Received: February 10, 2022. Revised: March 16, 2022.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Conditional Variational Autoencoder with Inverse Normalization Transformation on Synthetic Data Augmentation in Software Effort Estimation

Robert Marco[1]* **Sharifah Sakinah Syed Ahmad[2]** **Sabrina 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: The fundamental challenge in developing a prediction model for software effort estimation (SEE) is related to the availability of inadequate public data or the small amount of information contained in the data. As a result, because the information contained in the data is small, it can be challenging to make accurate estimates with insufficient SEE data. Data augmentation techniques, and synthetic data have been used successfully to overcome problems with limited data sets. In this study, conditional variational autoencoder (CVAE) and an inverse normalization transformation (INT) were investigated as methods for data augmentation and synthetic data generation. Eleven data sets in the field of software engineering (such as the PROMISE repository and ISBSG) that will be used as case studies. Overall, the CVAE-INT model was used for the augmented data set (synthetic data). Based on the Mann-Whitney test, it shows that our method provides a significant level with a mean p-value greater than 0.90 in the data set. Our model yields lower mean absolute error (MAE) and root mean squared error (RMSE) values. Increased mean error for the datasets Albercht (0.0269; 0.0842), China (0.0444; 0.1079), Cocomo81 (0.0544; 0.1155), Desharnais (0.0702; 0.1258), IFPUG (0.0225; 0.0448), ISBSG10 (0.0996; 0.1836), Kemerer (0.0776; 0.1457), Kitchenham (0.0001; 0.0010), Maxwell (0.0746; 0.1230), Nasa93 (0.0980; 0.1488), and UCP (0.0427; 0.0899). Furthermore, we assessed the influence of synthetic data using five machine learning prediction approaches (such as CART, k-nearest neighbors (kNN), Support Vector Regression (SVR), MultiLayer Perceptron (MLP), and Random Forest (RF)). The results show that our method can provide data augmentation techniques for the transformation of variance data that can produce synthetic data that has the same distribution as real data. In addition, this study shows that the synthetic data produced has a significant effect on increasing the accuracy of the five predictor methods in the machine method.

Keywords: Data augmentation, Autoencoder, Inverse normalization transformation, Software effort estimation.

1. Introduction

The critical challenge in developing the SEE model is the lack of data and the high expenses involved with data collecting [1, 2]. This is because software project collection is costly and can consume significant time and workload [3]. As a result, obtaining accurate predictions with minimal SEE data may be problematic since the information included in such small data may not be sufficient to enable the training of the SEE model [3, 4].

The data augmentation (DATA) technique is a technique that aims to complement a data set with similar data created from the information in the dataset [5]. The use of supervised methods produces a model that meets the requirements for completing a particular mission with a labeled dataset [6]; a function's output might be a constant value (regression) or it can anticipate the input object's class label (classification) [7]. One of the most challenging types of data expansion in this technique is combining different samples with the same label in the feature space to create a new sample with the same label [5, 8].

Only two studies in the SEE context have tackled

small datasets to generate synthetic data using data augmentation to the best of our knowledge. For example, Kamei et al. (2008), augmentation method extends synthetic minority over-sampling technique (SMOTE) to use analogy-based conventional over-sampling from classification to regression by relating class imbalances to the most predictive features of SEE [9]. Furthermore, Song et al. (2018), proposed a synthetic data generator by slightly shifting several randomly selected training examples. Where γ (syn.rate) and $\lceil . \rceil$ used for rounding up operator, τ (categorical) overcoming categorical data and σ^2 (Gaussian) for the probability of each categorical. It can be used as a data preprocessor with any SEE technique [1].

Previous research has shown that data augmentation will serve as a regulator to help prevent overfitting and improve performance with imbalanced distribution of data [10, 11]. This is mostly done in the technique of adding data to enlarge the training set and improve training performance [12, 13]. One of the traditional method that is often used as data augmentation in dealing with data limitations and small data, such as: synthetic minority over-sampling technique (SMOTE) [9], Gaussian Copula [14], multiple imputation chained equations (MICE) [15, 16]; while the most popular deep learning-based generative models, such as generative adversarial networks (GAN) [18, 19], and variational autoencoders (VAE) [17].

SMOTE oversampling is likely the most often utilized approach [20, 21]. In the work of Torgo et al. (2015), several resampling techniques were successfully applied for regression such as SMOTER [22]. This method is very susceptible to the problem of generating noise samples [23]; they still have several flaws, such as overgeneralization and a lot of variation [24]. In contrast to MICE, for data sets containing different variables, this method is not suitable for drawing from shared distributions when the conditional models are not compatible [25]. Gaussian Copula creates synthetic data and judges its usefulness to be too esoteric, expensive, and convoluted [26]. Meanwhile, GAN has its strength in capturing complex data distributions [27, 28]. However, GANs require multiple models to train, which makes identifying optimal model parameters difficult and time-consuming [29].

VAE is significantly easier to train by utilizing a gradient-based approach [30, 31]. VAE is also effective at capturing the original sample's uncommon distribution characteristics and producing synthetic samples that are identical to the original [32]; and has better convergence property [24]. nonetheless, the variational autoencoder technique imposes significant distributional assumptions, which may be deleterious to the generative model [29]. Unfortunately, while VAE can generate samples, it cannot generate specific samples based on labels [33]. Kingma et al. (2014) developed conditional variational auto-encoder (CVAE) which is an extension of VAE to overcome these shortcomings [34]. CVAE is a new deep generative network that reconstructs input features using output vectors [35]; and shrink the network's dimensions, but also create new attack samples in the chosen category [33].

Even though CVAE is extensively used for data synthesis, all of these methods assume that there is enough original data to train the CVAE model adequately. CVAE effectively extracts prospective data features using a robust learning encoder and reconstructs the decoder to offer enough data for deep neural networks. If this condition is not met, the trained CVAE model will not adequately replicate the original data's distribution properties, rendering the synthetic data created by such a model incorrect. The CVAE structure, in particular, must be redesigned to make synthetic data, as the current structure is only capable of two-dimensional processing data [35]. In addition, the existing tabular data generative model has difficulty in dealing with complex and diverse types of marginal distributions because the gradient problem disappears, and this model pays less attention to correlations between attributes. After we embed attribute vectors and labels encoded in ordinal encoding to convert categorical data. Thus, we propose an inverse normalization transformation (INT) method to accurately model correlations between attributes by introducing additional monitoring tasks to aid correlation extraction. Then, for each continuous column, we train the neural network to conduct an inverse transformation of the generated data into the target distribution, resulting in synthetic data that closely resembles the real data in the SEE context.

The rest of this document is laid out as follows: In section 2, we'll go over some related research, and in section 3, we will present our approach using the variational autoencoder and our proposed method. We'll talk about experimental design in section 4 of this paper. Section 5 presents a performance evaluation, comparing our proposed technique with others and the impact of our synthetic data on machine learning in a SEE context. Section 6 concludes with a discussion of the conclusions and future work.

2. Related studies

In this paper, we will review some of the prior work on the variational autoencoder approach, which has been used extensively for synthetic data creation or representational learning in general. A variety of studies have previously been published that use VAE-based solutions as a data augmentation strategy.

For example, Xu et al. (2021) developed a deep learning-based intrusion detection technique called the log-cosh conditional variational autoencoder (LCVAE) to produce new intrusion data with intriguing variations. Unfortunately, this technique has trouble identifying characteristics that are very similar to those found in standard logs, which might lead to classifier misjudgments [36]. Gong et al. (2020) proposed CVAE as a data augmentation strategy for detecting electricity theft, taking into account the shape of the sample and the nature of the distribution at the same time. However, there are difficulties in modeling the CVAE because the training process requires multiple power curves labeled electricity theft which is challenging to obtain in some cases, making the CVAE model impossible to train [35]. Fans et al. (2022) offer a CVAE-based time series data augmentation approach that produces high-quality synthetic data samples with an average RMSE performance increase ratio of 12 to 18 %. However, in order to increase the model's generalization performance, this strategy requires a larger latent dimension to produce synthetic data [37].

Adaptive increase dimension of variational autoencoder (ADA-INCVAE) by Huang and Wang (2021) shows that limiting the generation range based on local information can help classifiers improve precision (1.17 %) and recall (3.95 %) of minority samples resulting in synthetic samples. Unfortunately, this method takes a little longer to run during the data generation stage [38]. Abdel-Aty and Islam (2021), to encode all events into latent space, use a variational autoencoder. After training, the model successfully differentiated crash and non-crash events, resulting in synthetic data that followed the real data pattern with enhanced specificity of 8 % and 4 %, respectively. However, because this method is trained with a small amount of non-crash data, it has lower specificity than other techniques and so cannot correctly distinguish some of the non-crash events in the test data [39].

Based on the literature review results, from the perspective of tabular data augmentation, our work will propose a conditional variation autoencoder (CVAE) with inverse normalization transformation (INT). To our knowledge, this is the first study of CVAE-INT and regression problems in the SEE field in situations of addressing small data availability.

Table 1. Description dataset

Notations	Description
x	Independent variable
y	Target variable/label
μ	Mean
σ	Variances
z	Latent variable
$\odot$	Element-wise product
$q_\phi(z\|x^n)$	Encoder network
$p_\theta(x^n\|z)$	Decoder network
ϕ , θ	Network parameter
$p_\theta(z)$	Latent variable's prior distribution
Σ	Diagonal matrix constraint
$g(.)$	Nonlinear decoder network transformation
β	Vector parameter
$p_d(x\|y)$	Multi-modal
$p_\psi(z\|y)$	Conditioned on y and is represented by a neural network with parameter ψ
$\{\epsilon_n\}$	Independent normally distributed error with constant variance
$\mathcal{D}, \mathcal{D}^*, \mathcal{D}'$	Dataset, New data, Synthetic data
F	Functions
s^2	Squares of the errors
$\mathbb{R}$	Real number

3. Our approach

This section provides a description and mathematical modeling for the conditional variational autoencoder. We first explain to establish a theoretical background for modeling the problem. All the notations and abbreviations used in this paper are summarized in Table 1.

3.1 Variational autoencoder

VAE (variational autoencoder) is a generative model capable of modeling data's latent representation [40], [41]. Fig. 1 shows how the VAE encoder translates real data into a set of means and variances (denoted as μ and σ^2), establishing the normal distribution features [42]. The latent vector z is then obtained from the latent normal distribution using random sampling, where $z = \mu + \sigma \odot \epsilon$, ϵ is an arbitrary value taken from the standard normal distribution, and $\odot$ represents the element-wise product [37]. Gradient backpropagation through the entire VAE model is possible with a random sampling process while ensuring stochasticity [30]. Regularization loss is a technique for reducing the risk of overfitting while also creating a well-structured latent space. It's worth noting that the latent vector's size must be tuned to guarantee that the

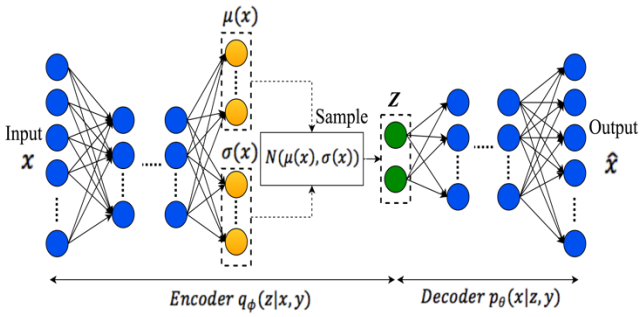

Figure. 1 Architecture variantional autoencoder

synthetic data is of high quality [37].

In the generative model in Eq. (1), define the data distribution based on $p_\theta(x)$ on the observed data and $\{x^n\}_{n=1}^N$ for the log-likelihood of the data [24]:

$$log \prod_{n=1}^{N} p_\theta(x^n) = \sum_{n=1}^{N\Sigma_\theta^n} log \quad (1)$$

Thus, the log probability may be calculated by combining the kullback-leibler (KL) divergence with the variational evidence lower bound (ELBO) [40].

$$log p_\theta(x^n) = D_{KL}\left(q_\phi(z|x^n)\right) + L(\theta, \phi; x^n) \quad (2)$$

Where z is the latent variable and $q_\phi(z|x^n)$ is a close approximation of the true posterior that $p_\theta(z|x^n)$ cannot solve. Since there is no negative KL divergence between $q_\phi(z|x^n)$ and $p_\theta(z|x^n)$.

$$log p_\theta(x^n) \geq L(\theta, \phi; x^n) \quad (3)$$

$log p_\theta(x^n)$ maximization is identical to ELBO $L(\theta, \phi; x^n)$, maximization, which may be divided into two terms:

$$L(\theta, \phi; x^n) = E_{q_\phi(Z|x^n)}[log p_\theta(x^n|z)] - D_{KL}(q_\phi(z|x^n)||p_\theta(z)) \quad (4)$$

Assume that $q_\phi(z|x^n)$ and $p_\theta(x^n|z)$ are the encoder and decoder networks, respectively, while ϕ and θ are the network parameters. For example, $p_\theta(z)$ represents the latent variable's prior distribution. In VAE, the latent Gaussian prior distribution is used $p(z) = N(z; \mathbf{0}, \boldsymbol{I})$. As a multivariate gaussian distribution, the approximate posterior distribution is multivariate gaussian.

$$log\ p_\phi(z|x^n) = logN(z; \mu(x^n), \Sigma(x^n)) \quad (5)$$

Where μ and Σ are deterministic encoder functions with variation parameters and a diagonal matrix constraint. Because $p(z)$ and $q_\phi(z|\ x^n)$, have two Gaussian distributions, it may be computed in ELBO as:

$$D_{KL}(q_\phi(z)||p_\theta(z)) = D_{KL}[N(\mu(x^n), \Sigma(x^n))||N(\mathbf{0}, \boldsymbol{I})] = \frac{1}{2}\sum_{l=1}^{L}((\mu_l^n)^2 + (\sigma_l^n)^2 - 1 - \log((\sigma_l^n)^2) \quad (6)$$

The dimension of z is L . As a result, maximization of ELBO in Eq. (4) equates to minimization of the loss function, as follows:

$$L^*(\theta, \phi; x^n) = \|x^n - g(z^n)\|^2 + \frac{1}{2}\sum_{l=1}^{L}((\mu_l^n)^2 + (\sigma_l^n)^2 - 1 - \log((\sigma_l^n)^2) \quad (7)$$

$g(.)$ is a nonlinear decoder network transformation. To balance the reconstruction error and the KL divergence, an adjustable hyperparameter β is added to the second term of the loss function [43].

$$L^*(\theta, \phi; x^n) = \|x^n - g(z^n)\|^2 + \frac{\beta}{2}\sum_{l=1}^{L}((\mu_l^n)^2 + (\sigma_l^n)^2 - 1 - \log((\sigma_l^n)^2) \quad (8)$$

A higher value for the hyperparameter β, leads to a more structured latent space with lower reconstruction costs, while a lower value leads to a better reconstruction with a less structured latent space and lower reconstruction costs.

3.2 Conditional variational autoencoder

Conditional variational autoencoder (CVAE) [34] extends the variational autoencoder (VAE) [40] basic model, which comprises an encoder and a decoder, to describe the distribution of observed data in an unsupervised way using latent variables. The conditional distribution $p_d(x|y)$ [44], is approximated using the conditional variational autoencoder (CVAE), as shown in Fig. 2. When the

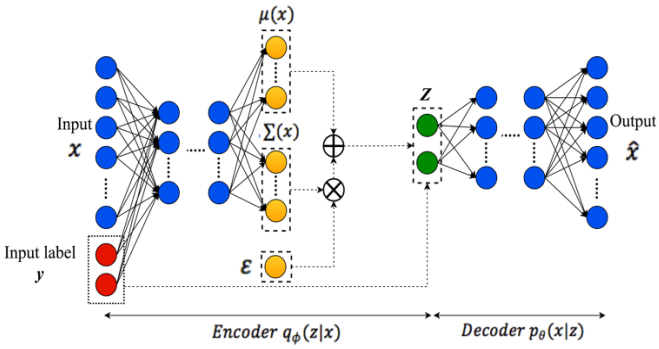

Figure. 2 Architecture conditional variantional autoencoder

distribution of $p_d(x|y)$ is multi-modal, it can outperform the deterministic model (xs varies in probability for a given y). If x is real, deterministic regression model with a mean squared error loss will predict x average value. CVAE investigates the x distribution, which may be used to sample a wide range of realistic objects [45].

By conditionalizing the distributions investigated at y [45], the lower limit of variation for VAE may be obtained similarly to VAE:

$$\begin{aligned} L_{CVAE}(x,y;\theta,\psi,\phi) &= E_{q_{\phi(z|x,y)}} \log p_\theta(x|z,y) \\ &\quad - D_{KL}(q_\phi(z|x,y)||p_\psi(z|y)) \\ &\leq \log p_{\theta,\psi}(x|y) \end{aligned} \quad (9)$$

The repameterization method is used to optimize CVAE aims. The preceding distribution, $p_\psi(z|y)$ is conditioned on y and is represented by a neural network with parameter ψ. CVAE employs three trainable neural networks, while VAE only employs two [45].

3.3 Inverse normalization transformation

Assume that the dependent variable y and the independent variable x, and use $\mathcal{D} = \{x_n, y_n\}$ to model and estimate this relationship. The regression model's generic form for representing y in the form of x is:

$$y_n = F(x_n, \beta) + \epsilon_n \quad (10)$$

Where $\{\epsilon_n\}$ is an independent normally distributed error with constant variance, and β is a vector parameter p. The values of the parameters $\beta_0, \beta_1, \dots, \beta_p$ are commonly calculated to reduce the number of squares of mistakes [46]:

$$s^2 = \min_\beta \sum_{n=1}^{N} [y_i - F(x_n, \beta)]^2 \quad (11)$$

The least squares approach's parameters are adjusted using either linear or nonlinear regression techniques, depending on the nature of the function *F*. A different way to model the box-cox transformation is to transform the dependent variable so that the error variance constant and normality of the error distribution are obtained when the model is linear for *F* [47].

$$y^{(\lambda)} = \begin{cases} \frac{y^\lambda - 1}{\lambda} \ (\lambda \neq 0) \\ \log(y) \ (\lambda = 0) \end{cases} \quad (12)$$

If this transformation works, $y^{(\lambda)}$ should be properly represented by a linear function of the standard normal variable $x, F(x;\beta) = \beta_0 + \beta_1 x$. The "maximum likelihood" technique is the λ value for that best satisfies this criteria. Eq. (12) can be used to derive the box-cox transformation, which can then be used to develop the inverse normalization transformation (INT) with various error distributions, yielding equation [46]:

$$y = \begin{cases} [\lambda(\beta_0' + \beta_1' x) + 1]^{\frac{1}{\lambda}} = M(1 + \beta_0 x)^{\beta_1}, \lambda \neq 0 \\ \exp(\beta_0' + \beta_1' x) = M \exp(\beta_2 x), \quad \lambda = 0 \end{cases} \quad (13)$$

In each example, the middle term is rebuilt on the right to keep the Y, M median. Eq. (13) to give a generally poor representation of the quantile relationship between y (the quantile of the nonnormal variable *Y*) and the corresponding standard normal quantile, *X*.

Eq. (14) is obtained by generalizing the log term by presenting it as a box-cox transformation [46].

$$y = M \exp\{\beta_1[(1 + \beta_0 x)^{\frac{\beta_3}{\beta_0}} - 1]/(\frac{\beta_3}{\beta_0}) + \beta_2 x\}, x > -1/\beta_0 \quad (14)$$

INT has four parameters, one of which is the median, which is used as a scale parameter and is kept by INT (when $x = 0, y = M$). The suitable installation process will decide the remaining four characteristics.

4. Experiment design

In this section, we introduce an experimental design using a deep learning approach as a data augmentation method that aims to generate synthetic data on a regression problem:

4.1 Problem statement

Building an SEE model usually requires data, but collecting data requires a lot of time, workload, and high costs associated with data collection. This results in the sample training data being available being a small data set, leading to unsatisfactory performance of the SEE model. Instead of collecting as much data on the completed software project as possible (takes a considerable amount of time) to build a sophisticated SEE model.

So, we will generate a data synthesis project from the completed data based on data $\mathcal{D}$ by collecting small data in the SEE context, as: $\mathcal{D}^* = \mathcal{D} \cup \mathcal{D}'$. Where $\mathcal{D}'$ is the resulting synthesis data project, then

the training process is based on $\mathcal{D}^*$ as the new data. Given a randomly selected training example $\mathcal{D} = \{(X_n, y_n)\}_{n=1}^{N}$, $X \in \mathbb{R}^d$ where $X_n = (x_1, x_2, \dots, x_i) \in \mathbb{R}^d$ synthetic project creation with conditions all distributions considered on y, for $y \in \mathbb{R}^1$ as:

$$q_\phi : (\mathbb{R}^d, \mathbb{R}) \to p_\theta : (\mathbb{R}^d, \mathbb{R})$$
$$\left(\begin{bmatrix} q_\phi(z|x_1) \\ \dots \\ q_\phi(z|x_d) \end{bmatrix}, q_\phi(z|y)\right) = p_\theta(x^{syn}, y^{syn}) \quad (15)$$

Where q_ϕ is the encoder that is used for the feed as input data and p_θ is the decoder that generates the synthesis data from z.

4.2 Development of the data augmentation approach via DVAE-INT

However, the main limitation of classical variational autoencoder inference is the need for conjugation of probability and priors in order for most problems to be carefully optimized, which in turn may limit the application of the algorithm [48]. VAE uses neural networks to show conditional posteriors [40], allowing the variational inference goal to be properly optimized via stochastic gradient descent and conventional backpropagation.

VAE has a higher convergence rate than GAN, one of its main advantages. Even though VAE is often used for data synthesis, these approaches presume enough real-world data to train the VAE model effectively. If this assumption is incorrect, the trained VAE model will not capture the real data's distribution features, resulting in erroneous synthetic data [24]. Kingma et al. (2014) developed Conditional variational auto-encoder (CVAE) which is an extension of VAE to overcome these shortcomings [34]. CVAE successfully extracts prospective features using an encoder with high learning capabilities. The decoder reconstructs the input features, providing adequate data for the deep neural network [35].

However, based on the findings of previous research studies, CVAE use in generating synthetic data to overcome the availability of small data in regression problems is still limited. We use CVAE which is effective in extracting potential features in the data, and reconstructing it to produce synthetic data. We also redesigned the CVAE structure to make it suitable for processing one-dimensional data. Also, we perform categorical conversion of data using ordinal encoding embedded in attribute vectors and labels. Furthermore, to strengthen the correlation

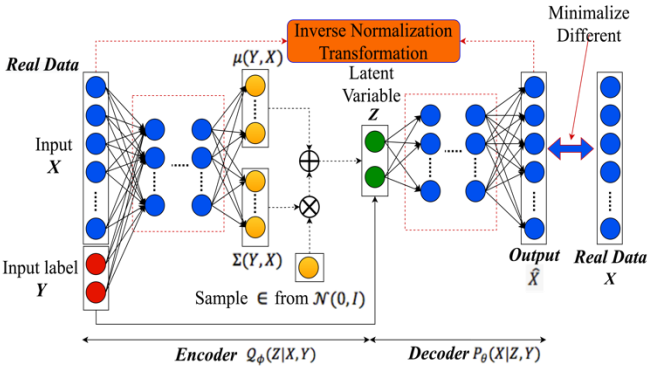

Figure. 3 Our proposed improved CVAE-INT

between the attributes in the regression, we used the inverse normalization transformation (INT) method to aid correlation extraction. Then, we train the neural network to perform an inverse transformation into the target distribution to obtain synthetic data that looks like real data for each continuous column.

However, some of the other popular augmentation methods that will be used for comparison in the method we are developing have some drawbacks. Torgo et al (2015) proposed SMOTER with a re-sampling method to generate synthetic data for a regression task, where the target variable is continuous [22]. Unfortunately, this method is very prone to problems generating noise samples and resulting in overgeneralization/high variance. Goodfellow et al. (2014), the GAN trains two models at the same time: the generative model *G*, which captures the data distribution, and the discriminative model *D*, which estimates the probability of the sample being drawn from the training data. The disadvantages are mainly that there is no explicit representation of $p_g(x)$, and that *D* must synchronize nicely with *G* during training, making optimal model parameters difficult and time consuming [49]. Yoon et al. (2019) developed a new time series GAN (TGAN) architecture for generating realistic time-series data, integrating the unsupervised paradigm's flexibility with the control afforded by supervised training. This method generates realistic samples using a variety of real-time and synthetic datasets. However, the time GAN algorithm has system-wide compatibility, overall training time, and is not very sensitive to hyperparameters [50]. Xu et al. (2019) proposed conditional tabular generative adversarial networks (CTGAN) to generate realistic synthetic data from tabular data having a mix of discrete and continuous columns. To solve the problem the continuous column may have several modes while the discrete column is sometimes unbalanced making modeling difficult [51].

4.3 Dataset collection and preprocessing

We selected eleven publicly accessible real-life datasets, nine from the PROMISE Repository and two from the ISBSG [52], [53], [54], which is one of the most prominent datasets, such as Maxwell, Cocomo81, Kitchenham, Nasa93, Kemerer, Albrecht, Desharnais, China, and UCP are the datasets accessible in Promise Repository. Meanwhile, ISBSG has two datasets available: ISBSG18-IFPUG and ISBSG10.

These data sets come in different sizes and dimensions to conduct comprehensive experiments. After data collection, the data preprocessing stage will be carried out to improve data quality. The data preprocessing step usually includes feature reduction, i.e., removing irrelevant features. The next step is to convert categorical data into numeric using ordinal encoding. Note that the ISBSG dataset has a sample with the value "?" as missing values; we address the problem of missing data using the Denoising autoencoder approach for missing imputation of the dataset. Table 2 summarizes all variables in the real data, including mean, standard deviation (std), skewness (Skew), and kurtosis (Kurt).

4.4 Performance analysis

We present a model training and validation technique on the suggested model in Fig. 4 to determine whether the resultant data performs as intended. The real data is then divided into three groups: 70 % training, 15 % testing, and 15 % validation. After then, only training data is used, while test data is saved for the ultimate evaluation. The next step is to use CVAE-INT to analyze and create synthetic data. Finally, carry out the data augmentation evaluation procedure.

The performance of our model is evaluated in this study using error metrics. The model is superior if the mean absolute error (MAE) and root mean squared error (RMSE) values are low and the R-squared (R^2) value is high.

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|X_i - Y_i| \tag{16}$$

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(X_i - Y_i)^2} \tag{17}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(X_i - Y_i)^2}{\sum_{i=1}^{m}(\bar{Y} - Y_i)^2} \tag{18}$$

Statistical testing uses non-parametric which aims to validate the model further. Usually, the mann-whitney U test is used if the data is continuous. The procedure for calculating the mann-whitney *p*-value is described below:

$$U = N_1N_2 + \frac{N_2(N_2+1)}{2} - \sum_{i=N_1+1}^{N_2} R_i \tag{19}$$

Where, N_1 as the first sample size (real data), and N_2 as the second sample size (synthetic data). While,

Table 2. Description dataset

Dataset	Mean	Std	Skew	Kurt
Albercht	178.847	354.367	0.104	2.564
China	331.375	1850.14	2.458	18.95
Cocomo81	45.633	468.473	3.491	22.91
Desharnais	116.506	162.231	0.110	4.345
IFPUG	240.639	1224.507	0.103	3.385
ISBSG10	178.847	354.367	-0.854	3.624
Kemerer	366.230	561.459	0.408	3.069
Kitchenham	1675.802	6052.215	0.357	8.560
Maxwell	335.977	2539.394	0.729	8.147
Nasa93	41.195	295.891	3.919	30.46
UCP	1167.333	2435.962	0.387	2.841

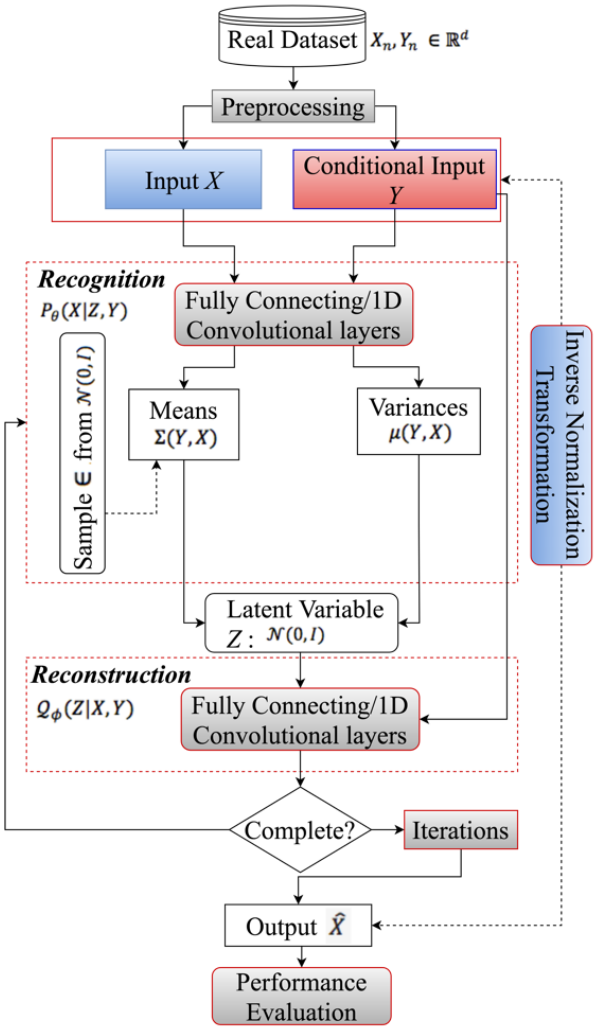

Figure. 4 Prosedur of the training scheme

R_i as a sample size rating. The maximum possible value of R_i can be $N_1N_2 + \frac{N_2(N_2+1)}{2}$. The first null and alternate hypotheses are formed when testing statistical significance between two data samples using the mann-whitney *p*-value.

- Null hypothesis (H0) states that the distributions of the two samples are identical.
- Alternative hypothesis (H1): The distributions of the two samples are different.

The significance level between real data and synthetic data in all investigated data sets is greater than 0.05 (p-value>0.05), implying that they are statistically quite close or have the same distribution. Therefore, the alternative hypothesis must be rejected.

5. Result and discussion

We shall assess the success of the strategy provided in this study. The keras platform, tensorflow, and the scikit-learn python package are used to execute our suggested technique and other basic models. Setting the hyperparameters is a crucial step in training a deep learning model. The trial and error technique was used to identify the best parameter settings for all deep learning models. Our simulation was run on an NVIDIA GeForce GPU Titan 100 with an i7-6800k processor, 64 GB of RAM, TensorFlow 1.2.1, and Ubuntu 14.04 LTS. The default Python 3.0 parameter settings are utilized.

5.1 Hyperparameter analysis

In our work, we are evaluating the performance of the suggested model through experiments. In PROMISE and ISBSG, we employed 11 distinct small datasets from the Repository set. We will compare the proposed approach to various standard models for creating synthetic data in our study. To identify the parameters in our experiment, the sequence of data available was randomly partitioned into a training data set of 70 %, a testing data set of 15 %, and a validation data set of 15 %. The training data set was the only data set used for data augmentation. The resulting synthetic data is represented by developing synthetic data from actual data and then combining the synthetic and real data ($\mathcal{D}^* = \mathcal{D} \cup \mathcal{D}'$).

CVAE-INT is a work-in-progress that uses fully linked layers to generate synthetic data. The model was built with a 1D convolution layer and optimized with a grid search mode to capture the temporal relationships in the regression data [37]. The number of layers and hidden filters are tuned in model optimization to decrease computational costs since they significantly influence model performance. For all involved data sets, the hyperparameters defined in our trained model employ the embedding dimension of the hidden layer of 256, which seeks to improve the DNN generalization performance [37], [33]. Adam default was learning speed and the rectified linear unit (ReLU) optimizer we utilized in this experiment in TensorFlow. For our model, Adam for a network with a learning rate of 1×10^{-3} [33], [55]. The ReLU is utilized as the activation function for all levels except the encoder and decoder output layers, which use the linear activation function. We'll choose a latent size of 2.5. The ideal value of ReLU, on the other hand, may vary depending on the model [37], [56]. To overcome the suggested model's overfitting problem, we incorporated a 0.2 dropout strategy by lowering the number of neurons. If gain too much weight, will become less fit.

We separated the model into three subsets and ran it five times to ensure that each data subset had an equal chance of being utilized in the test section. The accuracy score was then obtained by taking the average model accuracy in the test subset. Finally, the best parameter is chosen as the one with the highest cross-validation score.

5.2 Regression performance

We consider advanced augmentation techniques in dealing with regression problems to validate the effectiveness of the proposed model. In this situation, we will use error measurements and statistical approaches to assess the efficacy of our model, as mentioned in section 4.4.

CVAE-INT was explicitly trained on 11 different samples. Each CVAE-INT model was trained over 100 epochs, with 15 % of the dataset being utilized for validation. Fig. 5 shows the model losses for the positive and negative data sets in the training and validation sets, respectively. Our model mainly converges after 20-40 epochs.

The test is then carried out by comparing the distribution of real data with the synthetic sample produced by our advanced method. Our technique has the benefit of addressing the problem of overfitting because broad generalizations are difficult to obtain when working with small samples. Our suggested approach has a more similar distribution to real data and has more variety, and it can overcome the regression problem while maintaining the same structure. To validate the effectiveness of our strategy, we gave the average results of five runs for each method.

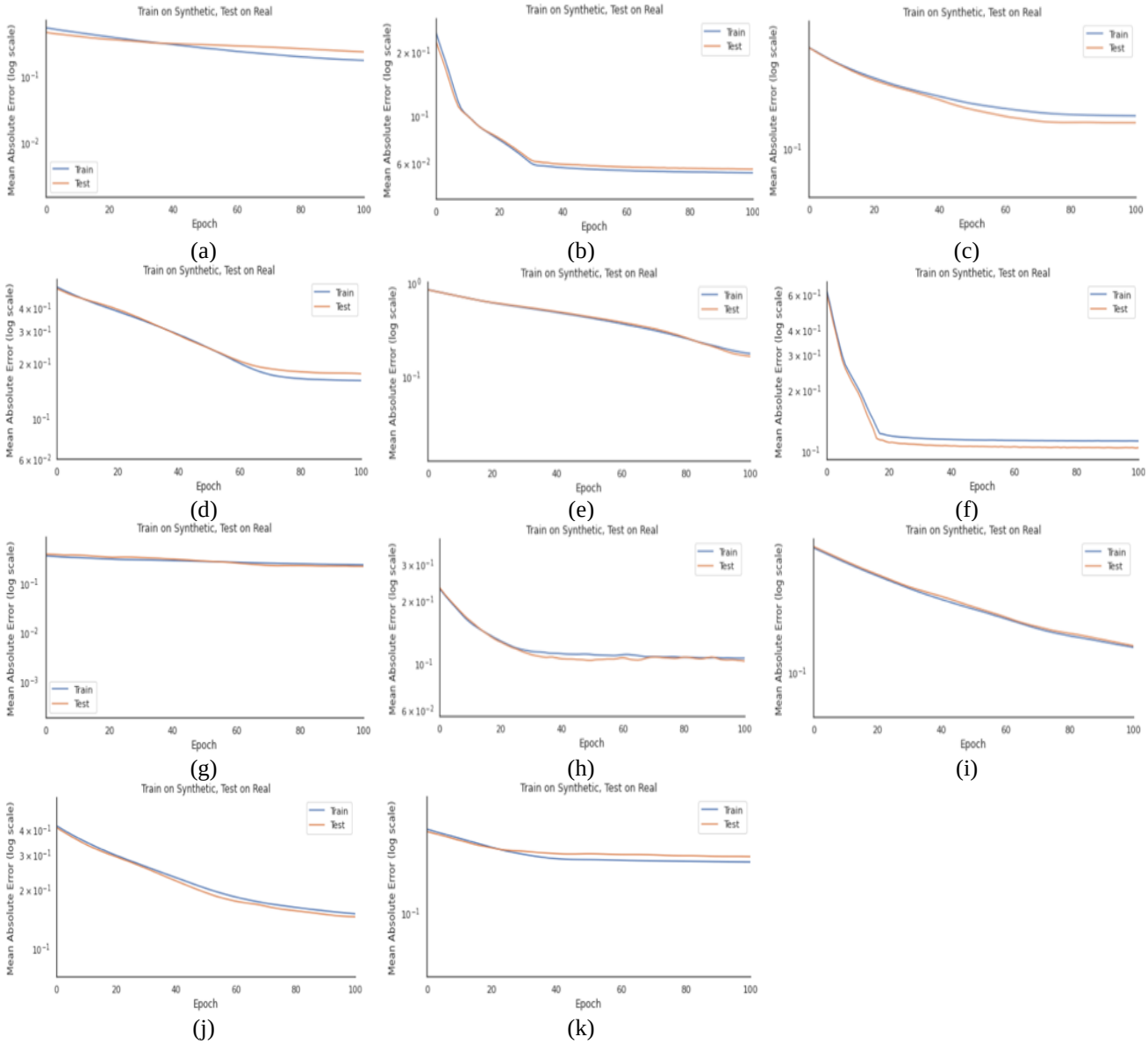

Figure. 5 Training and validation loss of the our model (a) Albercht (b) China (c) Cocomo81 (d) Desharnais (e) IFPUG (f) ISBSG10 (g) Kemerer (h) Kitchenham (i) Maxwell (j) Nasa93 (k) UCP

Table 3. Evaluation performance our model

Dataset	MAE	RMSE	R^2
Albercht	**0.0269**	**0.0842**	**0.9262**
China	0.0444	0.1079	*0.0900*
Cocomo81	0.0544	0.1155	**0.6072**
Desharnais	0.0702	0.1258	**0.6738**
IFPUG	**0.0225**	**0.0448**	**0.7566**
ISBSG10	0.0996	0.1836	*-0.0066*
Kemerer	0.0776	0.1457	*0.4057*
Kitchenham	**0.0001**	**0.0010**	**0.9982**
Maxwell	0.0746	0.1230	*0.4542*
Nasa93	0.0980	0.1488	*0.2047*
UCP	**0.0427**	**0.0899**	**0.8929**

MAE, RMSE, and R^2 were used to apply a set of assessment criteria listed in Table 3. In terms of MAE and RMSE, our model yielded the lowest error, however neither one had the highest error. In our model, the best MAE and RMSE values are bolded. In addition, R^2 in our model delivers the greatest value, even though there is a lowest and negative value (italics) that shows that the dataset has no regression association.

We also use t-SNE to display synthetic data which is used to compare the distribution of real and synthetic data. The findings suggest that our approach can assist in bridging the gap between real and synthetic data. Fig. 6 depicts an analysis of the data distribution of each of the 11 datasets utilized in this study.

To compare the distribution of real data with the resulting data (synthetic data), we used scatter plots.

p-value = 0.9466; mean = 166.26; t-test = -0.0672; std = 337.508
(a)

p-value = 0.9755; mean = 346.55; t-test = 0.0306; std = 1563.85
(b)

p-value = 0.9047; mean = 55.397; t-test = -0.1190; std = 526.238
(c)

p-value = 0.7956; mean = 101.50; t-test = 0.2590; std = 136.039
(d)

p-value = 0.9691; mean = 246.35; t-test = -0.0388; std = 1042.851
(e)

p-value = 0.8549; mean = 372.91; t-test = -0.1828; std = 1788.975
(f)

p-value = 0.9558; mean = 1509.7; t-test = -0.0553; std = 563.746
(g)

p-value = 0.9656; mean = 364.47; t-test = 0.0434; std = 6230.028
(h)

p-value = 0.9516; mean = 338.28; t-test = -0.0607; std = 2519.062
(i)

p-value = 0.9892; mean = 43.025; t-test = 0.0134; std = 296.752
(j)

p-value = 0.9825; mean = 1158.4; t-test = 0.0219; std = 2409.071
(k)

Figure. 6 Evaluation data distributions using t-SNE visualization of bridged by CVAE-INT: (a) Albercht dataset, (b) China dataset, (c) Cocomo81 dataset, (d) Desharnais dataset, (e) ISBSG IFPUG dataset, (f) ISBSG10 dataset, (g) Kemerer dataset, (h) Kitchenham dataset, (i) Maxwell dataset, (j) Nasa93 dataset, and (k) UCP dataset

This allows us to compare distributions and offers us an estimate of the mean of the two data sets. In Fig. 6, real data is shown in blue, while synthetic data is shown in orange. The resulting data pattern is very similar to the real data distribution for all variables. The value of the standard deviation of the distribution for the synthetic dataset, which has a value smaller than the real data, indicates that the data generated from the augmentation process contains fewer outliers. Based on the mann-whitney test shows that our method provides a significant level greater than 0.05 with a mean p-value greater than 0.90 in the data set, as can be observed. This indicates that the mean of all datasets is similar. Thus, we can conclude that the two data sets are statistically comparable.

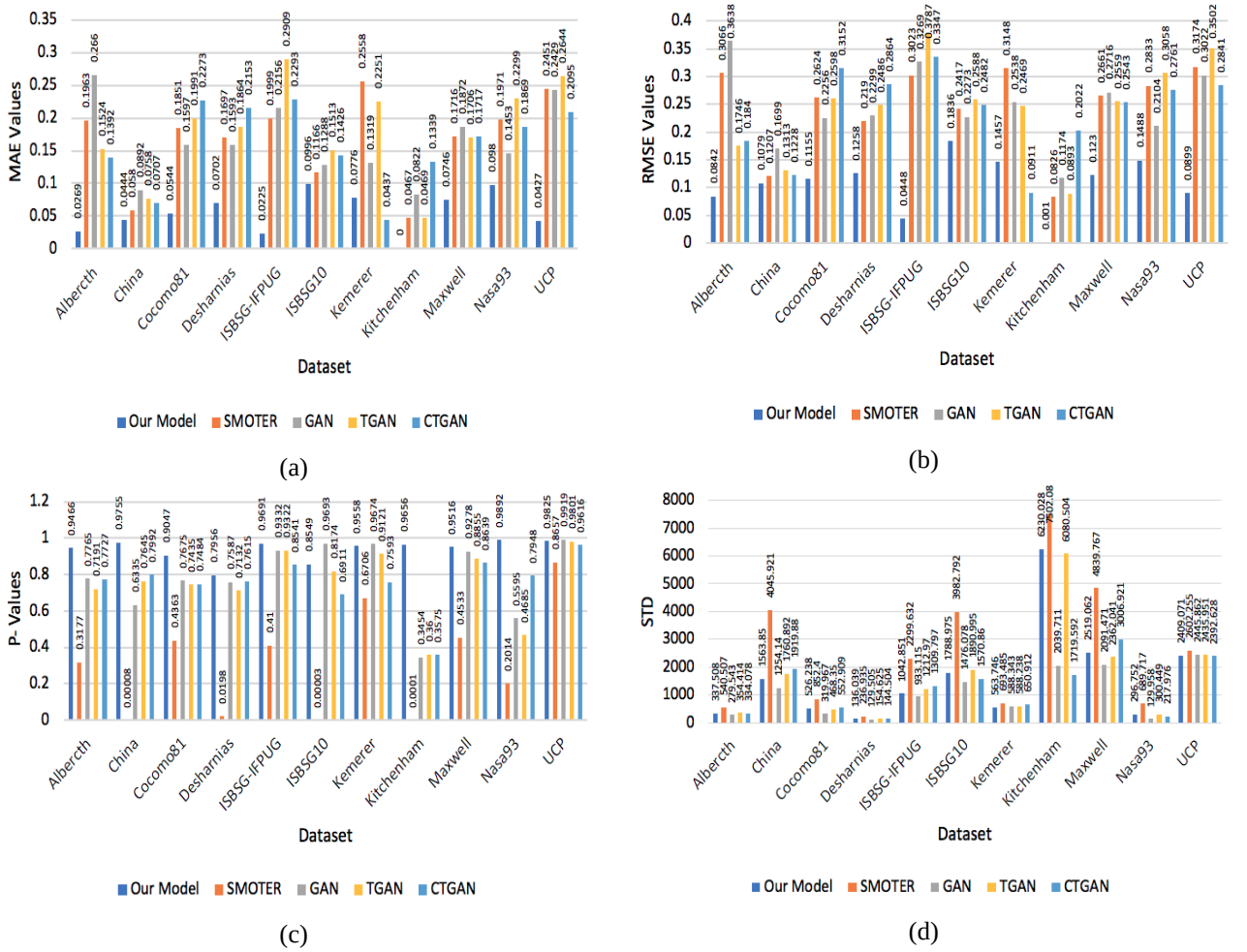

Figure. 7 Comparison performance evaluation of the algorithm (a) MAE, (b) RMSE, (c) P-Values, and (d) std

5.3 Comparison with state-of-the-art algorithms

In this part, we compare our proposed technique to popular data augmentation methods, including SMOTE Regression (SMOTER) [22], which was modified from SEE, generative adversarial networks (GAN) [49], TimeGAN (TGAN) [50], and conditional tabular generative adversarial networks (CTGAN) [51].

Fig. 7 shows a performance comparison of our proposed technique with other standard data augmentation methods based on error metrics and statistical tests.

The comparison of our technique with various prominent methods is based on the MAE and RMSE values, as shown in Fig. 7. The results show that our method produces the best accuracy value. Where, our method has the lowest value on the evaluation of the error metrics with MAE and RMSE values respectively, in the data set albercht (0.0269; 0.0842), China (0.0444; 0.1079), cocomo81 (0.0544; 0.1155), desaharnias (0.0702; 0.1258), ISBSG18-IFPUG (0.0225; 0.0448), ISBSG10 (0.0996; 0.1836), kemerer (0.0776; 0.1457), kitchenham (0.0001; 0.0010), maxwell (0.0746; 0.1230), nasa93 (0.0980; 0.1488), and UCP (0.0427; 0.0899). while the poor values were obtained by the SMOTER method on the kemerer data (0.2558; 0.3148), the GAN method on the albercht data (0.2660; 0.3638), china (0.0892; 0.1699), and maxwell (0.1872; 0.2716). Furthermore, by the TGAN method on ISBSG18-IFPUG data (0.2909; 0.3787), ISBSG10 (0.1513; 0.2588), nasa93 (0.2299; 0.3058), and UCP (0.2644; 0.3502), the last by the CTGAN method on cocomo81 data (0.2273; 0.3152), desharnais (0.2153; 0.2864), and kichenham (0.1339; 0.2022).

Based on the mann-whitney test, it shows that our method provides a significant level greater than 0.05 with a mean p-value greater than 0.90 in the data set, as can be observed. The findings reveal that our technique beats all other popular methods in most datasets (equal distribution of real and synthetic data), except for ISBSG10, Kemerer, and UCP, which outperform the conventional GAN method. However,

Table 4. Comparison of the impact of machine learning performance on synthetic and real datasets using MAE values

Dataset	CART		KNN		MLP		SVR		RF	
	Real	Syn.	Real	Syn.	Real	Syn.	Real	Syn.	Real	Syn.
Albercht	0.0534	0.0012	0.1016	0.0011	0.0888	0.0035	*0.1694*	0.0536	0.1217	**0.0009**
China	0.0178	0.0020	0.0244	0.0019	0.0284	0.0087	*0.0567*	0.0278	0.0115	**0.0016**
Cocomo81	0.0934	0.0027	0.0466	0.0019	*0.0949*	0.0115	0.1230	0.0409	0.0681	**0.0018**
Desharnais	0.0228	0.0020	0.0594	0.0018	*0.0678*	0.0066	0.0522	0.0401	0.0161	**0.0014**
IFPUG	*0.2557*	0.0010	0.1591	0.0008	0.1853	0.0084	0.1923	0.0173	0.1698	**0.0007**
ISBSG10	0.0174	0.0038	0.0268	0.0030	0.0228	0.0094	*0.0568*	0.0379	0.0132	**0.0027**
Kemerer	0.1167	0.0105	0.2732	**0.0084**	0.2442	0.0415	*0.3319*	0.0382	0.2326	0.0088
Kitchenham	0.0065	**0.0004**	0.0068	**0.0004**	0.0074	0.0008	*0.0823*	0.0811	0.0044	**0.0004**
Maxwell	0.0667	0.0018	0.0718	**0.0012**	0.0638	0.0060	*0.0982*	0.0679	0.0544	**0.0012**
Nasa93	0.0433	0.0018	0.0492	0.0014	0.0451	0.0093	*0.0945*	0.0486	0.0213	**0.0011**
UCP	0.1146	0.0058	*0.1367*	**0.0037**	0.2307	0.0390	0.0960	0.0504	0.1012	0.0045

Table 5. Comparison of the impact of machine learning performance on synthetic and real datasets using RMSE values

Dataset	CART		KNN		MLP		SVR		RF	
	Real	Syn.	Real	Syn.	Real	Syn.	Real	Syn.	Real	Syn.
Albercht	0.0776	0.0032	0.1538	0.0031	0.1238	0.0054	*0.2379*	0.0561	0.1702	**0.0025**
China	0.0479	0.0180	0.0479	0.0208	0.0567	0.0176	*0.0730*	0.0421	0.0400	**0.0146**
Cocomo81	*0.1911*	0.0068	0.0745	0.0041	0.1139	0.0189	0.1528	0.0479	0.1396	**0.0018**
Desharnais	0.0321	0.0050	0.0656	0.0061	*0.0861*	0.0092	0.0654	0.0462	0.0248	**0.0041**
IFPUG	*0.3762*	0.0051	0.2290	**0.0036**	0.2379	0.0125	0.2382	0.0284	0.2271	0.0039
ISBSG10	0.0387	0.0143	0.0455	0.0118	0.0309	0.0166	*0.0624*	0.0421	0.0280	**0.0113**
Kemerer	0.1964	0.0280	0.2934	0.0233	0.2680	0.0561	*0.3673*	0.0511	0.2674	**0.0219**
Kitchenham	0.0129	**0.0034**	0.0156	0.0054	0.0150	0.0023	*0.0831*	0.0829	0.0098	0.0045
Maxwell	0.1049	0.0055	0.0982	**0.0041**	0.0855	0.0090	*0.1120*	0.0699	0.0959	**0.0041**
Nasa93	0.1102	0.0083	*0.1223*	0.0071	0.0717	0.0148	0.1179	0.0531	0.0457	**0.0043**
UCP	0.2655	0.0197	0.1801	**0.0122**	*0.2932*	0.0541	0.1206	0.0593	0.1961	0.0146

in the Kitchenham dataset, the GAN, TGAN, and CTGAN methods have the lowest values. Meanwhile, the SMOTER method has a low value in almost all datasets.

Our technique has a lower distribution standard deviation number than actual data for a bigger dataset because it has a smaller fraction of outliers, as previously described. Most of the other approaches,have a greater standard deviation than the actual data, implying that they contain a higher percentage of outliers.

In general, our strategy outperformed the other methods utilized in this study. This is because VAE is the outcome of integrating the Bayesian variation approach with neural networks' flexibility and scalability [40], [57]. Using variational inference, it is feasible to transform an intractable inference into an optimization problem.

5.4 Sensitivity to data augmentation of machine learning

We investigated the impact of producing new synthetic data using the data augmentation strategy described in this study. Synthetic data generated from 11 publicly available real-life datasets was used to test how much impact they have on machine learning performance in the regression field. Synthetic data generated in our paper is synthetic data that produces the same amount of real data, which is then combined with real data.

We analyze the impact of using synthetic data on machine learning performance by comparing it to real data. The use of five classical supervised learning algorithms that are popular in the SEE context, such as classification and regression tree (CART), k-nearest neighbors (kNN), support vector regression (SVR), multilayer perceptron (MLP), and random forest (RF). Tables 4 and 5 show the performance results on machine learning based on the MAE and RMSE error metrics, with the best values declared in bold, on the contrary, the poor values in italics. Overall, the synthetic data created by our method improves the accuracy of the machine learning baseline prediction method significantly. The random forest algorithm has the best performance than other algorithms based on the Table 4 and 5. Nevertheless, other algorithms have performance that is almost the same or close to the random forest algorithm.

6. Conclusion

This paper investigates the conditional variation autoencoder method combined with the inverse normalization transformation (CVAE-INT) to generate synthetic data. Our study helps overcome the limitations of the small, generally available data in the SEE field. In this regard, our method is compared with four other popular methods, such as SMOTER, GAN, TGAN, and CTGAN. Our method was used to add eleven data sets to the PROMISE repository and the ISBSG. Overall, our method is the best-performing model. The mean of our method gives an increase in a p-value for the t-test greater than 0.90 for all datasets; MAE and RMSE are also lower. CVAE-INT effectively extracts potential features in 1-dimensional data capable of handling complex and diverse types of marginal distributions due to disappearance gradient problems to help extract correlations between attributes. Our method is very stable during the training process and has a fast convergence speed. The resulting synthetic data curve has a shape and distribution characteristic similar to the original data. Finally, we'll put our synthetic data results to the test using machine learning regression algorithms (such as CART, KNN, MLP, SVR, and RF). In the context of SEE, this research provides empirical proof that our synthetic data significantly impacts machine learning performance. Experiments on synthetic and real-world data sets reveal that our method outperforms state-of-the-art tabular data models.

It is hoped that in the future, researchers can develop CVAE in producing synthetic data to overcome the problem of small data in the field of classification and production methods that are resistant to outliers. It is hoped that further research can use several datasets from other fields.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

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

References

[1] L. Song, L. L. Minku, and X. Yao, "A novel automated approach for software effort estimation based on data augmentation", In: *Proc. of International Conf. on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 468–479 , 2018.

[2] S. Amasaki, T. Yokogawa, and H. Aman, "Applying cross project defect prediction approaches to cross-company effort estimation", In: *Proc. of International Conf. on ACM International Conference Proceeding Series*, pp. 76–79, 2019.

[3] B. Turhan, Association for Computing Machinery, and ACM Digital Library, "Building a Second Opinion: Learning Cross-Company Data", In: *Proc. of International Conf. on Predictive Models in Software Engineering*, pp. 1–10, 2013.

[4] M. Jørgensen and M. Shepperd, "A Systematic Review of Software Development Cost Estimation Studies", *IEEE Transactions on Software Engineering*, Vol. 33, No. 1, pp. 33–53, 2007.

[5] J. Lemley, S. Bazrafkan, and P. Corcoran, "Smart Augmentation Learning an Optimal Data Augmentation Strategy", *IEEE Access*, Vol. 5, pp. 5858–5869, 2017.

[6] Y. C. Chou et al, "Deep-learning-based defective bean inspection with GAN-structured automated labeled data augmentation in coffee industry", *Applied Sciences (Switzerland)*, Vol. 9, No. 19, 2019.

[7] B. Twala and M. Cartwright, "Ensemble missing data techniques for software effort prediction", *Intelligent Data Analysis*, Vol. 14, No. 3, pp. 299–331, 2010.

[8] S. Bazrafkan, T. Nedelcu, P. Filipczuk, and P. Corcoran, "Deep learning for facial expression recognition: A step closer to a smartphone that knows your moods", In: *Proc. of International Conf. on Consumer Electronics*, ICCE, pp. 217–220, 2017.

[9] Y. Kamei, J. Keung, A. Monden, and K. Matsumoto, "An Over-sampling Method for Analogy-based Software Effort Estimation", In: *Proc. of International Conf. on IEEE Empirical Software Engineering and Measurement*, pp. 312–314, 2008.

[10] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W.

P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique", *Journal of Artificial Intelligence Research*, Vol. 16, No. 2, pp. 321–357, 2002.

[11] X. Zhu, Y. Liu, Z. Qin, and J. Li, "Emotion classification with data augmentation using generative adversarial networks", *Advances in Knowledge Discovery and Data Mining*, Vol. 5, pp. 349–360, 2018.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *Advances in Neural Information Processing Systems*, pp. 1–14, 2012.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks", *Springer International Publishing*, Vol. 9908, pp. 632–645, 2016.

[14] N. Patki, R. Wedge, and K. Veeramachaneni, "The synthetic data vault", In: *Proc. of International Conf. on IEEE Data Science and Advanced Analytics*, pp. 399–410, 2016.

[15] T. Raghunathan, J. Lepkowski, J. Van Hoewyk, and P. Solenberger, "A multivariate technique for multiply imputing missing values using a sequence of regression models", *Journal of Survey Methodology*, Vol. 27, No. 1, pp. 85–96, 2001.

[16] J. Chen, J. Li, W. Chen, Y. Wang, and T. Jiang, "Anomaly detection for wind turbines based on the reconstruction of condition parameters using stacked denoising autoencoders", *Renewable Energy*, Vol. 147, pp. 1469–1480, 2020.

[17] Z. Wan, Y. Zhang, and H. He, "Variational autoencoder based synthetic data generation for imbalanced learning", In: *Proc. of International Conf. on IEEE Computational Intelligence*, pp. 1–7, 2018.

[18] P. S. Kumar and D. R. Venkatesan, "Improving Software Defect Prediction using Generative Adversarial Networks", *International journal of Science and Engineering Applications*, Vol. 9, No. 9, pp. 117–120, 2020.

[19] L. A. D. Souza et al, "Assisting Barrett's esophagus identification using endoscopic data augmentation based on Generative Adversarial Networks", *Computers in Biology and Medicine*, Vol. 126, No. June, 2020.

[20] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: Special Issue on Learning from Imbalanced Data Sets", *ACM SIGKDD Explorations Newsletter*, Vol. 6, No. 1, pp. 1–6, 2004.

[21] A. Fernández, S. D. Río, N. V. Chawla, and F. Herrera, "An insight into imbalanced Big Data classification: outcomes and challenges", *Complex & Intelligent Systems*, Vol. 3, No. 2, pp. 105–120, 2017.

[22] L. Torgo, P. Branco, R. P. Ribeiro, and B. Pfahringer, "Resampling strategies for regression", *Expert Systems*, Vol. 32, No. 3, pp. 465–476, 2015.

[23] J. Sun, J. Lang, H. Fujita, and H. Li, "Imbalanced enterprise credit evaluation with DTE-SBD: Decision tree ensemble based on SMOTE and bagging with differentiated sampling rates", *Information Sciences*, Vol. 425, pp. 76–91, 2018.

[24] Y. Lyu, J. Chen, Z. Song, and Q. Zhang, "Synthesizing data by transferring information in data-intensive regions to enhance process monitoring performance in data-scarce region", *Canadian Journal of Chemical Engineering*, Vol. 99, No. S1, pp. S521–S539, 2021.

[25] 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.

[26] A. Chen, S. Chen, and D. Zhao, "Creation of Fully-Synthetic, Multivariate Continuous Data Using Multivariate Adaptive Regression Splines with Multiple-Imputation", *Researchgate.net*, No. March, 2020.

[27] M. Simão, P. Neto, and O. Gibaru, "Improving novelty detection with generative adversarial networks on hand gesture data", *Neurocomputing*, Vol. 358, pp. 437–445, 2019.

[28] P. Xu, R. Du, and Z. Zhang, "Predicting pipeline leakage in petrochemical system through GAN and LSTM", *Knowledge-Based Systems*, Vol. 175, pp. 50–61, 2019.

[29] N. Tagasovska, D. Ackerer, and T. Vatter, "Copulas as high-dimensional generative models: Vine copula autoencoders", *Advances in Neural Information Processing Systems*, Vol. 32, pp. 1–13, 2019.

[30] I. Goodfellow, N. Papernot, and P. McDaniel, "cleverhans v0.1: an adversarial machine learning library", *arXiv preprint*, Vol. 2 , pp. 2–6, 2016.

[31] M. Bregere and R. J. Bessa, "Simulating Tariff Impact in Electrical Energy Consumption Profiles with Conditional Variational Autoencoders", *IEEE Access*, Vol. 8, pp. 131949–131966, 2020.

[32] S. Shao, P. Wang, and R. Yan, "Generative adversarial networks for data augmentation in machine fault diagnosis", *Computers in Industry*, Vol. 106, pp. 85–93, 2019.

[33] 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.

[34] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised Learning with Deep Generative Models", *Advances in Neural Information Processing Systems*, pp. 1–9, 2014.

[35] X. Gong, B. Tang, R. Zhu, W. Liao, and L. Song, "Data augmentation for electricity theft detection using conditional variational auto-encoder", *Energies*, Vol. 13, No. 17, pp. 1–14, 2020.

[36] X. Xu, J. Li, Y. Yang, S. Member, and F. Shen, "Log-Cosh Conditional Variational Autoencoder", *IEEE Internet of Things Journal*, Vol. 8, No. 8, pp. 6187–6196, 2021.

[37] 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.

[38] K. Huang and X. Wang, "ADA-INCVAE: Improved data generation using variational autoencoder for imbalanced classification", *Applied Intelligence*, Vol. 52, pp. 2838–2853, 2022.

[39] Z. Islam, M. A. Aty, Q. Cai, and J. Yuan, "Crash data augmentation using variational autoencoder", *Accident Analysis and Prevention*, Vol. 151, No. December 2020, pp. 105950, 2021.

[40] D. P. Kingma and M. Welling, "Auto-encoding variational bayes", arXiv preprint, No. 10, pp. 1–14, 2014.

[41] Q. Zhao, N. Honnorat, E. Adeli, A. Pfefferbaum, E. V. Sullivan, and K. M. Pohl, "Variational Autoencoder with Truncated Mixture of Gaussians for Functional Connectivity Analysis", *Springer International Publishing*, Vol. 11492, 2019.

[42] T. T. Um, F. M. J. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kuli´c, "Data augmentation of wearable sensor data for Parkinson's disease monitoring using convolutional neural networks", In: *Proc. of International Conf. on ACM Multimodal Interaction*, Vol. 2017-January, pp. 216–220, 2017.

[43] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space", In: *Proc. of International Conf. on Computational Natural Language Learning*, pp. 10–21, 2016.

[44] K. Sohn, X. Yan, and H. Lee, "Learning structured output representation using deep conditional generative models", *Advances in Neural Information Processing Systems*, Vol. 2015-January, pp. 3483–3491, 2015.

[45] O. Ivanov, M. Figurnov, and D. Vetrov, "Variational autoencoder with arbitrary conditioning", In: *Proc. of International Conf. on Learning Representations*, pp. 1–25, 2019.

[46] H. Shore, N. Brauner, and M. Shacham, "Modeling physical and thermodynamic properties via inverse normalizing transformations", *Industrial and Engineering Chemistry Research*, Vol. 41, No. 3, pp. 651–656, 2002.

[47] G. E. P. Box and D. R. Cox, "An Analysis of Transformations", *Journal of the Royal Statistical Society: Series B (Methodological)*, Vol. 26, No. 2, pp. 211–243, 1964.

[48] N. Dilokthanakul et al, "Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders", *Arxiv print*, No. 2016, pp. 1–12, 2016.

[49] I. J. Goodfellow et al, "Generative Adversarial Nets", *Advances in Neural Information Processing Systems*, pp. 3063–3071, 2014.

[50] J. Yoon, D. Jarrett, and M. V. D. Schaar, "Time-series generative adversarial networks", *Advances in Neural Information Processing Systems*, Vol. 32, pp. 1–11 , 2019.

[51] L. Xu, M. Skoularidou, A. C. Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN", *Advances in Neural Information Processing Systems*, Vol. 32, 2019.

[52] A. K. Bardsiri, S. M. Hashemi, and M. Razzazi, "Statistical Analysis of the Most Popular Software Service Effort Estimation Datasets", *Journal of Telecommunication, Electronic and Computer Engineering*, Vol. 7, No. 1, pp. 87–96, 2015.

[53] B. Vasilescu, A. Serebrenik, and T. Mens, "A historical dataset of software engineering conferences", In: *Proc. of International Conf. on IEEE Mining Software Repositories*, pp. 373–376, 2013.

[54] T. R. Benala, R. Mall, P. Srikavya, and V. HariPriya, "Software Effort Estimation Using Data Mining Techniques", *Advances in Intelligent Systems and Computing*, Vol. 248, pp. 85–86, 2014.

[55] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization", In: *Proc. of International Conf. on Learning Representations*, pp. 1–15, 2015.

[56] A. Krizhevsky and G. Hinton, "Convolutional

deep belief networks on cifar-10", *Unpublished Manuscript*, pp. 1–9, 2010.

[57] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models", In: *Proc. of International Conf. on Machine Learning ICML*, Vol. 4, pp. 3057–3070, 2014.