Received: February 22, 2024. Revised: May 1, 2024.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Imputation of Missing Data Using Masked Denoising Autoencoder with L2-Norm Regularization in Software Effort Estimation

Robert Marco[1]* **Sharifah Sakinah Syed Ahmad**[2]

[1]*Department of Informatics, Universitas Amikom Yogyakarta, Yogyakarta, Indonesia*
[2]*Faculty of Information & Communication Technology, Universiti Teknikal Malaysia Melaka, Malaysia*
*Corresponding author's Email: *robertmarco@amikom.ac.id*

Abstract: A frequent problem in building initial software effort estimation (SEE) models is the existence of many missing values in historical software engineering datasets. Due to human intervention, this is caused by frequent damage to software project data. Loss of information and bias in data analysis due to missing data are serious problems. This study proposes a method to estimate missing data using a masked-denoising autoencoder (Masked-DAE) with L2-norm regularization, which can handle various types of data, missing patterns, proportions, and distributions. In this study, Cocomo81 and ISBSG-IFPUG datasets from open-source repositories were used. This experiment involved five missing data techniques, eight missing data rates (from 10% to 80%), and two missingness mechanisms (MCAR: missing completely at random and MNAR: missing not at random). The results show that the proposed Mask-DAE method has the best imputation performance in terms of imputation errors by outperforming DAE, k-nearest neighbor imputation (kNNI), random forest (RF) imputation, multiple imputations by chained equation (MICE), mean imputation and mode imputation. We find that the prediction error rate increases with the rate of missing data. Furthermore, prediction errors generated by MCAR mechanisms are lower than those generated by MNAR. Nevertheless, our method can reduce the model variance, which results in lower generalization error.

Keywords: Software effort estimation, Missing data imputation, Denoising autoencoder, Missingness mechanisms.

1. Introduction

The problem in building an initial software effort estimation (SEE) model is that historical software engineering data sets often contain large amounts of missing values. This is because many collected software projects are typically corrupted due to human intervention [1]. In software engineering, the loss of information and bias in data analysis caused by missing data is a serious issue [2], which can cause a decrease in the performance of the algorithm used [3]. It often leads to errors in training and model analysis and negatively impacts the quality of the learning process, resulting in biased inferences if neglected [2, 4].

In the meantime, understanding the mechanism of missing data and the associated missing rates is crucial for comprehending the impact of missing data on a specific analysis or method that involves handling missing data [5, 6]. On the other hand, there are three approaches to solving problems with missing data, including tolerance, deletion, and the imputation technique [5, 7, 8].

The tolerance method is an embedded strategy where the analysis is performed directly on a dataset containing missing data [5, 8]. Despite its apparent ease of use, the tolerance method is not a dependable approach and, at times, produces a less accurate estimate than the deletion procedure [5, 6, 8]. In contrast, practitioners find delete techniques most enticing due to their simplicity. However, they have several disadvantages after deleting valuable data, including a loss of precision and result bias [8]. Meanwhile, Huang et al. (2017) stated that listwise deletion is becoming less common because it reduces data completeness, making it less suitable for applying derived models [9].

The missing data imputation technique replaces the missing value with an appropriate estimate by utilizing the available data and connecting the missing data, followed by the application of the standard complete data method for the filled data [5, 6, 8]. This makes the imputation method preferred and attracts the most research attention from academia and industry [2, 10]. Nevertheless, it's crucial to customize the imputation method according to the specific missingness mechanism present, as an inappropriate selection can introduce bias in performance [11].

Several studies on SEE have offered various approaches to imputing missing data. These methods range from simply substituting missing values with column averages to more complicated imputations based on several different machine learning models and statistical techniques [2], for example, mean imputation [12], hot-deck imputation [13], random forest (RF) imputation [14, 15], k-nearest neighbor imputation (kNNI)[12],[16-18], and multiple imputations by chained equation (MICE) [15, 19]. Implicitly assumes that the imputation model fails to address errors or ambiguities in the imputation process [2] and exhibits an increasingly constrained capacity to capture highly nonlinear relationships [10]. In addition, several imputation models cannot handle mixed data types [20] and random missing data patterns [2].

Several generative methods rooted in deep learning attempt to address this issue by modeling the joint distribution of all features of the missing data simultaneously [21]. Deep learning techniques are a popular topic, but their application for imputation purposes has received less attention [11]. Denoising autoencoder (DAE) is an unsupervised learning method used for unlabeled data to restore missing data from noisy input [22]. Unfortunately, the generative approach suffers from the problem of overfitting when working on small data sets [23]. Meanwhile, this approach can be utilized to reduce process uncertainty and the influence of corruption [24] and can deal with more complicated data sets (higher number of samples and dimensions) [11]. Because of its ability to learn the representation of noisy data, DAE is of great interest in many fields.

This study aims to devise a DAE-based deep learning approach capable of autonomously acquiring latent representations and connections among intricate variables. The DAE was developed to generate a clear output from a noisy input. However, missing data may rely on unobservable latent representations within the input dataset space. As a result, the DAE can map our input data into a higher-dimensional subspace, enabling us to retrieve the missing information subsequently.

The subsequent sections of this paper are structured as follows: Section 2 reviews relevant research. Section 3 presents the methods/approach adopted in this paper. Section 4 presents our model and elucidates the experimental configuration. Section 5 presents the empirical assessment and presents the findings. Finally, Section 6 concludes with conclusions and offers recommendations for future work.

2. Related work

The issue in building an initial estimation model is that historical software engineering data sets often contain large amounts of missing values. The loss of information and bias in data analysis caused by missing data is a severe issue in software engineering [2, 3], which can cause a decrease in the performance of the algorithm used [3, 25]. If neglected, it often leads to errors in training and model analysis and negatively impacts the quality of the learning process, leading to biased inferences [2, 4, 25]. Thus, many approaches have been proposed to handle this problem.

For instance, they proposed a new technique, imputing missing data, based on the multi-spike neural network (IMD-SNN) learning method. This study uses the MonitorAr dataset with three attributes: humidity, temperature, and atmospheric pressure. The results show that IMD-SNN provides high prediction accuracy compared to I-MLP (Imputation-based multilayer perceptron) and I-kNN for three attributes with missing percentages (5%, 10%, 25%, and 50%). However, SNN requires less time for the training and testing process. Unfortunately, neural networks may struggle to capture underlying patterns in data effectively [26].

Ou et al. (2023) proposed a missing time series data interpolation method based on random forest and generative adversarial interpolation network (RF-GAIN). Use of public datasets from Western Reserve University in the United States. The results show that the RMSE of interpolation results based on RF-GAIN in cases of single-segment and multi-segment missing data is only 0.0157 (3%), 0.0386 (10%), and 0.0527 (20%) better than the random forest algorithm, generative adversarial interpolation network, and k-nearest neighbor. The advantages of the RF-GAIN algorithm are combined so that the interpolation results are close to the actual value. Unfortunately, this method has its computational complexity and is expensive [27].

Meanwhile, the study of Li et al. (2024) comprehensively compared the performance of 8 imputation methods (Simple imputation, regression imputation, expectation-maximization (EM), MICE, kNN, clustering imputation, random forest (RF), and decision tree (CART)) in each scenario in the study real-world cohort in Xinjiang, China. The results of the study, using a missing rate of 20%, showed that the most effective imputation methods were achieved by kNN (MAE: 0.2032, RMSE: 0.7438, AUC: 0.730, CI: 0.719-0.741) and RF (MAE: 0.3944, RMSE: 1.4866, AUC: 0.777, CI: 0.769-0.785). EM, CART, and MICE achieved the next best performance, while simple, regression and cluster imputation obtained the worst. kNN and RF showed superior performance and were more adept at imputing missing data in the predictive modeling of Cohort study datasets [28].

Psychogyios et al. (2023) propose a new missing value imputation method based on denoising autoencoders (DAE) with kNN for pre-imputation tasks. We used four Electronic Health Records (EHR) data sets, and the missing proportions were 10%, 20%, 30%, 40%, and 50% for the three missingness mechanisms. Our proposed deep learning approach performs better than baseline standards (such as Simple, kNN, RF, MICE, and GAIN), resulting in better imputation and predictive results. Unfortunately, this method is complex in managing missing data. This approach requires complex preprocessing steps and is computationally intensive to produce a correct data representation [29].

A denoising autoencoder (DAE) based on time series data representation was proposed by Huamin et al. (2020). This DAE was created by reconstructing the data with the help of a recurrence plot (RP) and a gramian angular field (GAF). Based on the experimental results using MSE values, assign values to the dataset of ECG200 (GAF: 0.0048; RP: 0.0037), Face all (GAF: 0.0134; RP: 0.0221), Swedish leaf (GAF: 0.0098; RP: 0.0092), OSU leaf (GAF: 0.0077; RP:0.0121), Wafer (GAF:0.0240; RP:0.0069), 50 words (GAF:0.0101; RP:0.0108), and Coffee were given values by the experimental results utilizing (GAF:0.0336; RP:0.0234). On the other hand, this approach is only helpful for univariate time series. It only applies to more complex multivariate time series [30].

Abnane et al. (2023) proposed and built 11 heterogeneous ensemble imputation techniques, whose members are from the following single imputation techniques: kNN, expectation maximization, support vector regression (SVR), and decision trees (DTs). Evaluated over six SDEE datasets from the PROMISE repository. Overall, SVR and DT imputations are the best techniques for constructing ensemble imputations. The results show that ensemble imputation significantly improves the performance of SEE techniques. However, no particular ensemble imputer provides the best results in all contexts [31].

Kim and Chung (2020) proposed a multi-modal stacked denoising autoencoder (MMSDAE) that aims to estimate missing data during the data collection and processing stages of the Korean National Health Nutrition Examination Survey (KNHNES). Our method yields higher accuracy at a missing rate of 5%-30% than other conventional methods (such as kNNI, singular value decomposition, and Mean). Unfortunately, at a missing rate of 25%, our method yielded a value of 0.9217. At the same time, the single modal denoising autoencoder (DAE) had an accuracy of 0.932, with a slight difference of around 0.01, which is within the limits allowed in data analysis. On the other hand, our MMSDAE model saves additional time when processing large amounts of data [32].

In the meantime, Tihon et al. (2021) proposed the DAE with mask attention (DAEMA) and released it in the UCI repository. This method outperforms other methods (such as DAE, MIDA, MissForest, Mean, and AimNet) currently used on multiple missing data samples under MCAR and MNAR. Unfortunately, when working with small datasets, the performance of these methods is

Table 1. List of notations

Notations	Description
$\hat{x}$	Input that has been corrupted
$c(\hat{x})$	Hidden representation
W'	The weight matrix that links the input and hidden layers
b'	The bias vector of nodes that are part of the hidden layer
$sig.$	Logistics activate function
z	Reconstructed vectors
c'	The bias vector of nodes belonging to the output layer
x	Original data
θ	Model parameter
$\emptyset$	Unidentified parameters
X^n	Missing values
X^o	Observed values
x_i^o ; x_i^n	The observed and missing features
$f(\cdot)$	User-specified activation function
$g(\cdot)$	User-defined activation function
$L(\cdot)$	Loss function
D	Dataset
M	Missing indicator matrix
X_n	Input variable dataset
y_n	Target variable dataset

relatively poor. Thus, DAEMA requires adequate data to predict the data distribution [21].

According to the author, this study differs from other research. Most previous research addresses missing data in the classification field using machine learning or deep learning approaches. Meanwhile, our study addresses missing data in the field of regression with a rate of missing data from 10% to 80%; two missingness mechanisms and a small data set are applied. An improved deep learning approach using masked-denoising autoencoder (Masked-DAE) with L2-norm regularization.

3. Background knowledge

In this section, we have discussed the introduction and studies related to the formulation of the problem and explained the list of notations used in this work. A list of notations is presented in Table 1.

3.1 Denoising autoencoder

The denoising autoencoder (DAE) follows a process similar to the autoencoder, with the distinction of introducing noise to the input data [33]. Vincent et al. (2010) designed the DAE to eliminate noise from data. This noise is characterized by high dimensions in hidden layers and stochastic input corruption [34]. The DAE reconstruction capability estimates the data distribution implicitly as an asymptotic distribution of a Markov chain that alternates between corruption and denoising [35], as shown in Figure 1.

Encoding is performed on the corrupted inputs, introducing noise to the original input data via random mapping [33].

$$c(\hat{x}) = sig(W'\hat{x} + b') \tag{1}$$

Where $\hat{x}$ represents the input that has been corrupted through stochastic mapping, $c(\hat{x})$ denotes the hidden representation, $sig.$ signifies the logistics activate function, W' represents the weight matrix that links the input and hidden layers, and b' indicates the bias vector of nodes that are part of the hidden layer.

The decoding process involves reverse mapping the $c(\hat{x})$ back into the original feature space.

$$z = sig(W'c(\hat{x}) + c') \tag{2}$$

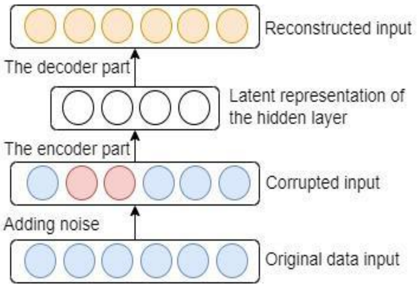


Figure. 1 Denoising Autoencoder Architecture

The reconstructed vectors are denoted as z, the weight matrix between the hidden layer and output layer is denoted as W', and the bias vector of nodes belonging to the output layer is denoted as c'.

In order to acquire the optimal DAE model, the reconstruction error can be minimized as follows:

$$\theta = \|x - z\|^2 \tag{3}$$

Where x represents the original data, z represents the reconstructed vectors, and θ refers to the encoder and decoder parameters, encompassing weight matrices and bias vectors.

3.2 Missingness mechanisms

Many methods for estimating missing values have been proposed in recent decades [11]. Missing data are indicated by null, NaN, n/a, or a blank space, which means free space for non-zero data [32]. Most missing data occur irregularly and differently, and deleting observations can lead to loss of information [6]. Comprehending the impact of missing data on a specific analysis or method entails understanding the mechanism behind the missing data [5, 6]. Missing data can be categorized into three groups: missing completely at random (MCAR) denotes a scenario where the absence of data is uncorrelated with all features and happens randomly. Missing at random (MAR) characterizes a scenario where the absence of data is not linked to the missing features but rather to other features. Missing not at random (MNAR) transpires when the lack of data is associated with the missing features and is not random [6]. Unfortunately, MNAR is the most difficult because it requires modeling the missing data mechanism [36].

This mechanism can be generated in various methods, and the literature provides numerous examples [37]. Huang et al. (2015) examined the mechanisms of missingness, making assumptions regarding the rates of missing data (10%, 20%, and 30%) across three mechanisms: MCAR, MAR, and non-ignorable (NI) [12]. Abnane et al. (2018) conducted 1134 experiments using seven datasets, employing kNN imputation methods with Euclidean and Manhattan distances. They investigated three types of missingness mechanisms (MCAR, MAR, and NI) across missing data rates ranging from 10% to 90% [3, 5, 38]. Abnane et al. (2021) investigated how the ensemble imputation technique affects the accuracy of the SEE method as measured by the standard accuracy criteria. They did this by comparing it to the single imputation technique. The findings indicate that the accuracy of the SEE method significantly improves when employing heterogeneous ensemble-based imputation as opposed to single imputation [39]. In the meantime, Jing et al. (2016) used a semi-supervised regression technique to impute the missing effort labels (10%, 20%, and 40%) without investigating the effect of the missingness mechanism [13].

3.3 Mask based missingness

Collier et al. (2020) introduces a latent variable model that perceives the observed data as a result of a corruption process affected by a binary missingness mask. This methodology can be applied to data sets with both ignorable and non-ignorable missingness. However, the choice of model architecture should be customized to suit the specific underlying missingness mechanism [36].

Let the data be decomposed into components such that $X = \{X^o, X^n\}$, with X^o represents the observed values and X^n is the missing values. For each observation vector x_i^o, x_i^n denote the observed and missing features of x_i, respectively. Additionally, consider R as a matrix of the same dimensions as X, where the entries $r_{ij} = I$ (x_{ij} is observed) for the ith observation and jth feature, with $I(.)$ representing the indicator function. Thus, R serves as the "mask" matrix corresponding to X, such that $x_i^o = \{x_{ij}: r_{ij} = 1\}$ and $x_i^n = \{x_{ij}: r_{ij} = 1\}$ for all $i = 1, \dots, n$ and $j = 1, \dots, p$ [40].

Missingness was classified into three major categories, or mechanisms, in the research by Little and Rubin (2002). These MCAR, MAR, and MNAR satisfy the following relations: (1) MCAR: $p(x_i, z_i, \emptyset) = p(r_i|\emptyset)$, (2) MAR: $p(x_i, z_i, \emptyset) = p(x_i^o|\emptyset)$, and (3) MNAR: $p(x_i, z_i, \emptyset) = p(r_i|x_i^o,\ x_i^n, z_i, \emptyset)$. In this context, $\emptyset$ represents the unidentified parameters associated with the missingness model $p(x_i, z_i, \emptyset)$, where $r_1 = \{r_{i1}, \dots, r_{ip}\}$. When considering the missingness mask R, the marginal log-likelihood can be expressed as:

$$log p_{\psi,\phi}(X^o, R) = log \iint p_{\psi,\phi}(X^o, X^n, Z, R) dX^n dZ \qquad (4)$$

4. Experiment design

4.1 Problem formulation

Our method is an easy way to generate data imputation based on a project D in the context of SEE. Consider a training set of N software projects. Then, we can denote Equation (5).

$$D = \{(X_n, y_n)\}_{n=1}^{N} \qquad (5)$$

Consider a typical unsupervised learning setting with the training set according to Equation (5). $X \in R^{n\times d}$ is a data matrix arranged with n (input, target) pairs $D_n = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ and d includes device features software development such as the type of software development, team expertise and functional measures consisting of numerical and categorical data types, then $x_i = (x_{i1}, x_{i2}, \dots, x_{ij}, \dots, x_{id}) \in R^d$. Meanwhile, $y \in R^1$ is the actual effort to develop this software. In this work, categorical data is converted to numerical data using ordinal encoding to keep losses and metrics simple so that a data set contains only numeric features.

Consider an *i.i.d* (independent and identically distributed) sample of the unknown distribution $q(X, y)$ with corresponding marginals $q(X)$ and $q(y)$. Express the $q^0(X, y)$ and $q^0(X)$ empirical distributions determined by the sample in D_n. X is a random vector with d-dimensions, and its R^d ordinal features typically have discrete ordinal values such as low, normal, high, very high, and extra high for programmers capability. On the other hand, its numerical features typically have continuous values such as lines of code. X can also be represented as $[0,1]^d$ (categorical features, such as the networked, mainframe, personal computer, mini computer, and multi-platform).

On the other hand, it uses a complete dataset of two public datasets in the context of SEE. Thus, our data set will be assigned to X using the MCAR and MNAR mechanism [41], meaning the probability that a missing value does not depend on any value in the data set. It can be defined in the missing rates matrix, given the dataset X the missing indicator matrix is denoted by $M \in (0,1)^{n\times d}$ for $M = \{m_1, m_2, \dots, m_i\}$ to show the missing value in X, where the ith vector $m_i = \{m_{1j}, m_{2j}, \dots, m_{ij}\}$ corresponds to the observation x_{ij}. In such a way that, x_{ij} missing if and only if $m_{ij} = 0$, otherwise if not missing $m_{ij} = 1$. State that $D^* = (X_n^*, y_n), X_n^* \in R^d$ is a basic truth dataset with no missing data.

The use of dataset D was accomplished by establishing the suitable missing rates matrix M (10% to 80%). As a result, the imputation function can be defined in accordance with the equation:

$$f: R^d \times (0,1)^d \rightarrow R^d = (x, m) \rightarrow f'(x, m) \quad (6)$$

Implementing this function (f') is to find the best function by minimizing the reconstruction metric.

4.2 Inducing missingness

Initially, it introduces missingness in two distinct ways for each data set. Generally, our goal for all scenarios is to generate a missing proportion between 10% and 80% by altering the tuning probabilities by the below-described stages:

- Add a uniform vector v with n observations and values ranging between 0 and 1 into the dataset, where n represents the total number of observations in the dataset.
- MCAR with a uniform distribution, ensure that all attributes contain missing values, where $v_i \leq t, i \in 1{:}n, t$ represents the missing threshold, This threshold varies from 10% to 80%.
- MNAR with a uniform distribution, take a random sample of two attributes from the dataset, x_1 and x_2, then calculate the median of each of those attributes, m_1 and m_2. Set all attributes to be missing values where $v_i \leq t, i \in 1{:}n$ and $(x_1 \leq m_1 \; or \; x_2 \geq m_2)$.

4.3 Model ratining scheme

The experiment started by collecting two publicly available real-world data sets (Cocomo81 and ISBSG-IFPUG datasets) to analyze the effects of missingness mechanisms and inducing missingness on imputation methods. All data sets in this experiment contain complete datasets (without missing values). Because the data set includes a mixture of numeric and categorical, it uses the ordinal encoding technique to convert categorical data to numerical to keep losses and metrics simple. On the other hand, min-max standardization can also be applied to the input data for faster convergence. As a result, normalization ensures that all inputs are contained within a range that is comparable to one another [42].

Several of the initial datasets underwent preprocessing to eliminate instances containing only a small number of missing values. Subsequently, we generated missing data by introducing missing values at eight distinct levels (ranging from 10% to 80% dataset missing rates), adhering to both MCAR and MNAR mechanism. This was achieved using a state-of-the-art generation method employed in our experiments.

In this study, missing x_{ij} values in the dataset will be represented as "NaN". Additionally, the matrix $X \in R^{n \times d}$ is utilized to determine whether the value in x_{ij} is missing. X components can be described as:

$$X = \begin{cases} 0 & x_{ij} = NaN \\ 1 & otherwise \end{cases} \quad (7)$$

Assume dataset X is represented by an $n \times d$ matrix, where $i = 1, \ldots, n$ as a pattern and $j = 1, \ldots, d$ as attributes. The elements of X are represents by x_{ij}, each individual feature in X is represents by x_j and each pattern is referred to as $x_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{id})$. It uses the multivariate MCAR and MNAR configuration. Where MCAR and MNAR select all attributes with missing values, x_{ij} in the dataset will be missing until the desired rate of between 10% and 80% is reached.

In this section, the framework is the DAE-based missing value imputation method. Assume x_{ij} is a basic truth. That is, there is no missing value in x_{ij}. Then, the missing value is denoted by $\hat{x}_{ij}$. Before to data entry, the DAE randomly selects a value from the original dataset and transforms it to 0. In neural network training, missing values are commonly represented as 0. When missing data, the noise is computed as 0 and returned to the original data. Consequently, a value of 0 is inserted for missing data, which is then replaced by a non-zero predicted value obtained through a trained neural network. To enhance the robustness of the learned model and prevent overfitting, the DAE distorts the original input x to $\hat{x}$ by introducing additional slight noise (isotropic Gaussian noise) [10].

The encoder mapped the corrupted input $\hat{x}$ to the h-dimensional of the hidden representation (embedding) $y = f(W\hat{x} + b)$. Where $f(\cdot)$ is the user-specified activation function. W is the coding of the weight matrix $d * h$, while b is the bias vector for coding h. Finally, the y insertion result is remapped to reconstruct x original input via the decoder. The transformation function is similar to $z = g(W'y + b')$, where $g(\cdot)$ is a user-defined activation function.

The primary goal of the DAE is to minimize the reconstruction error between the initial input x and

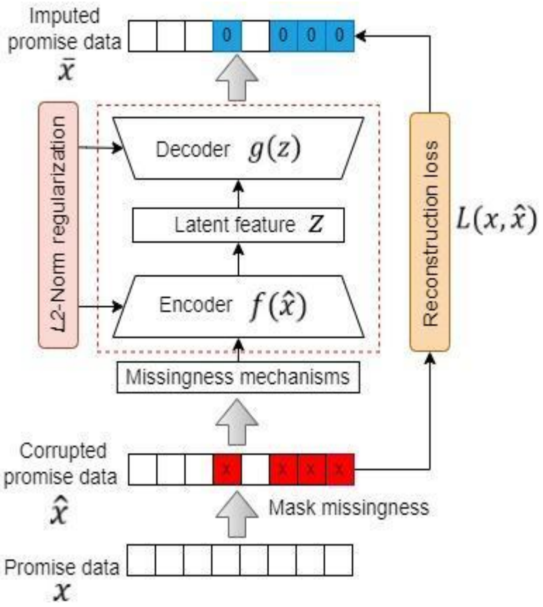


Figure. 2 Proposed Masked DAE L2-norm regularization model

the reconstructed output $z \to arg\,min_{\theta} L(x,z)$, where $\theta = \{W, W', b, b'\}$ represents the optimization parameters, and $L(\cdot)$ denotes a loss function used to quantify the discrepancy between the input x and z. It's worth mentioning that the output z generated by the DAE is a deterministic outcome of the corrupted input $\hat{x}$, not the original input x. The optimization of parameters aims to ensure that if the embedding y effectively captures the crucial features of the original input x from its corrupted version $\hat{x}$, it enables the reconstruction z of the original input x.

The proposed framework of the model can be seen in Figure 1.

4.4 Hyperparameter setting

The proposed Masked-DAE model, incorporating L2-norm regularization, requires a tailored set of hyperparameters to effectively address the specific problem, adjusting according to the extent of corruption within the dataset. To enhance the generalization performance of the deep neural network (DNN), the embedding dimension of the hidden layer is set to 128 for all training benchmarks, ensuring a comprehensive representation of relevant datasets. As DNNs are proficient in feature extraction, the number of hidden units is gradually reduced to mitigate dimensionality issues. Training is conducted for 100 iterations (epochs) with an early stop strategy to monitor reconstruction losses over known elements. Given the limited dataset in our study, L2-norm regularization parameters are set to $1{\times}10^{-4}$, accompanied by a dropout probability of 0.5 to prevent overfitting. However, excessive L2 regularization may lead to overweighting and suboptimal fitting. Each autoencoder optimization phase involves training the simple 3-layer feed-forward network 10 times. Using the Adam optimizer in TensorFlow, the default learning rate for both networks is set to $1{\times}10^{-3}$ for the DAE. Rectified linear unit (ReLU) activation functions are utilized in the hidden layers to address the issue of vanishing gradients associated with sigmoid functions [43, 44]. Additionally, random Gaussian noise of 0.2 is added at each time step of the training data progression to mitigate noise-related challenges throughout the training process [45]. Mean squared error (MSE) loss was used for each internal pretraining autoencoder's output layer.

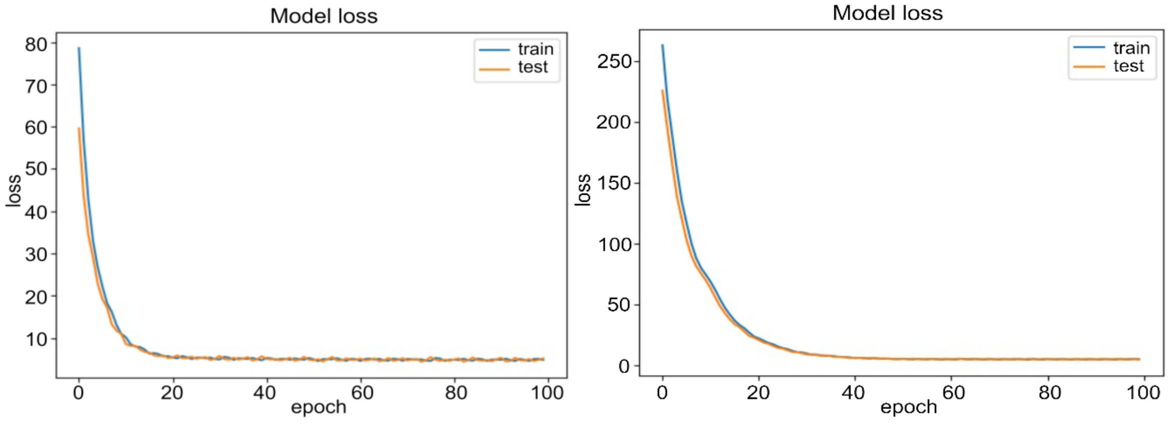
This study has compared six advanced techniques, including DAE [2], MICE [46], RF imputation [14], kNNI [5], mean imputation, and mode imputation, to validate our methodology. kNNI connects incomplete patterns according to the value of kNN [11, 47]. For kNNI, consider Euclidean distances and a set of nearest neighbors [11]. The kNNI parameter is set as k=5 due to low imputation error within an acceptable timeframe and being able to reconstruct the complete data set at once [48].

MICE, a multiple imputation method, creates separate conditional models for each missing data item [49]. The number of iterations for each imputation is established as two prominent hyperparameters in MICE. These parameters are set according to the convergence criterion of the model parameters. RF imputation mainly targets repair datasets. RF imputation uses a maximum of ten iterations (n=10) and 100 estimators with no leaf-node limitations. For the RF imputation approach, the number of iterations and trees utilized in the ensemble are specified [50]. MICE and RF imputation have the same maximum number of iterations.

5. Result and discussion

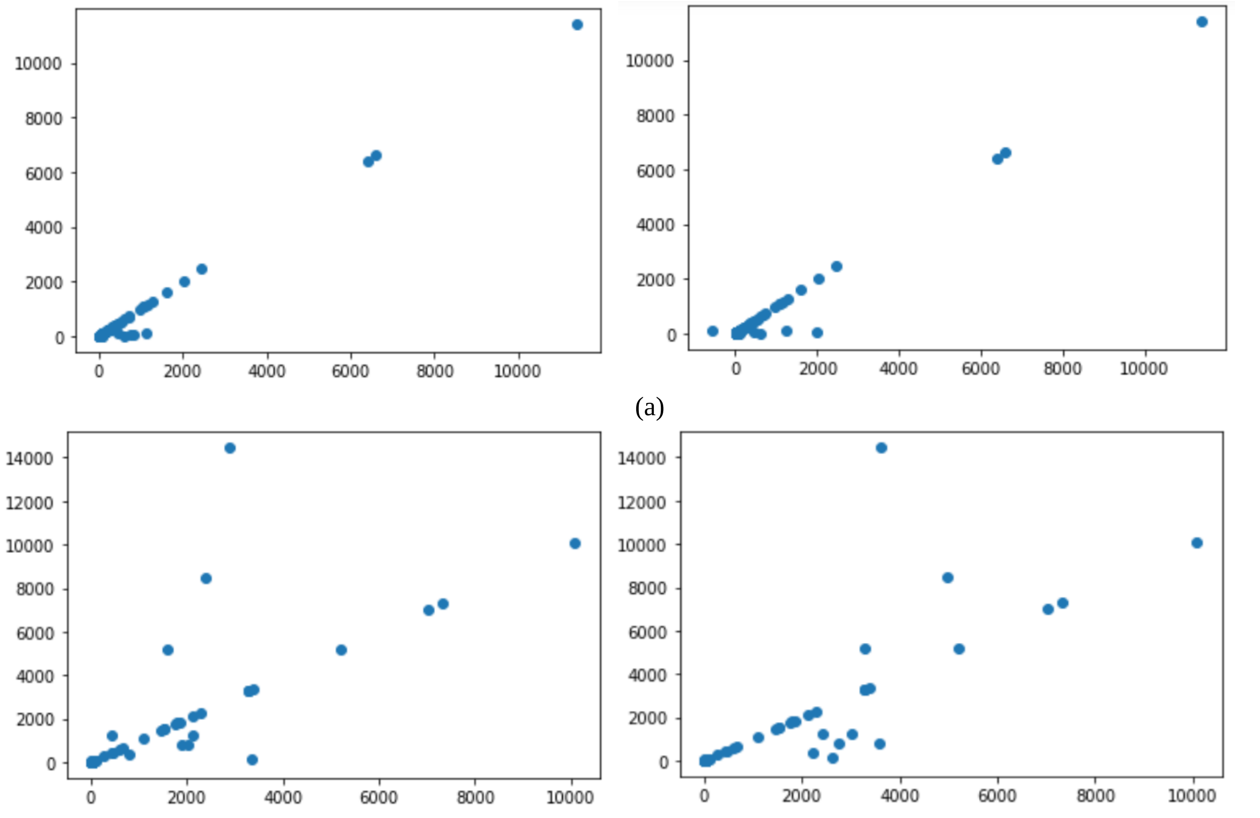
5.1 Convergence of model loss analysis

The simulation is carried out on previously processed data with a batch size of 128. Figure 3 illustrates the Masked-DAE learning process based on training and validation losses with the best configuration of the model. It can be seen that the loss values converge rapidly within 100 iterations. A gradual increase in the accuracy of the training and



(a) (b)

Figure. 3 Validation Loss During Training of Masked DAE L2-Norm Regulization Models: (a) Cocomo81 dataset and (b) IFPUG dataset



(a)

(b)

Figure. 4 Scatterplot of Dataset Showing Outliers: (a) The MCAR (left) and MNAR (right) for Cocomo81 data and (b) The MCAR (left) and MNAR (right) for IFPUG data

validation data is depicted in the first 10-20 epochs. There is a rapid decrease in the training and validation data loss, and a constant curve is observed. This shows the smooth learning process of the Masked-DAE model, and there will be no overfitting until the 100th epoch. The Masked-DAE model performs best and reduces the training and validation losses to a close of 0. This indicates a gradual convergence of training and validation losses, indicating that the time complexity of Masked-DAE is very low. The optimal global solution can be obtained when the resulting sample distribution is the same as the actual data distribution. This confirms that our model does not "overfit" the entire data set.

This discovery suggests that the Masked-DAE model effectively minimizes the expected loss across the empirical distribution of observed data and a subset of previously observed corrupted data, thereby mitigating potential biases.

Using an autoencoder learning approach that is unmodified during fine-tuning will result in high processing costs and decreased learning efficiency due to the large number of parameters [32]. Consequently, customization necessitates a computationally efficient strategy. In this instance, the training data must be reconfigured depending on the workload. When constructing the autoencoder using dropout techniques, numerous parameters are considered to get superior outcomes over non-dropout procedures. Encoder-decoders can learn the regression data pattern more quickly and produce better performance in preventing overfitting, which will be observed if the NN is deeper when the dropout strategy is applied.

5.2 Outlier detection evaluation

See the Figure 4, the distribution of outliers in the dataset using a scatterplot. Our method performs better in detecting outliers in the missingness mechanism of MCAR than MNAR.

At least one variable in each data set exhibited outliers, consistent with earlier research on this topic, which found that outliers are a regular occurrence in empirical software engineering datasets [51]. In particular, there are outliers in the distribution of effort values for data sets. However, there are only a small number of outliers in our imputed data, as slight deviations may not represent a significant assumption violation.

5.3 Comparison with existing methods

Our study involved a phase in which missing values were intentionally created, followed by imputation. The results are categorized based on metrics such as mean absolute error (MAE), root mean square error (RMSE), and two missingness mechanisms, MCAR and MNAR. The configuration type is uniform, and the missing rate (MR) ranges from 10% to 80%. Table 2 illustrates the average outcomes for all datasets utilized in this analysis. As anticipated, an increase in the missing rate correlates with a decline in imputation quality (as indicated by MAE and RMSE). The method employed by the researcher is evaluated by assessing the point prediction performance using MAE and RMSE, with the best values highlighted in bold. In contrast, the poorer values are indicated in italics.

Table 2 on the Cocomo81 dataset, it can be explained that our Masked-DAE method has the lowest MAE and RMSE values (presented sequentially), which shows that it is the method that has the best performance on the MCAR mechanism of 3.306; 48.385 (MR 10%) and MNAR mechanism 2.724; 30.274 (MR 20%). Meanwhile, RF in the MNAR mechanism is 0.686; 10.648 (MR 10%), and in the MCAR mechanism, 5.303; 64.881 (20%). On the other hand, at MR (30%), the Mode imputation has the best performance in all mechanisms. At MR 70%, our Masked-DAE method performs best on the MNAR mechanism of 24.675; 214.988, while the Modus imputation on the MCAR mechanism was 22.561; 238.603. In contrast to MR 40%, 50%, 60%, and 80%, our Masked-DAE method performs best in all mechanisms. kNNI has the worst performance at MR 80%. In contrast, Mode imputation performs poorly at MR 10%, while Mean imputation performs poorly at MR 20%, 40%, and 50%. Meanwhile, MICE at MR is 10%, 20%, 30%, 70%, and 80%. Finally, RF was worst at MR at 30%, 50%, 60%, 70%, and 80%.

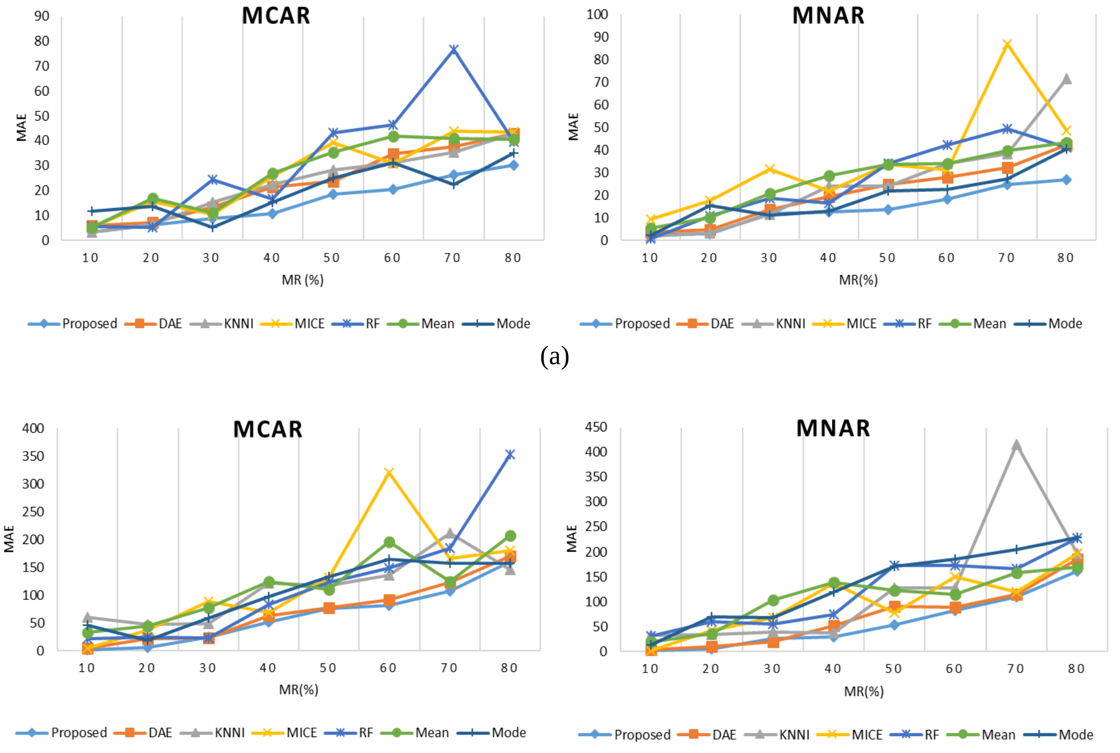
The IFPUG dataset shows that MR 10%, 20%, 40%, 50%, 60%, and 70% show that our Masked-DAE method performs best (for MAE and RMSE values) on the MCAR and MNAR mechanisms. Meanwhile, at MR 30%, the best performance of our Masked-DAE method is on the MNAR mechanism with an RMSE value of 211.665 and RF on the MCAR mechanism of 186.406. Meanwhile, the baseline model DAE performed well at MR 30% with MAE values of 22.597 (MCAR) and 19.507 (MNAR). On the other hand, at MR 80%, the method we propose has the best performance on MNAR with an MAE value of 161.195 and a mean RMSE value on MNAR of 741.625. On the other hand, kNNI has the best performance at MR 80% in the MCAR mechanism with a value of 145.383; 815.389. Considering the MCAR and MNAR mechanisms, our Masked-DAE is the best method in almost all scenarios. kNNI is also the best approach at the highest MR of 80%, although it has the worst performance at MRs of 10%, 20%, and 70%. In contrast, Mode imputation performs poorly at MR 50%, 60%, 70%, and 80%, while Mean imputation performs poorly at MR 40% and MICE at MR 30% and 60%. Finally, the RF is worst at MR 50% and 80%.

Table 2 summarizes the imputation performance of all methods compared across diverse datasets with various missing patterns. The objective of the imputation approach is to accurately recover missing data, ranging from minor to severe corruption. Therefore, this study anticipates that the erroneous distribution will align closely with the actual distribution after employing this strategy. The outcomes obtained from the proposed model demonstrate superior point prediction performance compared to those from the model discussed in the related work section. Specifically, our method

Received: February 22, 2024. Revised: May 1, 2024.

Table 2. The Comparison of Missing Data Imputation Methods based on MAE and RMSE Values

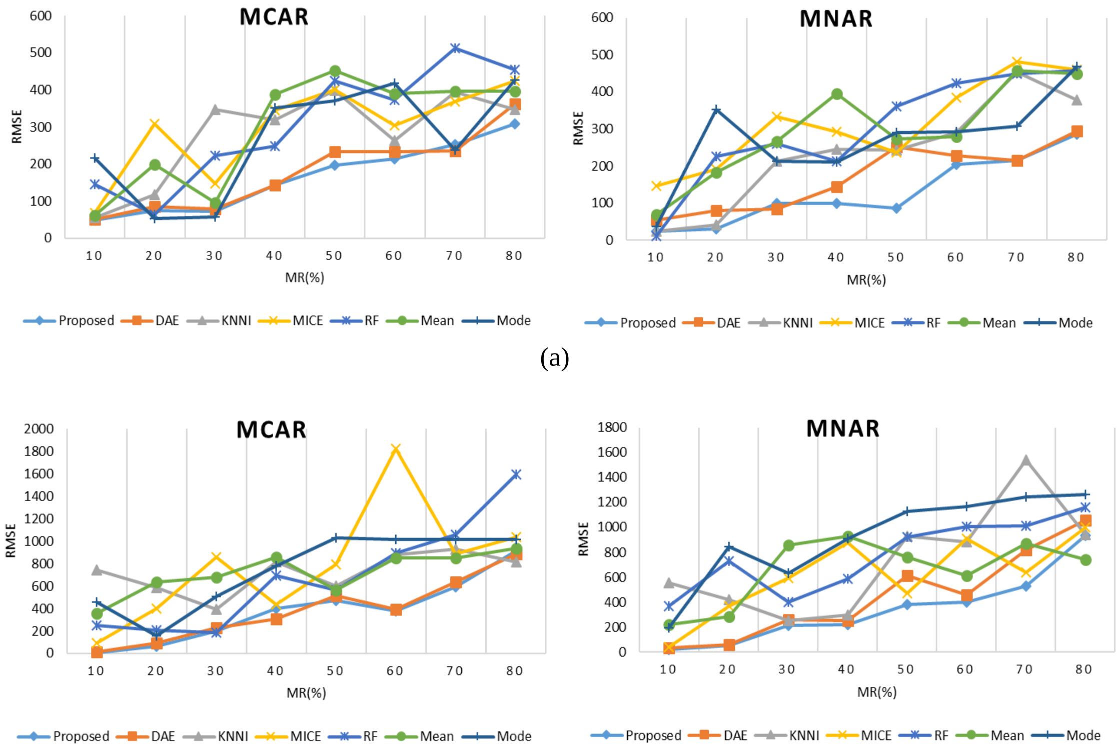
MR	Methods	Cocomo81 dataset				IFPUG dataset			
		MAE		RMSE		MAE		RMSE	
		MCAR	MNAR	MCAR	MNAR	MCAR	MNAR	MCAR	MNAR
10%	Proposed	**3.306**	2.404	**48.385**	24.270	**0.796**	**1.520**	**9.101**	**19.056**
	DAE	5.890	3.684	52.026	53.627	4.673	4.151	11.445	35.199
	kNNI	3.445	1.987	55.820	25.231	*60.155*	*32.317*	*744.333*	*556.344*
	MICE	5.311	*9.189*	67.972	*145.959*	4.975	2.293	93.353	40.036
	RF	5.644	**0.686**	144.833	**10.648**	20.753	31.598	249.505	368.768
	Mean	5.187	5.519	62.723	70.322	32.210	20.257	353.618	220.897
	Mode	*11.730*	2.010	*216.394*	37.977	45.107	13.969	456.142	193.937
20%	Proposed	6.316	**2.724**	75.991	**30.274**	**5.739**	**5.540**	**65.729**	**51.976**
	DAE	7.271	4.701	85.636	80.038	22.136	9.474	93.956	59.663
	kNNI	6.646	3.365	118.592	41.158	*47.472*	33.916	586.159	418.784
	MICE	15.771	*17.442*	*308.598*	192.509	37.124	42.840	398.142	369.287
	RF	**5.303**	10.685	**64.881**	224.990	23.815	60.321	207.756	728.972
	Mean	*16.898*	10.034	199.959	182.011	44.996	35.466	*636.375*	283.311
	Mode	13.750	15.223	54.045	*352.547*	18.443	*69.793*	157.387	*844.081*
30%	Proposed	8.678	11.834	72.741	**98.392**	24.676	26.392	202.184	**211.665**
	DAE	13.367	13.677	79.842	84.197	**22.597**	**19.507**	226.120	259.782
	kNNI	15.473	11.538	348.12	212.095	48.105	39.078	395.322	255.661
	MICE	10.419	*31.361*	147.685	*332.220*	*88.157*	67.967	*854.851*	593.036
	RF	*24.300*	18.599	*223.360*	261.175	22.946	54.685	**186.406**	403.866
	Mean	11.211	20.886	96.467	267.619	76.640	*103.846*	681.657	*857.599*
	Mode	**5.202**	**11.261**	**57.370**	213.097	58.658	68.636	508.257	631.472
40%	Proposed	**10.617**	**12.660**	**143.815**	**98.800**	**52.143**	**29.525**	**398.964**	**219.776**
	DAE	21.515	19.447	143.362	143.509	62.720	52.004	310.034	255.163
	kNNI	22.567	23.829	318.633	244.953	121.985	37.617	811.585	295.468
	MICE	25.697	21.905	345.227	292.651	69.073	137.426	438.957	879.304
	RF	16.548	16.298	248.651	212.620	82.895	74.661	692.553	586.787
	Mean	*27.117*	*28.450*	*387.962*	*395.681*	*124.070*	*138.314*	*859.921*	*925.915*
	Mode	15.279	12.891	352.234	211.058	96.630	119.925	782.265	912.196
50%	Proposed	**18.599**	**13.687**	**197.893**	**85.533**	**76.038**	**53.318**	**469.233**	**381.913**
	DAE	23.719	24.570	234.159	251.787	77.887	90.851	517.310	614.748
	kNNI	28.288	23.801	399.586	242.154	117.596	127.250	602.568	928.755
	MICE	39.360	33.760	401.452	237.054	131.128	78.384	792.996	472.419
	RF	*43.292*	*33.840*	424.769	*361.185*	122.832	*172.077*	562.737	924.259
	Mean	35.435	33.755	*452.248*	273.073	110.735	122.160	566.573	760.592
	Mode	25.099	21.707	370.625	290.342	*133.217*	171.049	*1027.238*	*1126.739*
60%	Proposed	**20.400**	**18.297**	**215.043**	**204.279**	**81.882**	**82.129**	**376.911**	**403.129**
	DAE	34.892	27.982	233.659	227.895	91.266	88.445	395.561	456.647
	kNNI	31.104	33.850	263.076	290.560	135.758	127.854	876.388	885.781
	MICE	30.940	31.198	304.300	383.954	*320.712*	149.847	*1826.202*	911.182
	RF	*46.371*	*42.163*	374.008	*423.405*	148.538	173.024	894.732	1007.413
	Mean	41.902	33.841	389.637	278.517	195.561	114.394	850.860	615.531
	Mode	31.217	22.643	*418.524*	292.401	165.013	*186.022*	1017.772	*1164.86*
70%	Proposed	26.260	**24.675**	253.586	214.988	**107.498**	**109.234**	**592.747**	**531.431**
	DAE	37.672	32.239	236.108	**214.103**	122.864	114.154	635.749	822.252
	kNNI	35.303	38.113	394.687	452.334	*211.032*	*415.663*	927.164	*1540.646*
	MICE	43.822	*87.003*	369.255	*481.355*	165.836	119.508	884.633	635.878
	RF	*76.450*	49.289	*513.202*	447.956	184.978	165.668	*1054.253*	1013.437
	Mean	41.066	39.527	396.057	456.852	124.740	158.471	853.492	868.519
	Mode	**22.561**	27.132	**238.603**	307.087	157.856	205.404	1016.635	1243.637
80%	Proposed	**30.119**	**26.728**	**307.702**	**286.112**	160.065	**161.195**	914.576	936.705
	DAE	42.729	42.255	362.108	294.103	169.632	185.339	886.235	1057.880
	kNNI	43.001	*71.347*	346.867	378.263	**145.383**	197.361	**815.389**	943.972
	MICE	*43.393*	48.784	425.472	458.618	180.631	197.056	1032.851	998.224
	RF	39.671	41.271	*454.728*	458.193	*353.463*	227.259	*1593.668*	1160.655
	Mean	40.728	43.346	396.021	448.179	207.079	168.663	938.500	**741.625**
	Mode	35.210	40.352	425.935	*468.506*	157.842	*229.432*	1012.731	*1261.187*



(a)

(b)

Figure. 5 The Simulation Results of the Comparison of Imputation Methods: MAE Values: (a) Comparison between MAE results obtained from two mechanisms on the Cocomo81 dataset and (b) Comparison between MAE results obtained from two mechanisms on the IFPUG dataset



(a)

(b)

Figure. 6 The Simulation Results of the Comparison of Imputation Methods: RMSE Values: (a) Comparison between RMSE results obtained from two mechanisms on the Cocomo81 dataset and (b) Comparison between RMSE results obtained from two mechanisms on the IFPUG dataset

exhibits the best performance in two project datasets at high missing rates, ranging between 40% and 80%. This study shows that the prediction error rate increases with the rates of missing data.

Meanwhile, the prediction error produced by the MCAR mechanisms is lower than that produced by the MNAR mechanisms. We find that kNNI increases with the percentage of missing data, regardless of the mechanism of missing data. However, our developed method can reduce model variance, resulting in lower generalization errors.

Our Masked-DAE method is more consistent and performs better than other popular imputation methods, regardless of metric. As observed in Table 2, the extent of performance disparity varies depending on the dataset and competing methods. Also, strong metrics should have little effect on outliers. The noisy RMSE distribution shifts more than the MAE ones. Thus, it is identified that the RMSE is less strong against outliers. On the other hand, MAE suffers from imprecision when the model error follows a normal distribution. However, MAE was also reported to have unbiased results and did not show an asymmetric distribution.

On the other hand, Figures 5 and 6 compare imputation methods using MAE and RMSE. In this case, it also observed that the measurement error rate for all models was presented in visualization to facilitate observing the performance of the imputation methods.

Findings from two real-world datasets indicate the applicability of the proposed method, which can be used on numerical and categorical data. Moreover, the DAE approach can avoid multiple computations, local minimums, and missing gradient problems [32], and using backpropagation techniques with various propagation and optimizer functions makes learning easier [52]. Conversely, L2-norm regularization serves to decrease the variance of the model, thereby leading to a reduction in generalization error. Moreover, dropout reduces overfitting when training limited data sets [53]. This explains why our model has the highest accuracy rate among the popular approaches utilized in this study. Our method can perform much better in almost all datasets with small set sizes. Even with adding the value, the proportion of missingness increases, indicating that our method appears less affected by the missing rates. This gives a profit for a high missing rate. However, for missing data that has an extremely high rate, our method takes a lot of time to calculate and has a higher computational cost.

The RF method provides an inbuilt procedure for dealing with missing values by weighing the frequency of values observed in the variable with the random forest closeness after being trained on the calculated mean initial data set [54]. On the other hand, this method requires a complete set of response variables to train the forest [14]. The kNNI method offers straightforward computation. Consequently, it proves particularly effective when applied to datasets characterized by less complex feature variance issues. This simplicity is similar to more basic techniques like mean imputation, where all missing values are replaced with the sample mean, potentially resulting in a significant reduction in variance [55]. kNNI has overlapping steps for evaluating similarities between software projects. MICE relies internally on simple regression, which may not be able to map complex patterns. MICE has no resistance to noise/outliers and is strongly affected by the missing rate. Other methods, such as mean and mode imputation, can change the data variance so that it can provide less efficient estimates.

6. Conclusion

In this paper, we enhance existing deep learning architectures by introducing improvements to the training procedure and its architecture. We leverage two comprehensive datasets in the SEE field to assess the proposed approach's efficacy. In our investigation, various missing rates and missingness mechanisms patterns are applied. The results indicate that the introduced Masked-DAE with L2-Norm Regularization approach outperforms others. However, at an 80% missing rate, the kNNI method exhibits the best performance on the IFPUG dataset. Findings from this study show that the prediction error rate increases with the rate of missing data.

Furthermore, prediction errors generated by MCAR mechanisms are lower than those generated by MNAR mechanisms. We find that kNNI improves with the percentage of missing data regardless of the missingness mechanism. Overall, the error of the missing data technique is influenced by the missingness mechanism. Nevertheless, our method can reduce the model variance, which results in lower generalization error.

Further work could be developed in several directions, such as creating a real-time missing data imputation framework based on recurrent neural networks. Additionally, deep learning should be integrated with expert assessment approaches to assess the accuracy and validity of imputed data as a practical approach.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, R. Marco; methodology, R. Marco, and S. S. S. Ahmad; validation, R. Marco; formal analysis, R. Marco; investigation, R. Marco, and S. S. S. Ahmad; resources, R. Marco, and S. S. S. Ahmad; data curation, R. Marco; writing—original draft preparation, R. Marco, and S. S. S. Ahmad; writing—review and editing, S. S. S. Ahmad; visualization, R. Marco; supervision, S. S. S Ahmad; funding acquisition, R. Marco, and S. S. S. Ahmad.

References

[1] L. Song, "Learning to Cope with Small Noisy Data in Software Effort Estimation", *Thesis Doctor of Philosophy*, No. January, 2019.

[2] L. Gondara and K. Wang, "MIDA: Multiple imputation using denoising autoencoders", *Journal of Advances in Knowledge Discovery and Data Mining*, Vol. 10939, pp. 260–272, 2018.

[3] I. Abnane, M. Hosni, A. Idri, and A. Abran, "Analogy Software Effort Estimation Using Ensemble kNN Imputation", In: *Proc. of International Conf. On Software Engineering and Advanced Applications, Kallithea, Greece*, pp. 228–235, 2019.

[4] F. A. Amazal, A. Idri, and A. Abran, "Software development effort estimation using classical and fuzzy analogy: A cross-validation comparative study", *International Journal of Computational Intelligence and Applications*, Vol. 13, No. 3, pp. 1–19, 2014.

[5] I. Abnane and A. Idri, "Improved analogy-based effort estimation with incomplete mixed data", In: *Proc. of the 2018 Federated Conference on Computer Science and Information Systems, FedCSIS 2018*, Vol. 15, pp. 1015–1024, 2018.

[6] R. J. A. Little and D. B. Rubin, "The Analysis of Social Science Data with Missing Values", *Journal of Sociological Methods and Research*, Vol. 18, No. 2–3, pp. 292–326, 1989.

[7] J. Li, A. Al-Emran, and G. Ruhe, "Impact Analysis of Missing Values on the Prediction Accuracy of Analogy-based Software Effort Estimation Method AQUA", In: *Proc. of International Conf. on Empirical Software Engineering and Measurement (ESEM), Madrid, Spain*, pp. 126–135, 2007.

[8] A. Idri, I. Abnane, and A. Abran, "Dealing with missing values in software project datasets: A systematic mapping study", *Studies in Computational Intelligence*, Vol. 653, No. Mv, pp. 1–16, 2016.

[9] J. Huang, Y. F. Li, J. W. Keung, Y. T. Yu, and W. K. Chan, "An empirical analysis of three-stage data-preprocessing for analogy-based software effort estimation on the ISBSG data", In: *Proc. of International Conf. on Software Quality, Reliability and Security, Prague, Czech Republic*, pp. 442–449, 2017.

[10] Q. Ma, W. C. Lee, T. Y. Fu, Y. Gu, and G. Yu, *MIDIA: exploring denoising autoencoders for missing data imputation*, Vol. 34, No. 6, 2020.

[11] A. F. Costa, M. S. Santos, J. P. Soares, and P. H. Abreu, "Missing data imputation via denoising autoencoders: The untold story", *Springer Nature Switzerland*, Vol. 11191 LNCS, pp. 87–98, 2018.

[12] J. Huang, H. Sun, Y. F. Li, and M. Xie, "An Empirical Study of Dynamic Incomplete-Case Nearest Neighbor Imputation in Software Quality Data", In: *Proc. of 2015 IEEE International Conference on Software Quality, Reliability and Security, QRS 2015*, No. Mi, pp. 37–42, 2015.

[13] X.-Y. Jing, F. Qi, F. Wu, and B. Xu, "Missing data imputation based on low-rank recovery and semi-supervised regression for software effort estimation", In: *Proc. of the 38th International Conference on Software Engineering - ICSE '16*, No. 1, pp. 607–618, 2016.

[14] D. J. Stekhoven and P. Bühlmann, "Missforest-Non-parametric missing value imputation for mixed-type data", *Journal of Bioinformatics*, Vol. 28, No. 1, pp. 112–118, 2012.

[15] C. Zhang, X. Cheng, J. Liu, J. He, and G. Liu, "Deep sparse autoencoder for feature extraction and diagnosis of locomotive adhesion status", *Journal of Control Science and Engineering*, Vol. 2018, 2018.

[16] L. L. Minku and X. Yao, "Ensembles and locality: Insight on improving software effort estimation", *Journal of Information and Software Technology*, Vol. 55, No. 8, pp. 1512–1528, 2013.

[17] W. Zhang, Y. Yang, and Q. Wang, "Using Bayesian regression and EM algorithm with missing handling for software effort prediction", *Information and Software Technology*, Vol. 58, pp. 58–70, 2015.

[18] J. Huang *et al.*, "Cross-validation based K nearest neighbor imputation for software

quality datasets: An empirical study" , *Journal of Systems and Software*, Vol. 132, pp. 226–252, 2017.

[19] R. A. Hughes, I. R. White, S. R. Seaman, J. R. Carpenter, K. Tilling, and J. A. C. Sterne, "Joint modelling rationale for chained equations", *BMC Medical Research Methodology*, Vol. 14, No. 1, 2014.

[20] S. Zhang, "Nearest neighbor selection for iteratively kNN imputation" , *Journal of Systems and Software*, Vol. 85, No. 11, pp. 2541–2552, 2012.

[21] S. Tihon, M. U. Javaid, D. Fourure, N. Posocco, and T. Peel, "DAEMA: Denoising Autoencoder with Mask Attention", *International Conference on Artificial Neural Networks*, Vol. 12891, pp. 229–240, 2021. [Online]. Available: http://arxiv.org/abs/2106.16057.

[22] M. Kampffmeyer, S. Løkse, F. M. Bianchi, R. Jenssen, and L. Livi, "Deep Kernelized Autoencoders", *Springer International Publishing*, Vol. 10270 LNCS, pp. 419–430, 2017.

[23] R. Lall and T. Robinson, "Applying the MIDAS Touch : An Accurate and Scalable Approach to Imputing Missing Data", *APSA Preprints*, 2020.

[24] C. Jia, M. Shao, S. Li, H. Zhao, and Y. Fu, "Stacked Denoising Tensor Auto-Encoder for Action Recognition with Spatiotemporal Corruptions", *IEEE Transactions on Image Processing,* Vol. 27, No. 4, pp. 1878–1887, 2018.

[25] S. K. Sehra, Y. S. Brar, N. Kaur, and S. S. Sehra, "Research patterns and trends in software effort estimation", *Journal of Information and Software Technology*, Vol. 91, pp. 1–21, 2017.

[26] N. A. S. Al-Jamali, I. R. K. Al-Saedi, A. R. Zarzoor, and H. Li, "A New Imputation Technique Based a Multi-Spike Neural Network to Handle Missing Data in the Internet of Things Network (IoT)", *IEEE Access*, Vol. 11, No. September, pp. 112841–112850, 2023.

[27] H. Ou, Y. Yao, and Y. He, "Missing Data Imputation Method Combining Random Forest and Generative Adversarial Imputation Network", *Sensors*, Vol. 24, No. 4, pp. 1112, 2024.

[28] J. Li *et al.*, "Comparison of the effects of imputation methods for missing data in predictive modelling of cohort study datasets", *BMC Medical Research Methodology*, Vol. 24, No. 1, pp. 1–9, 2024.

[29] K. Psychogyios, L. Ilias, C. Ntanos, and D. Askounis, "Missing Value Imputation Methods for Electronic Health Records", *IEEE Access*, Vol. 11, No. February, pp. 21562–21574, 2023.

[30] T. Huamin, D. Qiuqun, and X. Shanzhu, "Reconstruction of time series with missing value using 2D representation-based denoising autoencoder", *Journal of Systems Engineering and Electronics*, Vol. 31, No. 6, pp. 1087–1096, 2020.

[31] I. Abnane, A. Idri, I. Chlioui, and A. Abran, "Evaluating ensemble imputation in software effort estimation", *Empirical Software Engineering*, Vol. 28, No. 56, 2023.

[32] J. C. Kim and K. Chung, "Multi-Modal Stacked Denoising Autoencoder for Handling Missing Data in Healthcare Big Data", *IEEE Access*, Vol. 8, pp. 104933–104943, 2020.

[33] Z. Sun and H. Sun, "Stacked Denoising Autoencoder With Density-Grid Based Clustering Method for Detecting Outlier of Wind Turbine Components", *IEEE Access*, Vol. 7, pp. 13078–13091, 2019.

[34] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol, "Stacked denoising autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion", *Journal of Machine Learning Research*, Vol. 11, pp. 3371–3408, 2010.

[35] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized denoising auto-encoders as generative models", *Advances in Neural Information Processing Systems*, pp. 1–9, 2013.

[36] S. Ghalebikesabi, R. Cornish, L. J. Kelly, and C. Holmes, "Deep Generative Pattern-Set Mixture Models for Nonignorable Missingness", *arXiv* , 2021. [Online]. Available: http://arxiv.org/abs/2103.03532.

[37] A. J. H. Almutlaq and D. N. A. Jawawi, "Missing Data Imputation Techniques for Software Effort Estimation: A Study of Recent Issues and Challenges", *Advances in Intelligent Systems and Computing,* Vol. 1073, pp. 1144–1158, 2020.

[38] I. Abnane, A. Idri, and A. Abran, "Fuzzy case-based-reasoning-based imputation for incomplete data in software engineering repositories", *Journal of Software: Evolution and Process*, Vol. 32, No. 9, 2020.

[39] I. Abnane, A. Idri, M. Hosni, and A. Abran, *Heterogeneous ensemble imputation for software development effort estimation*, Vol. 1, No. 1, Association for Computing Machinery, 2021.

[40] D. K. Lim, N. U. Rashid, J. B. Oliva, and J. G. Ibrahim, "Handling Non-ignorably Missing

Features in Electronic Health Records Data Using Importance-Weighted Autoencoders", *arXiv:2101.07357*, pp. 1–25, 2021. [Online]. Available: http://arxiv.org/abs/2101.07357.

[41] L. Gondara and K. Wang, "MIDA: Multiple imputation using denoising autoencoders", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 10939 LNAI, pp. 260–272, 2018.

[42] Y. Wang, K. Li, S. Gan, and C. Cameron, "Missing Data Imputation with OLS-Based Autoencoder for Intelligent Manufacturing", *IEEE Transactions on Industry Applications*, Vol. 55, No. 6, pp. 7219–7229, 2019.

[43] Y. Yang, K. Zheng, C. Wu, and Y. Yang, "Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network", *Sensors (Switzerland)*, Vol. 19, No. 11, 2019.

[44] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on cifar-10", *Unpublished manuscript*, pp. 1–9, 2010. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Convolutional+Deep+Belief+Networks+on+CIFAR-10#0.

[45] C. Fan, M. Chen, R. Tang, and J. Wang, "A novel deep generative modeling-based data augmentation strategy for improving short-term building energy predictions", *Building Simulation*, Vol. 15, No. 2, pp. 197–211, 2022.

[46] S. van Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in R", *Journal of Statistical Software*, Vol. 45, No. 3, pp. 1–67, 2011.

[47] J. Xia *et al.*, "Adjusted weight voting algorithm for random forests in handling missing values", *Pattern Recognition*, Vol. 69, pp. 52–60, 2017.

[48] P. Jönsson and C. Wohlin, "An evaluation of k-nearest neighbour imputation using lIkert data", In: *Proc. of International Software Metrics Symposium*, pp. 108–118, 2004.

[49] M. j Azur, E. A. Stuart, C. Frangakis, and P. J. Leaf, "Multiple imputation by chained equations: what is it and how does it work?", *International journal of methods in psychiatric research*, Vol. 20, No. 1, pp. 40–49, 2011.

[50] N. Abiri, B. Linse, P. Edén, and M. Ohlsson, "Establishing strong imputation performance of a denoising autoencoder in a wide range of missing data problems", *Neurocomputing*, Vol. 365, pp. 137–146, 2019.

[51] M. F. Bosu and S. G. Macdonell, "Experience: Quality benchmarking of datasets used in software effort estimation", *Journal of Data and Information Quality*, Vol. 11, No. 4, 2019.

[52] Fuqiang Gu, Kourosh Khoshelham, Shahrokh Valaee, Jianga Shang, and Rui Zhang, "Locomotion Activity Recognition Using Stacked Denoising Autoencoders", *IEEE Internet of Things Journal*, Vol. 5, No. 3, pp. 2085–2093, 2018.

[53] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors", *arXiv preprint*, pp. 1–18, 2012. [Online]. Available: http://arxiv.org/abs/1207.0580.

[54] L. Breiman, "Random forests", *Journal of Machine Learning*, Vol. 45, pp. 1–122, 2001.

[55] Q. Song and M. Shepperd, "A new imputation method for small software project data sets", *Journal of Systems and Software*, Vol. 80, No. 1, pp. 51–62, 2007.