sets from the previous subsection and set the significance level at 95%. We have calculated the prediction interval at 50% observation point with l=15 for all data sets. Table VI shows the result of predictive measures for all cases. First, we have found that the measures based on *PI_delta* method are wider than those of the proposed *PI_simulation* method. In addition, the transformation in RANN gives the wider PIs and the higher coverage rate over 95% in almost all cases. On the other hand, the non-transformation approach provides the narrow range and does not include the coverage probability. The observations from Table VI are illustrated in figures 8∼9, which show the two-sided 95% prediction intervals of software fault data in Case1 with the delta and simulation-based methods. In Case1, RANNs with AT1, FT, and BT include the real value and the point prediction in the Box plot. But, AT2, BCT, and non-transformation do not include those in their intervals.

V. Conclusions

RANN approach is used for long-term software fault prediction based on software fault count data. The prediction length could be given arbitrarily based on the system testing length. As far as we know, this paper is the first to construct and analyze the prediction intervals of the long-term software faults using the RANN approach. Here, the RANN has been compared with existing SRMs to show its effectiveness in predicting software faults in the early testing phase. we have also constructed the prediction intervals of the cumulative number of software faults by the *PI_delta* and proposed *PI_simulation* method in numerical experiments with eight real datasets and Synthetic datasets (four datasets). We find that *PI_simulation* encompasses both the real and predicted fault point within narrower interval width in comparison to existing *PI_delta* method. In future, these experimental results will be validated through the software metrics (e.g., McCabe, Halstead, OO Metrics) with different prediction interval methods.

TABLE IV: Assessment of prediction interval for DS1 $\sim$ DS8, where DT = Data transformation and NT=Non-transformation.

DS1 (60% observation point)						
DT	*PI_delta*			*PI_Simulation*		
	PICP	MPIW	PINAW	PICP	MPIW	PINAW
AT1	0.9589	2174.98	108.75	0.9785	2152.74	107.63
AT2	0.9525	2112.45	105.62	0.9512	1618.45	100.66
FT	0.9458	1049.78	52.48	0.9569	2013.25	98.23
BT	0.9545	2098.25	109.91	0.9584	1964.78	49.92
BCT	0.9558	1075.25	53.75	**0.9348**	998.48	80.92
NT	**0.9289**	1017.12	49.85	**0.9308**	1025.12	51.96
DS2 (50% observation point)						
AT1	0.9585	2945.85	147.29	0.9683	2143.57	107.19
AT2	0.9472	2515.56	125.78	0.9545	1945.41	97.27
FT	0.958	3258.86	162.94	0.954	2145.78	107.29
BT	0.9654	3982.41	199.12	0.9635	3120.12	156.01
BCT	0.9523	2134.57	106.73	0.9571	2051.23	102.56
NT	**0.9445**	1325.85	66.29	**0.9347**	1643.27	82.16
DS3 (50% observation point)						
AT1	0.9558	5015.47	250.77	0.9595	4019.92	200.99
AT2	0.9549	4045.89	200.64	0.9681	3647.72	182.38
FT	0.9581	4012.12	200.61	0.9533	3782.25	189.11
BT	0.9625	4317.82	215.89	0.9543	3758.87	187.94
BCT	**0.9455**	2915.85	145.79	0.9565	3247.29	162.36
NT	**0.9248**	2654.35	132.72	**0.9351**	2471.19	123.56
DS4 (60% observation point)						
AT1	0.9532	5263.56	263.19	0.9683	6458.51	322.93
AT2	0.9582	5796.59	289.83	0.9523	5894.73	294.74
FT	0.9685	6125.49	306.27	0.9519	4785.84	239.29
BT	0.9651	5872.2	293.61	0.9453	4506.89	225.34
BCT	**0.9451**	4969.58	248.48	**0.9351**	4891.47	244.57
NT	**0.9263**	2920.23	146.01	**0.9289**	1279.91	63.99

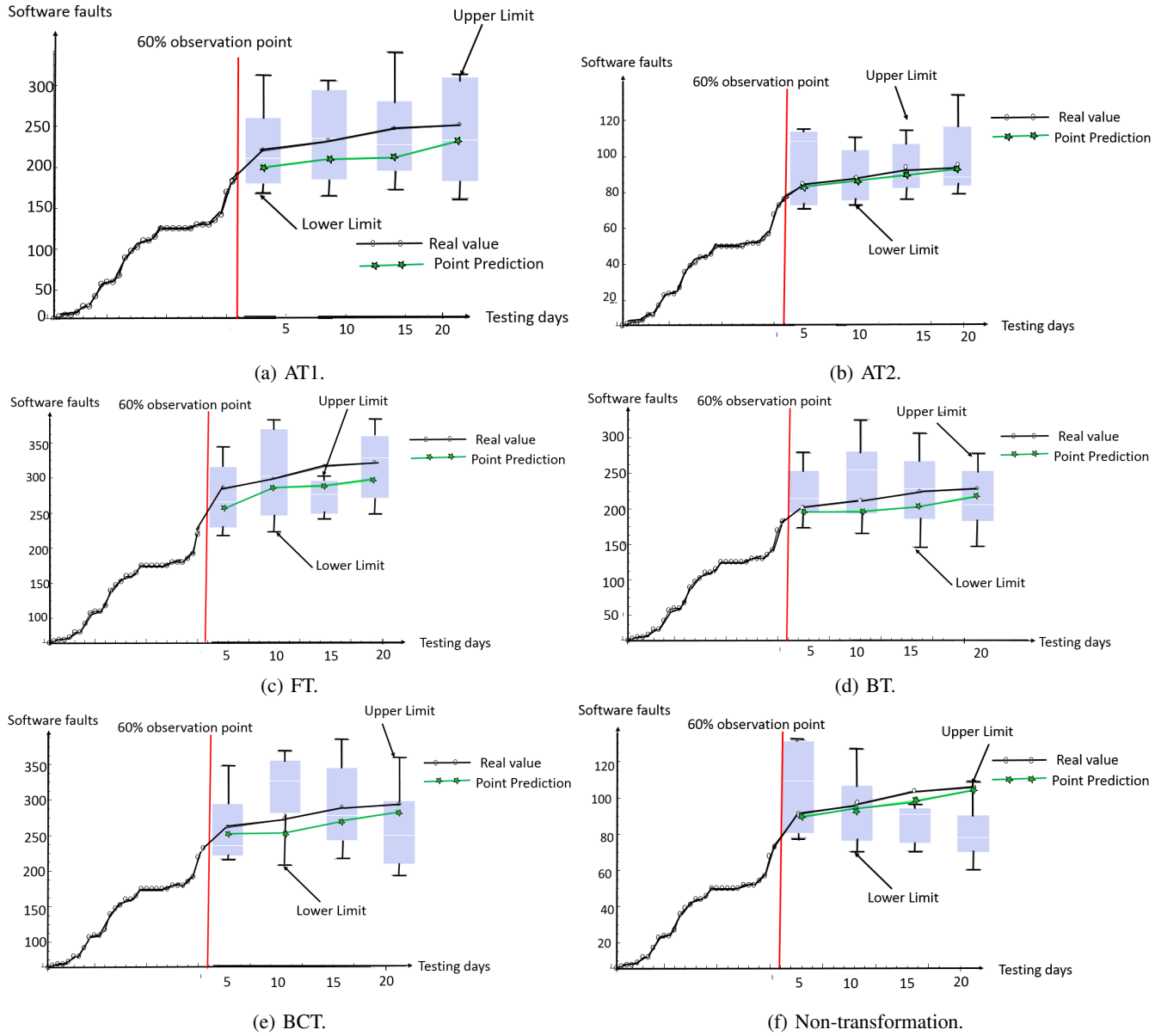
DS5 (60% observation point)						
DT	*PI_delta*			*PI_Simulation*		
	PICP	MPIW	PINAW	PICP	MPIW	PINAW
AT1	0.9545	6589.91	299.54	0.9735	6989.87	349.49
AT2	0.9659	7156.59	357.83	0.9686	6789.46	339.47
FT	0.9586	7258.51	362.92	0.9541	6243.31	312.16
BT	0.9647	7564.43	378.22	0.9547	5678.71	283.93
BCT	**0.9453**	5489.94	274.5	**0.9458**	5472.82	273.64
NT	**0.9382**	4234.37	211.72	**0.9453**	4989.59	249.48
DS6 (60% observation point)						
AT1	0.9582	4712.24	235.61	0.9535	4576.67	228.83
AT2	0.9512	4523.33	226.17	0.9579	4971.18	248.56
FT	0.9612	4871.17	243.56	0.9589	4283.23	214.16
BT	0.951	4279.29	213.96	0.9519	3986.58	199.33
BCT	0.9543	4585.83	229.29	0.9534	4575.51	228.77
NT	**0.9245**	2987.49	149.37	**0.9452**	3698.45	184.92
DS7 (60% observation point)						
AT1	0.9586	5698.45	284.92	0.9615	6125.89	306.29
AT2	0.9512	4953.23	247.66	0.9536	4989.91	249.46
FT	0.9586	5941.49	297.07	0.9585	6124.47	306.22
BT	0.9596	5210.31	260.52	0.9589	5426.63	271.33
BCT	0.9556	4653.28	232.66	**0.9495**	4523.21	226.16
NT	**0.9289**	3961.45	198.07	**0.9453**	3212.24	160.61
DS8 (60% observation point)						
AT1	0.9541	745.41	37.27	0.9555	643.23	32.16
AT2	0.9652	659.86	32.99	0.9656	712.2	35.61
FT	0.9553	598.89	29.94	0.9562	562.12	28.11
BT	0.9659	963.12	48.16	0.9585	496.56	24.83
BCT	0.9532	489.87	24.49	0.9594	589.89	29.49
NT	**0.9251**	233.53	11.68	**0.9343**	375.86	18.79

TABLE V: Comparison of AEs for Case1 $\sim$ Case4 ($l = 15$) at 50% observation point. Here, NT stands for Non-transformation.

Case1						
k	AT1	AT2	FT	BT	BCT(Best λ)	NT
10	1.0563	1.0952	0.8952	0.7514	1.3698(1.1)	2.4589
20	0.8531	1.1361	0.0756	1.369	0.9458(1.0)	1.9412
30	0.1362	0.2492	0.0911	0.3448	0.2771(0.7)	0.5062
40	0.3589	0.8781	**0.0396**	0.6391	1.069(0.3)	1.3497
50	0.1015	0.2639	0.0963	0.3940	0.6852(0.1)	0.7423
Case2						
10	2.5896	3.4712	2.4820	1.6352	3.6094(1.1)	4.8956
20	2.1356	2.9841	**0.0641**	2.4810	1.0489(0.8)	3.6523
30	3.0564	1.8971	0.6749	0.9423	1.6357(1.4)	1.4758
40	0.9852	0.0918	0.0686	1.2301	0.6987(-0.3)	0.3497
50	0.0779	1.2891	0.9852	0.3289	0.4024(1.3)	0.5045
Case3						
10	2.1324	0.0654	0.6894	2.4120	2.3145(-1.9)	4.2536
20	1.4578	1.6932	0.0335	1.9482	0.2837(2.0)	2.5896
30	0.0229	1.3472	1.2035	**0.0151**	1.9823(0.0)	0.3811
40	0.6417	0.9856	1.4712	0.7519	1.8742(1.2)	1.9786
50	0.2471	0.8945	0.3691	0.2981	0.8974(1.0)	1.8421
Case4						
10	3.6451	1.6523	0.6417	0.8741	2.8963(1.3)	3.6012
20	1.6481	1.9631	1.2013	1.2475	1.9876 (-1.1)	3.1987
30	1.9685	0.3781	1.2489	0.3419	1.0947(0.9)	2.8475
40	0.2896	1.0143	1.0321	0.6389	1.6981(-0.8)	1.2589
50	1.0241	0.2781	**0.2389**	0.3141	0.8949(-0.3)	1.3698

TABLE VI: Results of predictive measures for Case1 $\sim$ Case4 at 50% observation point, where DT = Data transformation and NT=Non-transformation.

Case1						
DT	Delta			Simulation		
	PICP	MPIW	PINAW	PICP	MPIW	PINAW
AT1	0.9632	157.95	10.51	0.9536	170.71	11.38
AT2	0.9482	229.02	15.27	0.9621	137.13	9.14
FT	0.9549	107.91	7.19	0.9429	106.47	7.09
BT	0.9521	113.71	7.59	0.9587	52.08	3.47
BCT	0.9479	183.75	12.25	0.9425	96.96	6.46
NT	**0.9189**	46.30	3.08	**0.9349**	43.13	2.88
Case2						
AT1	0.9592	197.62	13.17	0.9693	192.51	12.83
AT2	0.9653	196.38	13.09	0.9589	167.48	11.16
FT	0.9617	141.03	9.40	0.9623	124.48	8.29
BT	0.9551	157.57	10.51	0.9593	125.90	8.39
BCT	0.9546	177.88	11.85	0.9459	167.62	11.17
NT	**0.9263**	111.95	7.46	**0.9129**	135.25	9.017
Case3						
AT1	0.9512	142.94	9.53	0.9521	129.42	8.63
AT2	0.9641	198.22	13.21	0.9523	151.75	10.12
FT	0.9581	204.45	13.63	0.9557	188.67	12.58
BT	0.9547	75.34	5.02	0.9562	101.44	6.76
BCT	0.9596	143.83	9.58	0.9459	76.87	5.12
NT	**0.9154**	51.62	3.44	0.9591	92.24	6.15
Case4						
AT1	0.9589	125.07	8.34	0.9683	115.78	7.72
AT2	0.9492	156.76	10.45	0.9614	121.47	8.09
FT	0.9859	193.94	12.93	0.9519	158.89	10.59
BT	0.9459	179.42	11.96	0.9598	129.89	8.65
BCT	0.9587	157.64	10.51	0.9241	111.47	7.43
NT	**0.9459**	125.45	8.36	**0.9137**	89.78	5.99



(a) AT1.

(b) AT2.

(c) FT.

(d) BT.

(e) BCT.

(f) Non-transformation.

Fig. 4: prediction interval by delta method with DS1 ($l = 20$).

REFERENCES

[1] L. Lun, X. Chi, and H. Xu, "Testing approach of component interaction for software architecture." *IAENG International Journal of Computer Science*, vol. 45, no. 2, pp. 353–363, 2018.

[2] M. Begum and T. Dohi, "Prediction interval of cumulative number of software faults using multilayer perceptron," in *Applied Computing & Information Technology*. Springer, 2016, pp. 43–58.

[3] L. G. Marín, N. Cruz, D. Sáez, M. Sumner, and A. Núñez, "Prediction interval methodology based on fuzzy numbers and its extension to fuzzy systems and neural networks," *Expert Systems with Applications*, vol. 119, pp. 128–141, 2019.

[4] M. A. Zuniga-Garcia, G. Santamaría-Bonfil, G. Arroyo-Figueroa, and R. Batres, "Prediction interval adjustment for load-forecasting using machine learning," *Applied Sciences*, vol. 9, no. 24, p. 5269, 2019.

[5] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "Comprehensive review of neural network-based prediction intervals and new advances," *IEEE Transactions on neural networks*, vol. 22, no. 9, pp. 1341–1356, 2011.

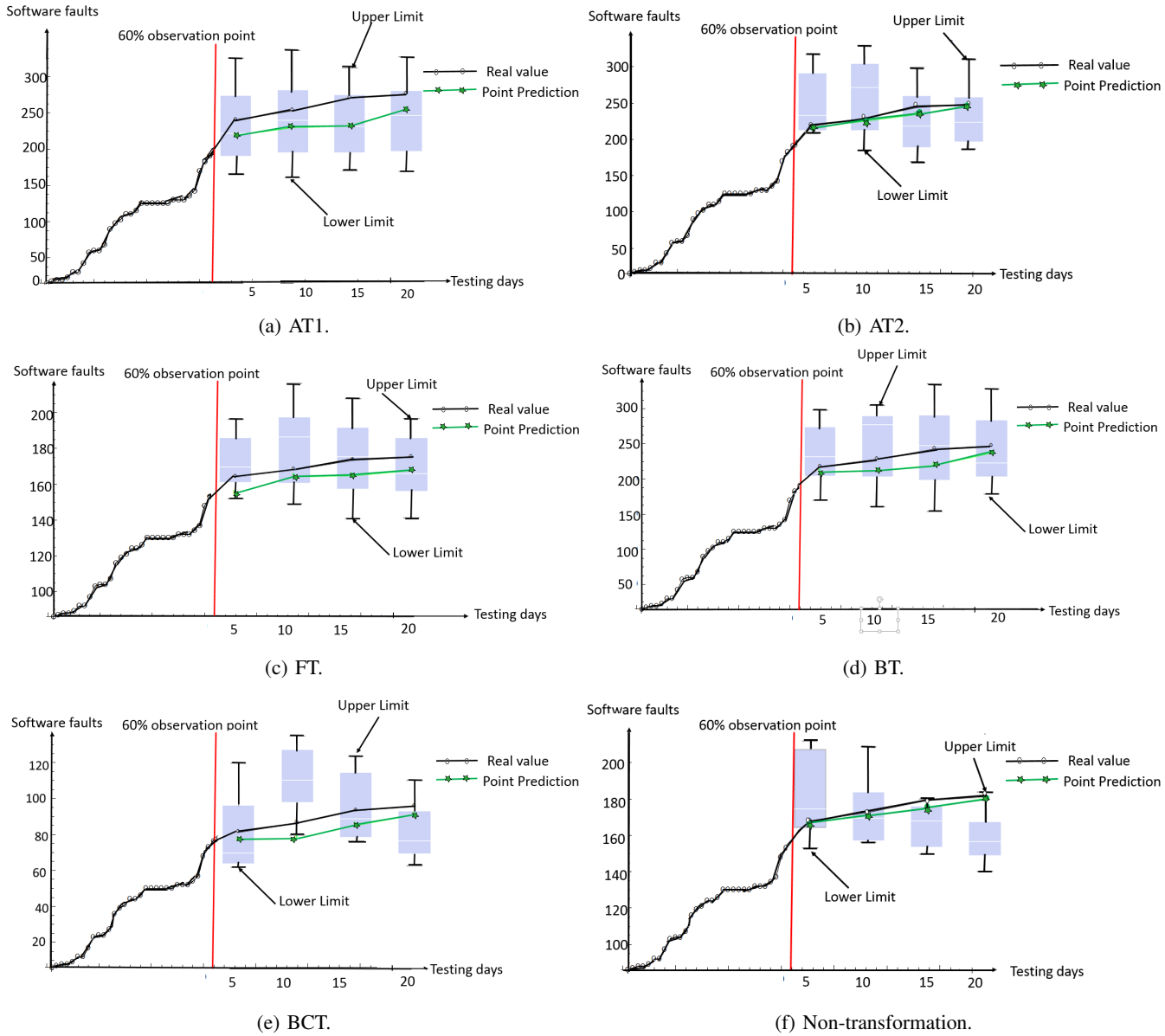
[6] K.-Y. Cai, *Software defect and operational profile modeling*. Springer Science & Business Media, 2012, vol. 4.

[7] M. R. Lyu *et al.*, *Handbook of software reliability engineering*. IEEE computer society press CA, 1996, vol. 222.

[8] J. D. Musa, A. Iannino, and K. Okumoto, "Software reliability: Measurement, prediction, application. 1987," *McGrawHill, New York*, 1987.

[9] K. Sharma, R. Garg, C. Nagpal, and R. Garg, "Selection of optimal software reliability growth models using a distance based approach," *IEEE Transactions on Reliability*, vol. 59, no. 2, pp. 266–276, 2010.

[10] Y. Wang, D. Niu, and L. Ji, "Short-term power load forecasting based on ivl-bp neural network technology," *Systems Engineering Procedia*, vol. 4, pp. 168–174,



(a) AT1.

(b) AT2.

(c) FT.

(d) BT.

(e) BCT.

(f) Non-transformation.

Fig. 5: prediction interval by simulation-based method with DS1 ($l = 20$).

2012.

[11] S. Santosa, R. Pramunendar, D. Prabowo, and Y. P. Santosa, "Wood types classification using back-propagation neural network based on genetic algorithm with gray level co-occurrence matrix for features extraction." *IAENG International Journal of Computer Science*, vol. 46, no. 2, pp. 149–155, 2019.

[12] N. Karunanithi, Y. K. Malaiya, and L. D. Whitley, "Prediction of software reliability using neural networks," in *ISSRE*, 1991, pp. 124–130.

[13] T. Wu, Y. Dong, Z. Dong, A. Singa, X. Chen, and Y. Zhang, "Testing artificial intelligence system towards safety and robustness: State of the art." *IAENG International Journal of Computer Science*, vol. 47, no. 3, pp. 449–462, 2020.

[14] W. Wongsinlatam and S. Buchitchon, "Criminal cases forecasting model using a new intelligent hybrid artificial neural network with cuckoo search algorithm." *IAENG International Journal of Computer Science*, vol. 47, no. 3, pp. 481–490, 2020.

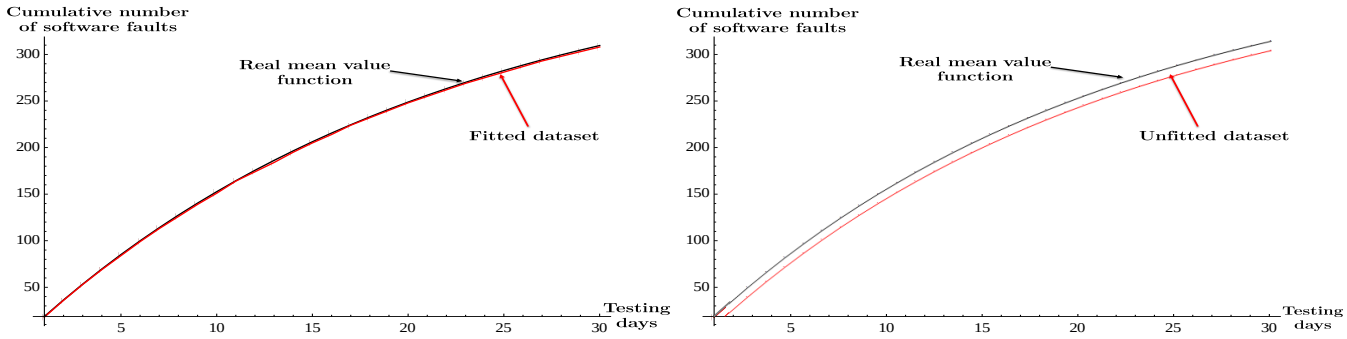
[15] R. Sitte, "Comparison of software-reliability-growth predictions: neural networks vs parametric-recalibration," *IEEE transactions on Reliability*, vol. 48, no. 3, pp. 285–291, 1999.

[16] L. Tian and A. Noore, "Evolutionary neural network modeling for software cumulative failure time prediction," *Reliability Engineering & system safety*, vol. 87, no. 1, pp. 45–51, 2005.

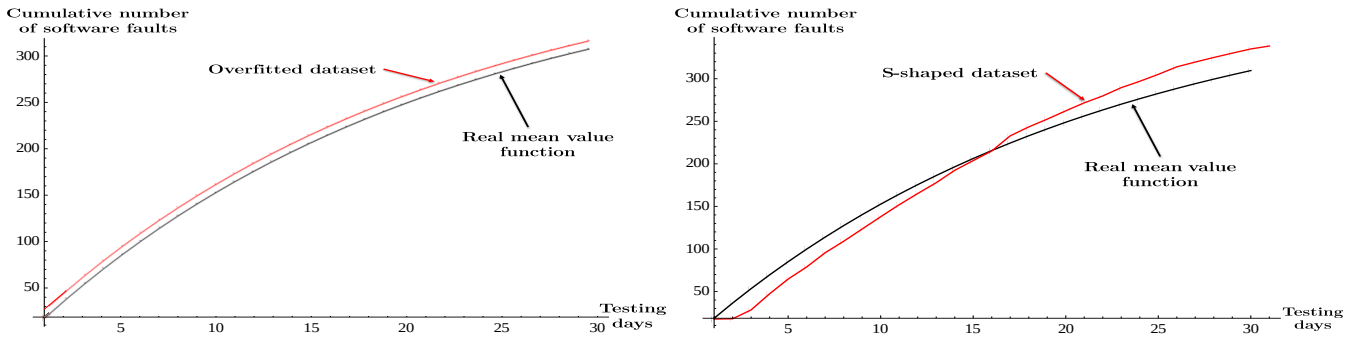
[17] Y.-S. Su and C.-Y. Huang, "Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models," *Journal of Systems and Software*, vol. 80, no. 4, pp. 606–615, 2007.

[18] S. Noekhah, A. A. Hozhabri, and H. S. Rizi, "Software reliability prediction model based on ica algorithm and mlp neural network," in *7th International Conference on e-Commerce in Developing Countries: with focus on e-Security*. IEEE, 2013, pp. 1–15.

[19] R. Mahajan, S. K. Gupta, and R. K. Bedi, "Design of software fault prediction model using br technique," *Procedia Computer Science*, vol. 46, pp. 849–858,

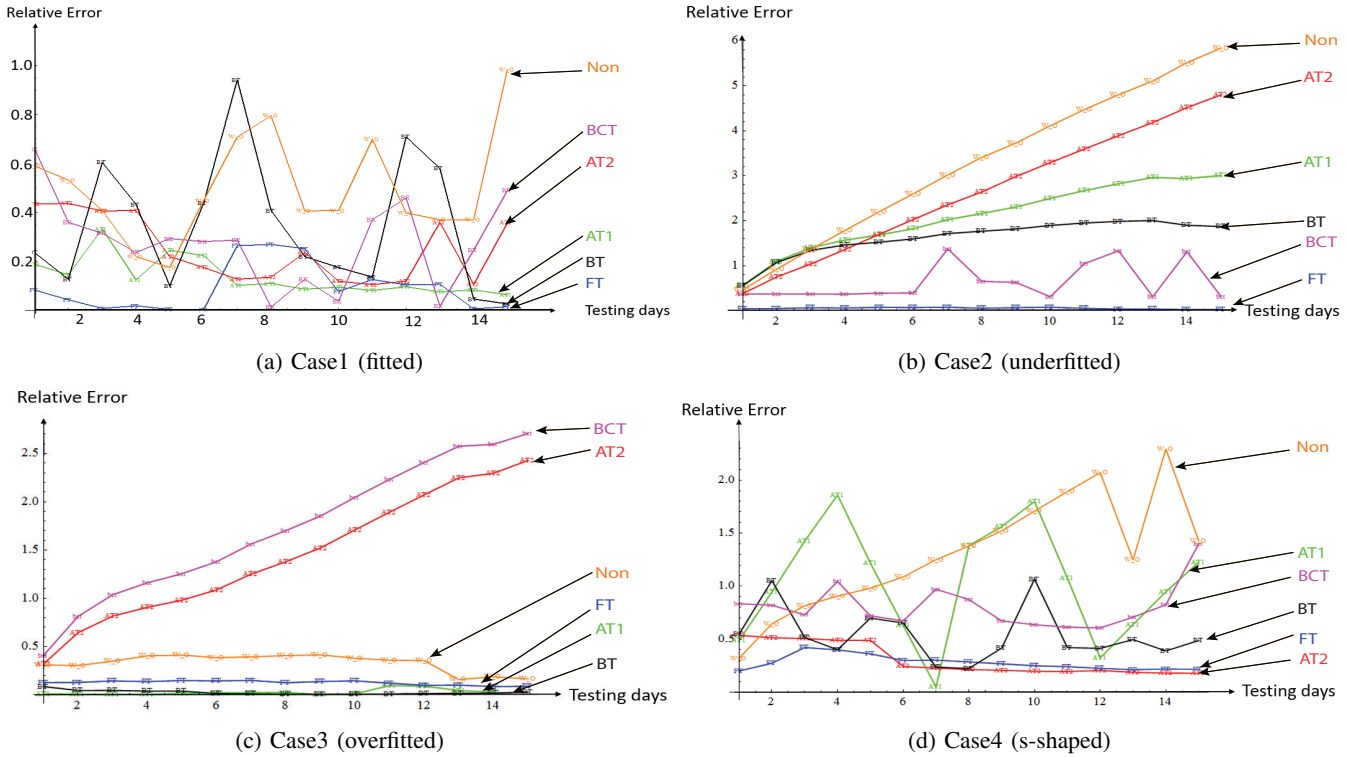


(a) Case1 (fitted) (b) Case2 (unfitted)



(c) Case3 (overfitted) (d) Case1 (S-shaped)

Fig. 6: Simulation datasets.



(a) Case1 (fitted) (b) Case2 (underfitted)

(c) Case3 (overfitted) (d) Case4 (s-shaped)

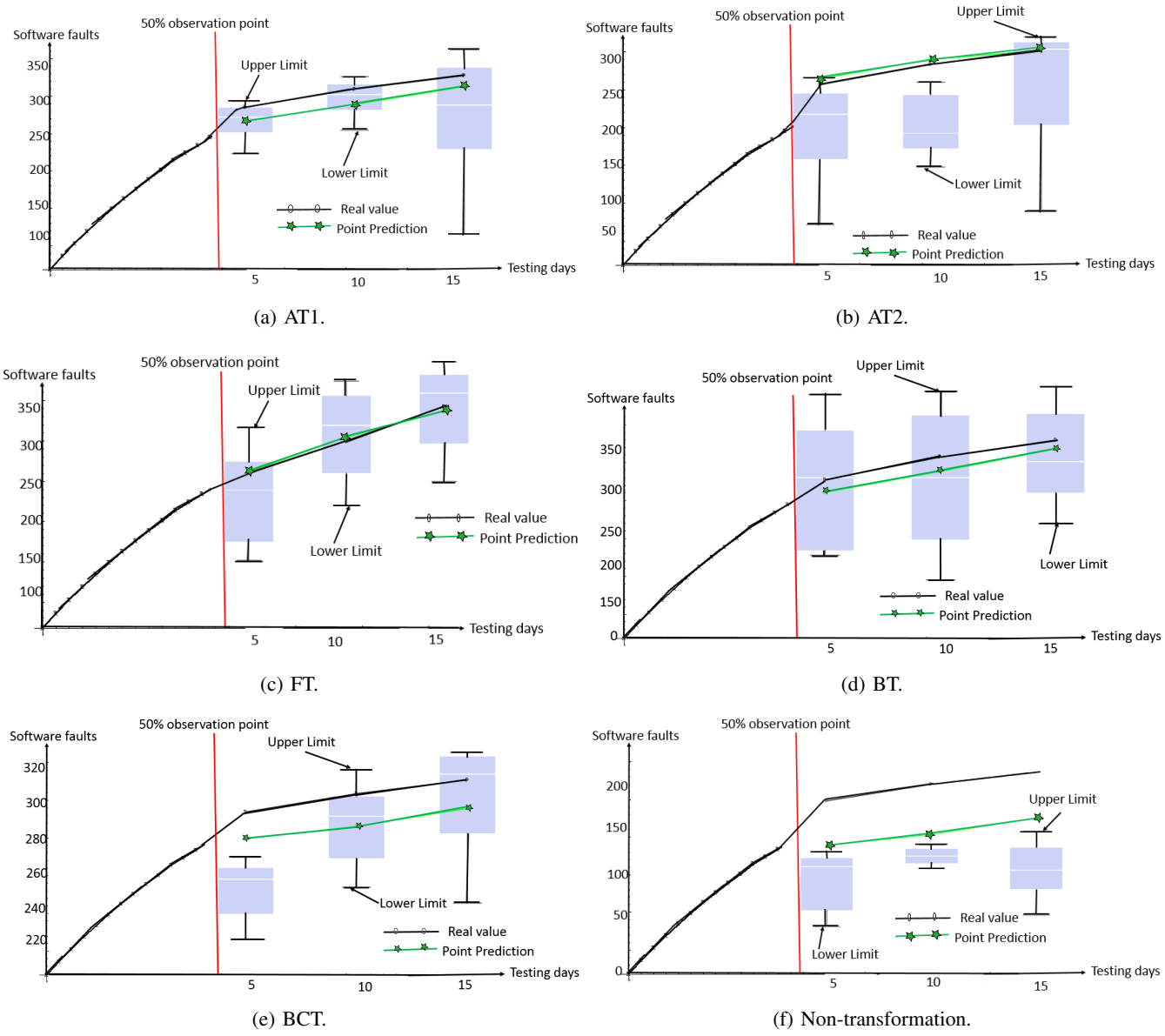
Fig. 7: Time-dependent behavior of relative error.

2015.

[20] Q. Hu, M. Xie, S. H. Ng, and G. Levitin, "Robust recurrent neural network modeling for software fault detection and correction prediction," *Reliability Engineering & System Safety*, vol. 92, no. 3, pp. 332–340, 2007.

[21] M. Begum and T. Dohi, "Optimal software release decision via artificial neural network approach with bug count data," *Advanced Reliability and Maintenance Modeling VII, McGraw Hill*, pp. 17–24, 2016.

[22] ——, "A neuro-based software fault prediction with box-cox power transformation," *Journal of Software*



(a) AT1.

(b) AT2.

(c) FT.

(d) BT.

(e) BCT.

(f) Non-transformation.

Fig. 8: prediction interval by delta method with Case1 ($l = 15$).

Engineering and Applications, vol. 10, no. 03, p. 288, 2017.

[23] ——, "Optimal release time estimation of software system using box-cox transformation and neural network," *International Iournal of Mathematical, Engineering and Management Sciences*, vol. 3, no. 2, pp. 177–194, 2018.

[24] ——, "Optimal stopping time of software system test via artificial neural network with fault count data," *Journal of Quality in Maintenance Engineering*, vol. 24, no. 1, pp. 22–36, 2018.

[25] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, and F. Törner, "Evaluating long-term predictive power of standard reliability growth models on automotive systems," in *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2013, pp. 228–237.

[26] H. Cheng, P.-N. Tan, J. Gao, and J. Scripps, "Multistep-ahead time series prediction," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2006, pp. 765–774.

[27] J. Park, N. Lee, and J. Baik, "On the long-term predictive capability of data-driven software reliability model: an empirical evaluation," in *2014 IEEE 25th International Symposium on Software Reliability Engineering*. IEEE, 2014, pp. 45–54.

[28] H. Okamura, T. Dohi, and S. Osaki, "Software reliability growth models with normal failure time distributions," *Reliability Engineering & System Safety*, vol. 116, pp. 135–141, 2013.

[29] A. L. Goel, "Software reliability models: Assumptions, limitations, and applicability," *IEEE Transactions on software engineering*, no. 12, pp. 1411–1423, 1985.

[30] K. Ohishi, H. Okamura, and T. Dohi, "Gompertz software reliability model: Estimation algorithm and empirical validation," *Journal of Systems and software*, vol. 82, no. 3, pp. 535–543, 2009.

[31] S. S. Gokhale and K. S. Trivedi, "Log-logistic software reliability growth model," in *Proceedings Third IEEE International High-Assurance Systems Engineer-*

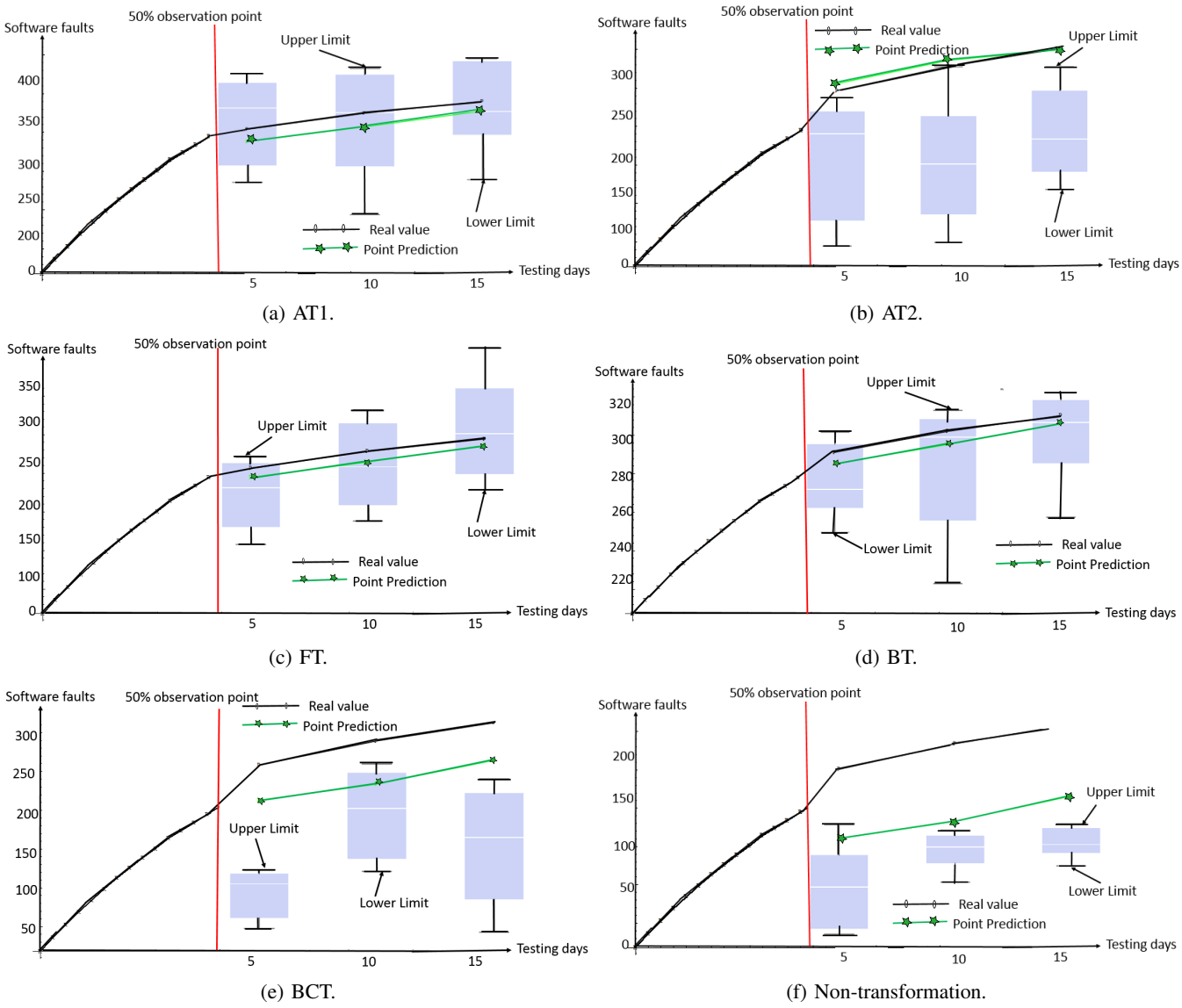


Fig. 9: prediction interval by simulation-based method with Case1 ($l = 15$).

ing Symposium (Cat. No. 98EX231). IEEE, 1998, pp. 34–41.

[32] M. Ohba, "Inflection s-shaped software reliability growth model," in *Stochastic models in reliability theory*. Springer, 1984, pp. 144–162.

[33] J. A. Achcar, D. K. Dey, and M. Niverthi, "A bayesian approach using nonhomogeneous poisson processes for software reliability models," in *Frontiers in reliability*. World Scientific, 1998, pp. 1–18.

[34] A. A. Abdel-Ghaly, P. Chan, and B. Littlewood, "Evaluation of competing software reliability predictions," *IEEE Transactions on Software Engineering*, no. 9, pp. 950–967, 1986.

[35] B. Littlewood, "Rationale for a modified duane model," *IEEE Transactions on Reliability*, vol. 33, no. 2, pp. 157–159, 1984.

[36] S. Yamada, M. Ohba, and S. Osaki, "S-shaped reliability growth modeling for software error detection," *IEEE Transactions on reliability*, vol. 32, no. 5, pp. 475–484, 1983.

[37] M. Zhao and M. Xie, "On maximum likelihood estimation for a general non-homogeneous poisson process," *Scandinavian journal of statistics*, pp. 597–607, 1996.

[38] A. L. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE transactions on Reliability*, vol. 28, no. 3, pp. 206–211, 1979.

[39] F. J. Anscombe, "The transformation of poisson, binomial and negative-binomial data," *Biometrika*, vol. 35, no. 3/4, pp. 246–254, 1948.

[40] M. Makitalo and A. Foi, "A closed-form approximation of the exact unbiased inverse of the anscombe variance-stabilizing transformation," *IEEE transactions on image processing*, vol. 20, no. 9, pp. 2697–2698, 2011.

[41] M. Bartlett, "The square root transformation in analysis of variance," *Supplement to the Journal of the Royal Statistical Society*, vol. 3, no. 1, pp. 68–78, 1936.

[42] M. Fisz, "The limiting distribution of a function of two independent random variables and its statistical application," in *Colloquium Mathematicum*, vol. 3, 1955, pp. 138–146.

[43] G. E. Box and D. R. Cox, "An analysis of transforma-

tions," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 26, no. 2, pp. 211–243, 1964.

[44] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "Comprehensive review of neural network-based prediction intervals and new advances," *IEEE Transactions on neural networks*, vol. 22, no. 9, pp. 1341–1356, 2011.

[45] R. D. De VlEAUX, J. Schumi, J. Schweinsberg, and L. H. Ungar, "Prediction intervals for neural networks via nonlinear regression," *Technometrics*, vol. 40, no. 4, pp. 273–282, 1998.

[46] P. W. Lewis and G. S. Shedler, "Simulation of nonhomogeneous poisson processes by thinning," *Naval research logistics quarterly*, vol. 26, no. 3, pp. 403–413, 1979.

[47] M. Begum, M. S. B. Hafiz *et al.*, "Supplementary material is added for detailed information on the comparison of average errors for datasets:1∼8," 2020. [Online]. Available: https://bit.ly/2CdIwxn