Original Research Article
Enhancing DAE Training Efficiency for Missing Data Imputation via Input Data Recombination

ABSTRACT

The growing significance of addressing data loss, has led to an increased focus on missing data imputation (MDI). Autoencoder (AE) models, known for their ability to autonomously learn and impute missing data, are gaining prominence in MDI. These models exhibit adaptability to diverse datasets, and their unsupervised nature makes them robust in handling data lacking clear labels.

AE training encompasses critical elements such as optimization algorithms, loss functions, and training epochs. Despite substantial research in this field, there is a lack of exploration regarding the impact of updating input data during AE training. Traditionally, AEs are trained on the original data, assuming it contains latent information. However, in the context of MDI, where data may be corrupted, it becomes imperative to evaluate whether updating input data can lead to superior results. Drawing inspiration from Gradient Boosting Machines, we introduce two methods: STR-IU (Short-Term Reconstruction with Iterative Updates) and LTR-SU (Long-Term Reconstruction with a single Update).

We utilize Denoising Autoencoder (DAE) models and examine how various optimization mechanisms affect our proposed methods. We conduct comparisons between Stochastic Gradient Descent (SGD) and the Adam optimization algorithm, And we transformed three distinct datasets into synthetic datasets with varying levels of missing data (5%, 15%, 25%). The results indicate that, while performance may not consistently excel across all training epoch settings, there is a noticeable overall improvement when updating input data, whether using SGD or Adam. Additionally, LTR-SU outperforms STR-RU, and models with DAE using SGD exhibit greater optimization compared to DAE using Adam

Keywords: Missing data imputation; Denoising Autoencoder; updating input data

1. INTRODUCTION

There are numerous reasons that can lead to missing data, such as equipment malfunction, human error, limitations in data collection (such as patients or clients dropping out or missing appointments), and transmission loss (such as an unstable network signal). Missing data can have a detrimental effect on data quality and can cause issues during data analysis [1-2] Without proper handling of missing data, it can increase computational costs, skew outcomes, and misguide researchers [3].

Missing data can be problematic, especially when there is a large amount of missing data. This can make it challenging to identify statistical effects because the statistical power of the data is diminished. Additionally, missing data can introduce bias to the parameter estimates being studied because the available observational data may not accurately represent the population. Decreased precision of an estimate means that it will be less accurate [4-5]. As such, missing data should be handled with care. If missing data is not properly imputed, it can lead to increased uncertainty in the analysis and incomplete conclusions.

Based on the relationship between the missing values and other observed variables in the dataset, missing data can be categorized into three mechanisms of missingness [6-8].

1.1. CATEGORIZATION OF MISSINGNESS MECHANISMS

1.1.1Missing Completely At Random (MCAR)

MCAR means that the missing values in the dataset are completely random and unrelated to the values of other observed variables. In other words, the probability of a value being missing is not related to the values of other variables in the dataset.

1.1.2 Missing AT Random (MAR)

By contrast, MAR means that the missingness is related to the observed variables in the dataset. In other words, the probability of a value being missing is dependent on the values of other observed variables, but not on the other missing values.

1.1.3 Missing Not At Random (MNAR)

MNAR means that the missingness is related to the missing value itself or with other unobserved data. In other words, the probability of a value being missing depends on the values of other unobserved variables or the missing value itself.

1.2. MISSING DATA IMPUTATION IN MCAR, MAR AND MNAR

The approaches to handling missing data may depend on the missing mechanisms and the proportions of missing data [9]. It is commonly recommended that when dealing with MCAR or MAR mechanisms, if the proportion of missing data is less than 5%, it can be suggested to omit or use traditional techniques (such as deletion or single imputation) to handle it [10-12]. This is because the influence of bias is likely to be minimal. However, if the proportion of missing data is over 5%, more advanced techniques (such as multiple imputation, model-based procedures, and machine learning methods) should be used to fill in the missing data [9].

On the other hand, due to the missingness mechanism of MNAR being dependent on the missing value itself or other unobserved data, researchers may use model-based methods or shared parameter models, which may require strong assumptions to more explicitly resolve the missingness mechanism. In order to account for missingness due to selection bias, sensitivity analysis and correction factors may be used to evaluate the robustness of the results to different assumptions [13].

In the realm of missing data imputation research, there has been a plethora of studies focusing on the application of AEs. However, a notable gap exists in the literature concerning the enhancement of input data quality. This gap is rationalized by the conventional assumption in normal circumstances that the chosen training dataset possesses the most comprehensive and complete information. In the domain of missing data imputation, the situation deviates from this norm, as the available datasets are inherently flawed and incomplete.

In such a context, the datasets accessible for training are defective and possess omissions. Considering this scenario, the simulated data generated after training may potentially offer a closer approximation to the true characteristics of the dataset. Integrating the simulated data, derived from the training process, with the original data containing missing values for subsequent training iterations could potentially yield more effective training outcomes.

1.3. APPLICATION OF AUTOENCODERS FOR THE IMPUTATION OF MISSING DATA

In the past couple of years, deep learning-based methods have been successfully applied to many research problems, including the missing data imputation community. Among all deep learning methods, the Autoencoder (AE) is particularly noteworthy. Firstly, AE is an ANN (Artificial Neural Networks)-based model, which can be trained in an unsupervised way and is designed to learn from incomplete data and produce new probable values for imputation. With the benefits of ANNs, AEs can model much more complex data patterns compared to other simpler methods (such as the KNN method based on simple distances or the MICE method, which relies internally on simple regressions).

Secondly, since AEs can learn directly from incomplete data, they natively support multivariate scenarios. This means that AE can perform the imputation of all features with missing values while accounting for the relations between them, which is more difficult for most imputation methods and vanilla ANNs. A comprehensive study[14] surveyed the use of AEs for the imputation of tabular data and considered 26 works published between 2014 and 2020. The analysis results showed

that both the variants of AE (Denoising AE and Variational AE) outperformed their competitors in the vast majority of works that reported imputation results.

The use of AEs for missing data imputation is a promising area of research. Various types of AEs have been investigated for this task, including Denoising AE [15-18], Variational AE [19-21], and other variants of AEs [22-26]. However, it is important to note that using AEs for missing data imputation is still in its early stages, and there is ongoing research to explore their effectiveness and limitations.

One potential limitation of AEs is that they directly learn from input data, making them more susceptible to noise and outliers. Additionally, the fitting capacity of AEs is limited, which can lead to overfitting and a weakening of the model's generalization ability. As a result, it is crucial to carefully consider the use of AEs for missing data imputation and explore ways to mitigate these limitations. Researchers have proposed various modifications to AEs, such as adding regularization terms to the loss function, using different activation functions or network architectures, and incorporating external information or constraints into the model.

Overall, while AEs show promise for missing data imputation, it is important to approach their use with caution and continue exploring their potential and limitations.

We propose an innovative approach involving the integration of simulated data, generated through the training of an AE for imputation, with the incomplete original dataset. This strategic integration aims to enhance training performance by supplementing deficient information within the input data using the imputed values derived from the AE training process. The amalgamation of simulated and original data is anticipated to provide a more comprehensive and accurate representation during subsequent training iterations, contributing to improved imputation outcomes.

2. RELATED WORK

2.1 APPROACHES TO MISSING DATA IMPUTATION

These approaches for handling missing data can be approximately divided into two types. One is statistical-based methods (such as single imputation, multiple imputation, and model-based procedures), which aim to restore the missing values with the most similar ones among the observed data. The other is machine learning-based techniques which replace the missing values by constructing a predictive model with the available data to estimate values.[14]

2.1.1 Statistical Methods for Missing Data Imputation

The most common traditional statistical-based methods used to handle missing data, which are deletion and single imputation

2.1.1.1 Deletion Methods

Deletion techniques are intuitive and easiest to execute, the ideas is solving the problem by to delete the cases that contain missing data. According to the difference of deletion case analysis, deletion techniques have two techniques which are complete-case analysis and available case analysis. As the name suggests, complete-case analysis involves the excision of all the data from an analysis that contains missing values. It is obvious when there are a large number of missing values and if the original data set is too small, this technique would introduce serious bias. Further, available case analysis is a more selective method, which decides the quantity of missing data on a case by case basis. If the cases with high levels of missing data would be deleted. By doing so to minimize data loss. Complete-case analysis and available case analysis both assumes the MCAR mechanism.

2.1.1.2 Single imputation Methods

Single imputation is a procedure that involves analyzing the data together via other variables in order to find the most likely value that can substitute in the data. There are many methods to find the imputation value, such as by mean, median, mode of the available values of that variable.

• Mean/Median Imputation:

This imputation technique involves replacing the missing value with the arithmetic mean/median of all the other cases, this methods only suit for small data samples because the results would be twisted owing to uncertainty in the sample distribution.

• Regression Imputation:

Regression imputation is often to use a function to estimate the relationship between a data point and its related variable. The commonly known function is least-squares fit form. Besides it also have other forms for instance multiple linear, quadratic, cubic as well as non-polynomial. The problem of regression imputation is the chosen regression model may be perfectly fit along, however it absence of an error term which could fails to account for data variability

• Hot Deck and Cold Deck:

Hot-Deck technique involves finding a similar matched dataset. The common approaches of finding the closely dataset are the distance function approach which is imputes the missing value by the smallest squared distance of the case with the missing value, and the pattern matching approach which ideas is to group similar cases together, and, the imputed value for the missing case is randomly selected from values in similar group. In contrast to Hot-Deck the Cold-Deck is mostly alike the hot-deck imputation, but the data source is based on other external source of information, which are not derived from the current dataset.

2.1.1.3Multiple imputation Methods

To address the limitations associated with single imputation, which may inadequately capture the variability in datasets and potentially introduce bias, researchers propose the use of multiple imputation techniques. The multiple imputation procedure involves the analysis of incomplete datasets through the application of various simulation models. This approach introduces variability to imputed data, generating a spectrum of plausible values for each missing data point.

In the initial phase, missing values in incomplete datasets are imputed over m iterations. During each iteration, one or more sets of imputations are generated using single imputation techniques, resulting in a comprehensive set of imputations for all missing values of the target variable. Simultaneously, standard errors are computed for each iteration. Upon completion of the m iterations, the estimated values and standard errors are integrated to produce a final result, typically represented as an averaged set of values.

While multiple imputation mitigates the issue of bias associated with single imputation by incorporating dataset variability and providing a range of plausible responses, its implementation is intricate. The process is time-consuming due to the need to run multiple iterations for estimation purposes. Additionally, careful design is required for combining results and ensuring the correct utilization of estimated values. These complexities underscore the importance of methodological rigor in the application of multiple imputation techniques.

2.1.2 Machine learning-based methods for Missing Data Imputation

Machine learning methods are modern procedures that involves the creation and construction of algorithms that make predictions on data that is missing which is based on information available in the dataset. Some the most common machine learning techniques are mentioned below

2.1.2.1 K-Nearest Neighbors (KNN) algorithm

KNN algorithm is widely employed for imputing missing values by replacing them with observed values from the nearest neighbors. For each observation with missing values, the algorithm identifies the nearest neighbors based on a predefined distance function. This distance function quantifies the similarity or dissimilarity between observations, and common metrics include Euclidean distance, Manhattan distance, or other suitable measures depending on the characteristics of the data. The imputation is then performed by incorporating observed values from the nearest neighbors, leveraging the local data structure to make informed and contextually relevant replacements.

2.1.2.2Decision Trees (DT)

DT are widely used tools in supervised learning. They are represented as tree structures and can be employed to map data and observations. The primary objective of decision trees is to draw conclusions about the target values, which are represented in the tree's leaf nodes. One of the major advantages lies in the ability to visualize data, allowing for easy comprehension of the data structure. Typically, the goal is to find the optimal decision tree by minimizing the standard error. This makes decision trees an interpretable model that provides a clear illustration of data mapping and prediction processes

2.1.2.3Neural Networks (NN)

A NN is comprised of layers of artificial neurons, collectively processing and transforming input data into meaningful output. Each connection between neurons is assigned a weight, and the network refines its performance by adjusting these weights during the training process.

Through iterative adjustments during training, the neural network minimizes the disparity between predicted and actual outcomes. This adaptive learning capability enables neural networks to generalize well to unseen data.

NN are particularly effective in capturing non-linear relationships within data, making them powerful tools for tasks like pattern recognition, classification, and regression. Their proficiency in handling complex and high-dimensional datasets allows them to learn and represent intricate patterns and dependencies.

2.2 AUTOENCODER (AE)

An AE is a type of artificial neural network composed of at least three layers: an input layer, a hidden layer, and an output layer. A typical AE consists of two components: an encoder and a decoder. In the encoder part, the input vector X is mapped to a latent feature Z. This process involves nonlinear encoding functions. In the decoder part, the latent feature Z is mapped back to reconstruct the original input X. Again, this process involves nonlinear decoding functions. In summary, an AE can be described as a neural network that utilizes nonlinear encoding and decoding functions to map an input to a latent representation and then reconstruct the input from that latent representation.

In the encoder part, the mapping process is a nonlinear transformation. The encoding function is represented as f(x) = σ(WX_e + b_e), where σ represents a nonlinearity such as the activation function, W represents the weight matrix, and b represents the bias vector. Similarly, in the decoder part, the latent space Z is mapped back to reconstruct the input as the output Y. The decoding function is represented as g(z) = σ(W_dZ + b_d), where σ is again a nonlinearity, W' represents the transpose of the weight matrix used in the encoder, and b' represents the bias vector. These functions f(x) and g(z) capture the nonlinear relationships between the input and the latent representation, as well as between the latent representation and the reconstructed output. The structure of the Villa AE is illustrated in Figure 1.

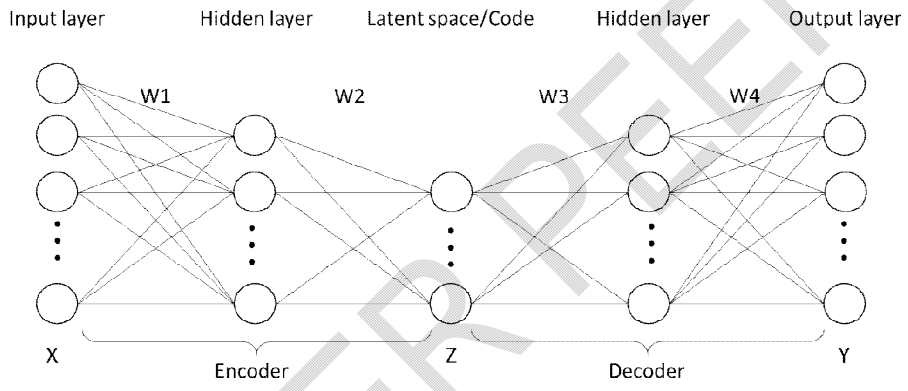


Figure 1. Structure of the vanilla AE.

The goal of an AE is to minimize the reconstruction error between the input X and the output Y. The training process of an AE involves optimizing the model parameters {W_e, W_d, b_e, b_d}) to achieve this objective. To optimize these parameters, a Stochastic Gradient Descent (SGD) variant is commonly used. SGD is an iterative optimization algorithm that updates the parameters based on the gradients of the loss function with respect to the parameters. The specific variant of SGD used can vary, and popular choices include Adam, RMSprop, or plain vanilla SGD. During training, the AE calculates the reconstruction error, which is the discrepancy between the input X and the output Y. This error is then used to compute a loss function, which quantifies the overall reconstruction performance. The choice of loss function depends on the nature of the data and the specific task at hand. Mean squared error (MSE) is a common choice for continuous data, while binary cross-entropy or categorical cross-entropy may be used for binary or categorical data, respectively. The AE iteratively adjusts the model parameters using the SGD variant, aiming to minimize the reconstruction error and improve the overall performance of the AE. The optimization process continues until the model converges or reaches a predefined stopping criterion.

2.2.1 Denoising Autoencoder (DAE) Application in Missing Data Imputation

The DAE[15] is a variant of AE specifically designed to handle noisy data, such as data corruption or missing data. Its denoising property makes it particularly useful for missing data imputation. Instead of using the original input X, DAE

corrupts the input vector X to obtain a partially corrupted input X̃. The encoding function of DAE is defined as f(X̃) = σ(WX̃ + b), where σ represents the activation function. In practical applications, the replacement of the input X can be achieved by adding additional noise according to a certain distribution or setting specific elements of the input to zero. Alternatively, different corruption functions can be utilized to substitute selected data with prepared values that represent the missing status.

In [16], a Daily Load Profile (DLP) based missing value imputation framework is proposed, utilizing a DAE. The framework consists of four stages. In Stage 1, the ground-truth data is constructed from the dataset by replacing missing values with the average of previous time slots. This step ensures the availability of complete data for training and evaluation. Stage 2 involves implementing data corruption using two strategies: random corruption and block-wise missing. These strategies introduce artificial missing values into the dataset, simulating the presence of real-world missing data. In Stage 3, the AE structure is selected, including the number of layers, neurons, and activation functions. The DAE network is built based on these choices. Additionally, three different loss function configurations are explored to optimize the imputation performance. Finally, in Stage 4, the imputation performance is compared across six different configurations. These configurations are based on three corruption functions and two test set formulations. The performance evaluation helps assess the effectiveness of the DAE framework in imputing missing values in the DLP dataset. By following these four stages, the proposed framework provides a comprehensive approach to leverage DAE for missing value imputation in DLP data. It allows for the exploration of various network configurations and corruption strategies to optimize the imputation results.

In order to address the sensitivity to initial imputation in the DAE, tackles two contributing factors[17]. The first factor is the feedback of the initial imputed value to the neural network, which introduces bias towards this initial value as the ground truth. This bias may hinder the DAE from reaching a potentially better imputed value. To mitigate this issue, the authors employ a technique called metamorphic truth. This technique involves modifying the truth reported to the optimizer based on the model's predictions, allowing for a more flexible learning process. The second factor is the divergence between the input and output, which contradicts the learning objective of the DAE. To address this, the authors propose using the predicted values to impute subsequent training of the DAE. By utilizing the model's own predictions as input for further training, the DAE can better align its output with the imputed values, reducing the discrepancy between the input and output. By mitigating these two factors, the proposed approach aims to improve the robustness and effectiveness of the DAE for imputation tasks.

In the context of imputing missing data in multivariate time series, [18] employs a bidirectional Long Short-Term Memory (LSTM) block as an encoder. This LSTM block takes the entire series as input and generates a representation for each time step, enabling the model to capture temporal information effectively. Similar to the DAE approach, the authors corrupt the input by randomly setting certain values to zero. The model is then trained to reconstruct the clean input for each time step, thereby learning the correlation between variables. This approach leverages the bidirectional LSTM's ability to capture dependencies in both forward and backward directions, which aids in understanding the temporal dynamics of the time series data. By incorporating the DAE-like corruption and reconstruction process, the model can effectively impute missing values in the multivariate time series.

2.2.2. Variational Autoencoder (VAE) Application in Missing Data Imputation

In comparison to the vanilla AE, which learns deterministic functions f(X) and g(Z) using neural networks, the VAE extends the capabilities of AE by incorporating probabilistic modeling and inference through variational Bayesian methods.

Indeed, the VAE is a generative model that maps the input space X to the latent space Z. Unlike the vanilla AE, which focuses on minimizing the error between the input X and the reconstructed output Y, the VAE takes a probabilistic approach. It aims to maximize the probability distribution of the data p_θ(X) in terms of the model parameters θ, which is equivalent to solving a maximum likelihood problem.

In Equation (1), the marginal likelihood p_θ(X) is defined as the integral of the joint probability distribution p_θ(X,Z)over the latent variable Z. However, computing this integral directly is often intractable because both the latent variable Z and the generative model parameter θ are unknown.

$$p_\theta(X) = \int p_\theta(X,Z)dz = \int p_\theta(Z)p_\theta(X|Z)dz \quad (1)$$

To overcome this problem, [19-20] proposed training the generative model jointly with an inference model using variational inference. The data model can be seen as consisting of two parts: the generative model p_θ(X,Z) = p_θ(Z) p_θ(X|Z) and the inference model q_θ(Z|X) parameterized with ∅ which approximates the true posterior p_θ(X|Z).

By optimizing the terms of ∅ and θ simultaneously, a variational lower bound on the marginal likelihood can be derived. This is achieved by maximizing the objective function L(θ,∅,X) defined in Equation (2). The objective function consists of two terms: the expected reconstruction error, which measures the similarity between the input X and the reconstructed output given the approximate posterior distribution, and the Kullback-Leibler (KL) divergence between the approximate posterior and the prior p_θ(X).

$$L(\theta, \emptyset, X) = E_{q\emptyset(Z|X)}[\log p_\theta(X|Z)] - KL[q_\theta(Z|X)||p_\theta(Z)] \quad (2)$$

The KL divergence acts as a regularization term, ensuring that the learned latent distribution matches a predefined prior distribution, such as a multivariate Gaussian. This encourages similar input data to be represented by similar latent spaces.

In summary, the VAE learns a generative model that maps the input space X to the latent space Z. By incorporating an inference model and using variational inference, the VAE is able to approximate the true posterior distribution and optimize the model parameters to maximize the likelihood of the data. The objective function of the VAE includes a reconstruction term and a regularization term to balance the reconstruction accuracy and the adherence to the prior distribution.

In [21], the authors highlight the insufficient empirical coverage of VAE when handling missing data, resulting in underestimation and overconfident imputations. To address this limitation, they propose the application of β-VAE, which offer a framework for approximate Bayesian inference of deep generative models using the power likelihood. In statistical analysis, the power likelihood has demonstrated robustness against model misspecification. Therefore, achieving reliable coverage and well-calibrated uncertainty becomes crucial while avoiding overfitting.

Determining an appropriate value of β is also essential in this context. To tackle this challenge, the authors employ cross-validation to fine-tune β, enabling accurate multiple imputation. By incorporating β-VAEs within the power likelihood framework and utilizing cross-validation for β tuning, the authors enhance the model's capacity to deliver precise imputations while effectively addressing the uncertainty associated with missing values.

In [22], the authors introduce two innovative submodels based on Deep VAE (DVAE). These submodels serve as the foundation for developing two distinct soft sensor frameworks designed to handle scenarios with and without missing data. The first submodel, called Supervised DVAE (SDVAE), enhances the conventional VAE by incorporating both samples and labels. By adding the label Y as an extra dimension to the sample X, the SDVAE captures additional information for improved prediction accuracy

The second submodel, known as Modified Unsupervised DVAE (MUDVAE), modifies the benchmark latent distribution to align with the latent distribution learned by SDVAE. This modification is motivated by the observation that when the KL divergence between the learned latent distribution of MUDVAE and the modified benchmark latent distribution is sufficiently small, sampling from MUDVAE's latent distribution is equivalent to sampling from SDVAE's. This equivalence enables the authors to make predictions for test samples. By introducing these novel submodels and soft sensor frameworks, the authors offer effective solutions for both missing data and non-missing data scenarios.

In [23], the authors introduce Adversarial AE (AAE) as a generative variant of AEs that incorporates adversarial training. AAEs build upon the framework of VAEs but utilize a different loss computation, specifically the adversarial loss commonly used in Generative Adversarial Network (GAN). AAEs consist of three main components: the encoder, decoder, and discriminator networks. The encoder and decoder are similar to those in VAEs. However, an additional discriminator network is introduced in AAEs, which is trained to distinguish between samples drawn from the latent space and samples sampled from the prior distribution. On the other hand, the encoder and decoder networks are trained to generate latent space samples that can deceive or "fool" the discriminator. This adversarial training process encourages the encoder and decoder to learn a more informative and representative latent space representation. By incorporating adversarial training, AAEs aim to improve the quality of generated samples by capturing the underlying data distribution more accurately. The adversarial loss encourages the latent space samples to resemble the prior distribution, leading to enhanced sample generation capabilities.

In [24], the authors propose the use of Wasserstein AE (WAE), which utilize the Wasserstein distance (also known as Earth Mover's distance) as a distance metric for training. WAE are built upon the framework of VAEs and share similar components. However, the key distinction lies in the choice of distance metric used to measure the dissimilarity between the learned latent space distribution and a prior distribution. Instead of the traditional divergence metrics like KL divergence, WAEs employ the Wasserstein distance to quantify the discrepancy between the two distributions. The Wasserstein distance provides a more meaningful and geometrically interpretable measure of dissimilarity, particularly for high-dimensional data. By optimizing the Wasserstein distance, WAE aim to align the learned latent space distribution with the prior distribution. This alignment facilitates more accurate and meaningful data generation, as well as improved latent space representations.

2.2.3 Other Variants of Autoencoders Application in Missing Data Imputation

In addition to the research conducted on DAEs and VAEs, there is a body of literature that specifically investigates the structural aspects of AEs. These studies primarily focus on exploring the hidden layer structure, determining the optimal number of neurons, and selecting appropriate encoder and decoder architectures. The objective of such research is to enhance the overall performance and feature extraction capabilities of AEs. By investigating these structural aspects, researchers aim to improve the efficiency and effectiveness of AEs in various applications
.
In [25], the authors introduce the Stacked AE (SAE), also known as the Deep AE. The SAE is designed to address the limitations of traditional AEs by incorporating multiple layers of encoder and decoder units. In the SAE architecture, each layer's encoder takes the output from the previous layer's encoder as its input. Similarly, each layer's decoder takes the output from the previous layer's decoder as its input. This connectivity pattern creates a hierarchical structure, allowing the network to learn increasingly abstract representations of the input data as it progresses through the layers.

The training process of the SAE starts from the bottom layer and proceeds in a layer-wise manner. Each layer is pretrained individually as an AE, where it learns to reconstruct its input data. This pretraining step initializes the weights of the network and helps in capturing meaningful initial representations. After pretraining, the entire network is fine-tuned jointly using backpropagation and gradient-based optimization algorithms, allowing for the refinement of the learned representations.

By leveraging the layer-wise training strategy, the SAE is capable of learning complex and hierarchical representations of the input data. Each layer learns representations that build upon the knowledge acquired by the previous layers, resulting in the extraction of increasingly higher-level features or abstractions..

In [26], the authors introduce the Sparse AE (SAE), a variant of the AE that incorporates sparsity constraints during training. The goal of the SAE is to enforce that only a small fraction of neurons or hidden units in the AE are active for each input sample. This sparsity property allows the SAE to learn more efficient and concise representations of the input data. One common method to achieve sparsity is by adding a sparsity regularization term to the loss function, which encourages a limited number of hidden units to activate for each input, promoting sparsity in the learned representations. By incorporating sparsity constraints, Sparse AEs promote more compact representations.

Sparse AEs and Stacked AEs have distinct focuses and architectural characteristics. However, they can be combined in certain scenarios to leverage the benefits of both sparse representations and hierarchical learning. The combination of these approaches allows for the learning of sparse activations within each layer while capturing increasingly abstract features of the data.

In [27] the authors aims to leverage both row-row and column-column relationships collaboratively to impute missing values. The proposed method consists of two AEs: a user-based AE that captures column-column relationships and an item-based AE that captures row-row relationships. To obtain the final imputations, the predictions from the user-based and item-based AEs are averaged. This fusion of predictions from both AEs helps to improve the accuracy and reliability of the imputed values.

By incorporating both row-row and column-column relationships in the imputation process, the double AEs method offers a more comprehensive and collaborative approach for handling missing values, leading to enhanced imputation performance in various applications.

In [28], the authors propose a missing data imputation optimization method called SAE-CD, which combines the use of a Sparse AE (SAE) model with a Coordinate Descent (CD) algorithm to address the missing data problem. First, an SAE model is trained to learn a compressed representation of the input data while capturing its essential features. By minimizing the reconstruction error, the SAE model can effectively fill in missing values and impute the incomplete data.

Subsequently, the CD algorithm is employed iteratively to refine the imputations. The CD algorithm optimizes the imputed values by adjusting them in a coordinate-wise manner, ensuring the authenticity and accuracy of the imputations. This iterative process helps to fine-tune the imputed values and enhance the overall imputation quality. This approach leverages the reconstruction capabilities of the SAE model and the refinement power of the CD algorithm to generate reliable and accurate imputations, facilitating further analysis and modeling tasks that require complete data.

In [29], the authors present an ensemble modeling technique applied to DAEs, which involves training multiple DAEs on different subsets of the data. Each DAE focuses on learning local transform functions that capture specific patterns or structures within the data. These local transform functions are then combined to form the final transform function. To ensure the robustness and generalization of the ensemble model, a linear weighting function is applied to each training sample, the global mapping function is estimated as:

$$f(\cdot) \triangleq \lambda_1(\cdot) f_1(\cdot) + \lambda_2(\cdot) f_2(\cdot) \quad (3)$$

Where $\lambda_1(\cdot)$ and $\lambda_2(\cdot)$ are the weighting coefficient functions. And for overcoming overfitting problem, the constraints $0 \leq \lambda_1(\cdot), \lambda_2(\cdot) \leq 1$ and $\lambda_1(\cdot) + \lambda_2(\cdot) = 1$ are added, allowing the model to adapt and assign appropriate importance to different instances and to mitigate the overfitting problem. By employing this ensemble modeling technique, the approach providing a more comprehensive and accurate representation of the data and to enhance the model's ability to handle complex patterns and improve its performance in various applications.

In [30], the authors introduce an AE variant called the Orthogonal-Least-Squares (OLS)-based AE. The process begins by incrementally adding neurons to the hidden layer, with each neuron corresponding to specific features extracted from the input variables. In each step, the best neuron is selected using the OLS method. This method aims to maximize the increase in the explained variance of the desired output, considering all existing candidates. By enforcing orthogonality among the hidden neurons, Orthogonality promotes diversity among the learned representations, preventing redundancy and facilitating more effective feature extraction. The proposed OLS-based AE can helps address the overfitting issue that can arise when using a large number of hidden neurons. Additionally. It offers an approach to optimize the composition of the hidden layer, ensuring the extraction of relevant and non-redundant features, which can lead to improved performance and generalization capabilities in various applications.

In [31], the authors propose the use of the Monte Carlo Dropout (MCD) technique within the layers of an AE and VAE to obtain multiple generated inputs and enhance imputation accuracy. This technique involves applying dropout, a regularization method, during the training phase of the AE and VAE decoder layers. Dropout randomly sets a fraction of the neurons to zero during each training iteration, the proposed MCD-AE model generates multiple imputed values for each missing data point. These multiple generated inputs allow for capturing the underlying distribution of the given data without solely relying on individual data points.

To obtain the final imputed value, the output values from the model are averaged. This averaging process provides a more comprehensive representation of the imputed value, incorporating the uncertainty and variability captured by the multiple generated inputs. The MCD-AE model improves the accuracy of imputation by considering a range of possible values for each missing data point, resulting in more robust and reliable imputation results.

In [32], the authors tackle the task of traffic data imputation, which involves estimating missing values in traffic flow features to enhance the performance of related applications. They highlight the limitations of traditional imputation methods that primarily focus on isolated traffic data sensors or road sections, thereby lacking the ability to capture complex spatial-temporal features. To address these limitations, the authors introduce a novel ensemble model called the Ensemble Convolutional AE. This model comprises multiple AEs, each trained using different input feature maps. These feature maps are created by filling the missing positions with zero values and historical average values, respectively.
In the ensemble model, the AEs process the input feature maps, denoted as X^1 (with missing positions filled with zeros) and X^2 (with missing positions filled with historical average values). Consequently, two matricized outputs, Y^1 and Y^2, are generated. These outputs are combined using a parameterized data aggregation rule:

$$Y = \alpha \times Y^1 + (1 - \alpha) \times Y^2 \quad (4)$$

where α is a self-adapting weight that determines the contribution of each autoencoder's output, which are updated using backpropagation during the training process. To assess the effectiveness of their proposed method, the authors conduct comprehensive experiments on diverse missing data scenarios. The experimental results demonstrate that the ensemble convolutional AE approach achieves superior imputation accuracy and robustness compared to baseline methods.

2.3. GRADIENT BOOSTING MACHINES (GBM)

Gradient Boosting Machines (GBM)[33] is a machine learning algorithm primarily used for supervised learning tasks, such as regression and classification. It falls under the category of boosting ensemble methods, which combine predictions from multiple weak learners, typically decision trees, to create a robust predictive model. GBM is utilized for constructing predictive models by employing back-fitting and non-parametric regressions. Unlike the approach of building a single model, GBM initiates the process by creating an initial model and then continuously fitting new models through the minimization of a loss function, aiming to generate the most accurate model possible. The ultimate goal of GBM is to find a functionF(x) [34], which minimize its loss function L(y,F(x)) through iterative back-fitting, and it can be expressed as

$$F^* = \underset{F}{\operatorname{argmin}} E_{y,x} L(y, F(x)) \tag{5}$$

As defined, a boosted model is a weighted linear combination of base learners. It can be expressed as:

$$F(x; \{\beta_m, \theta_m\}_1^M) = \sum_{m=1}^{M} \beta_m h(x: \theta_m) \tag{6}$$

Here, $h(x: \theta)$ represents a base learner parameterized byθ.These base learners are considered weak learners because their individual predictions only marginally outperform random guessing. However, it has been shown that recursive learning with weak learners can achieve performance comparable to that of strong learning algorithms.

In the equation, the weights β_m represent the importance or contribution of each base learner to the final model. The base learners are combined linearly, with each learner's prediction multiplied by its corresponding weight and then summed up. This weighted combination allows the boosted model to capture complex patterns and relationships in the data

The goal of boosting methods, as discussed in [35], is to optimize the function space by iteratively constructing an estimated function F(x) in the additive form:

$$F(\mathrm{x}) = F^I(x) = \sum_{i=0}^{I} F_i(\mathrm{x}) \tag{7}$$

Here, I represents the number of iterations, F_0 is the initial guess, and $\{F_i\}_{i=1}^I$ denotes the function increments or boosts. Each F^i contributes to the ensemble prediction at iteration i. Typically, a base learner model $h(x, \theta)$ is selected to form these incremental functions, and the process follows a "greedy stage-wise" strategy.
To begin, the model initializes F_0 as $h(x, \theta_0)$. Then, a new function $h(x, \theta_i)$ is chosen at each iteration to closely align with the negative gradient $g_i(x)$long the observed data:

$$g_i(x) = E_y \left[\frac{\partial L(y, F(x))}{\partial F(x)} | x \right]_{F(x)=F^{i-1}(x)} \tag{8}$$

In the above equation, $L(.)$ represents the loss function. By assigning a step size ρ for each step, the new function increment is selected to be highly correlated with $g_i(x)$. Assuming we have N samples, and ρ_i is the line search on the direction of steepest-descent. the optimization problem can be solved using the classic least-squares minimization method:

$$(\rho_i, \theta_i) = \underset{\rho, \theta}{\operatorname{argmin}} \sum_{j=1}^{N} \left[-g_i(x_j) + \rho h(x_j, \theta) \right]^2 \tag{9}$$

The objective is to find the optimal values of ρ_i and $\theta_i$that minimize the squared difference between the negative gradient and the base learner's prediction on the training data points x_j. This iterative process continues until the desired number of iterations I is reached.

3. METHODS

3.1 APPLYING THE SPIRIT OF GBM TO ENHANCE AE TRAINING

Many studies focus on improving the components of AEs to enhance imputation performance, often overlooking the significance of training data quality. However, when the training data is flawed, it inevitably constrains the efficacy of

training. To attain a superior training dataset, leveraging insights from previous training efforts becomes imperative. A noteworthy approach for such reuse entails employing GBM.

In recent years, the concept of "boosting" has emerged as a promising direction to enhance model performance by leveraging ensemble techniques. It suggests that combining multiple models can yield better results than using a single model. Among various boosting methods, Gradient Boosting has demonstrated its effectiveness, especially in dealing with complex and high-dimensional datasets to achieve accurate predictions. The core idea behind Gradient Boosting is to sequentially train models, with each subsequent model focused on predicting the errors of the previous models. This iterative process is guided by a loss function that measures the quality of predictions and residuals. Commonly used loss functions in gradient boosting include Mean Squared Error (MSE) for regression problems and cross-entropy loss for classification problems.

While Gradient Boosting has been successfully applied in various domains, such as classification, decision forests, credit card fraud detection, and medical data analysis, [35-39], there have been limited works exploring its application in missing data imputation. Because GBM is conventionally employed in supervised learning scenarios where datasets are complete, its direct applicability to missing data is constrained by the inherent absence of certain information.

Nevertheless, drawing inspiration from the foundational principles of GBM, we can adapt and employ analogous strategies within our proposed methodology. Despite the inherent unsuitability of typical supervised learning approaches for datasets with missing values, we can extract insights from the essence of GBM to augment the efficacy of our proposed method. In the following sections, we will present the detailed methodology of our proposed approach and evaluate its performance through extensive experiments, comparing it with existing methods in the field.

3.2 GRADIENT BOOST MODULE IN AE FOR MISSING DATA IMPUTATION

Typically, GBM is not directly employed for training AE, as these two fall under distinct machine learning paradigms. AEs are neural networks predominantly employed for unsupervised learning and dimensionality reduction. Nonetheless, the fundamental objective of GBM is to discover a function F(x) that minimizes its loss function L(y, F(x)) through iterative back-fitting. However, by drawing inspiration from the fundamental principles of GBM, we have the opportunity to adapt and implement similar strategies within the framework of our proposed methodology.

The fundamental concept of GBM implies that we can segment the entire AE training process into multiple stages. At each stage, training builds upon the achievements of the preceding stage, leading to an optimization process. In the domain of missing data imputation, where input data often contains gaps and irregularities, it becomes plausible to utilize data with enhanced reconstruction quality from one stage as the input for the subsequent stage. This essentially means that each stage of AE learning serves as a base learner, persistently learning and enhancing its performance iteratively. Consequently, this approach is expected to yield similar benefits in terms of data reconstruction and imputation.
Regarding the basic AE described in section 3.1, it consists of an encoding function f(X) that transforms the input X into a code in the latent space. Z represented as Z = $W_e X + b_e$. The code Z is then transformed by a decoding function g(Z), which reconstructs the input as Y = $W_d Z + b_d$.

For a basic AE, the estimated function F(x) can be expressed as the composition of the encoding and decoding functions:

$$\mathrm{F}(x) = \mathrm{g}(\mathrm{f}(x)) \tag{10}$$

The parameters of the AE are denoted as $\theta = \{W_e, W_d, b_e, b_d\}$, and the loss function is the least-squared loss function, which can be expressed as $L(X,\mathrm{Y}) = \|X - Y\|^2$, where X represents the input data and Y is the reconstructed output.
In the boosting context, utilizing the least-squared loss, the optimization process can be expressed as:

$$(\rho_i, \theta_i) = \underset{\rho,\theta}{\operatorname{argmin}} \sum_{j=1}^{N} \left[y_j - (F_{j-1}(x_j)) + \rho h(x_j;\theta)\right]^2 \tag{11}$$

Here, y_j represents the target output, $F_{\mathrm{j}-1}(x_j)$ denotes the ensemble prediction from the previous iteration, which implies the previous iteration's reconstruction back-fitting to input X, $h(x_j;\theta)$ is the base learner's prediction, and ρ_i and θ_i are the optimal values of the step size and the base learner's parameters at the i-th iteration, To solve for F(x), a stage-wise model can be used:

$$F_i(x) = F_{i-1}(x) + \rho_i h(x;\theta_i) \tag{12}$$

Nonetheless, practical application of this concept necessitates empirical validation, appropriate dataset selection, parameter tuning, and thorough evaluation to ensure its effectiveness within specific domains and problem contexts. This

staged AE training technique presents a promising approach for addressing challenges in missing data imputation while adhering to the iterative optimization philosophy of GBM. We propose two methods for implementing GBM concept within AE for missing data imputation.

3.3. SHORT-TERM RECONSTRUCTION WITH ITERATIVE UPDATES (STR-IU) METHOD

The Short-Term Reconstruction with Iterative Updates (STR-IU) method entails periodically direct replacement of training data X with the reconstructed values Y obtained from the AE. This approach assumes that the training of the prior base learner, which in AE corresponds to the previous training iteration, is effective and results in more informative reconstructions compared to the original input data. By substituting the missing input data with the improved reconstructions, the training process prioritizes the utilization of enhanced information from the preceding training iteration. To further refine the quality of the replacement data, a difference threshold parameter is employed.The parameters for applying STR-IU method in AE training experiments were determined as follows,

• Substitution Ratio (P_{SR})

P_{SR} was set to determine the proportion of the best-fit reconstructions that are replaced. P_{SR} governs the extent to which the training from the previous epoch contributes and is restricted within the range of $0 \leq P_{SR} \leq 1$. This constraint ensures a controlled updating process, progressively enhancing the overall ensemble prediction during the epochs of the reconstruction training interval. The choice of P_{SR} is crucial as it affects the balance between the original training data and the reconstructed data.

• Difference Threshold (P_{DT})

P_{DT} involves comparing the training results before and after optimization. This threshold ensures that replacement only occurs when the difference between the two exceeds a certain threshold value. Frequently replacing training data solely to achieve lower RMSE values could lead to excessive and potentially unhelpful replacements, potentially resulting in an insufficient number of epochs for training. Setting a threshold value is essential to enhance the quality of replacement data and prevent overly frequent and dense replacements

• Nnumber of epochs in each replace iteration ($\boldsymbol{N_{period}}$).

It determines the duration of the training period, which is the number of epochs, allocated for the reconstruction process. Please refer to Table 1 for a description of the STR-IU method algorithm.

Table1. the algorithm of STR-IU methods

Algorithm 1: STR-IU method

Require: training dataset X, Difference Threshold $\boldsymbol{P_{DT}}$, Substitution Ratio $\boldsymbol{P_{SR}}$,Number of short training iteration $\boldsymbol{N_{period}}$,

1 for Number of training iterations $\boldsymbol{N_{period}}$ do
2 Train AE with X to produce back fitting reconstruction y_{fit}
3 $when\ [err(X, y_{j+1}) - err(X, y_j)]$>$\boldsymbol{P_{DT}}$, y_{fit}=$(1 - \boldsymbol{P_{SR}})X + \boldsymbol{P_{SR}} y_{j+1}$
4 Until epoch=$\boldsymbol{N_{period}}$,take X as$F$(X), and byF_{j+1}(X) = $F_i(x) + \rho_i h(x; \theta_i)$
5 Substitutive X with P_{SR},$X_{iter+1} = (1 - P_{SR})X_{iter} + P_{SR} y_{fit\_iter}$
6 end for

The STR-IU approach entails replacing the missing values in the input data with the reconstruction values predicted by each of the previous training iterations, the fundamental concept of the STR-IU method is to continually update input predictions by integrating the reconstructions from each iteration, replacing missing values with imputed values, and minimizing the reconstruction error to enhance the quality of missing data imputation. Please refer to Figure 2 to visualize the AE training process of STR-IU.

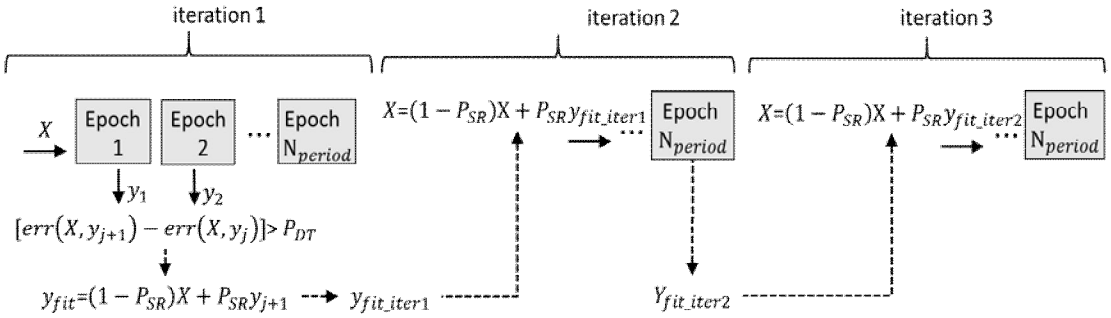


Figure2. STR-IU method in AE Training process

3.4. LONG-TERM RECONSTRUCTION WITH A SINGLE UPDATE (LTR-SU) METHOD

The Long-Term Reconstruction with a single Update (LTR-SU) method is focused on achieving the most accurate reconstruction of missing data within a reconstruction training interval to uncover underlying information. It also use P_{SR} to determine the proportion of the best-fit reconstructions that are replaced. Subsequently, the most accurate reconstructionis merged with the original missing data using a specified recombination ratio P_{RR}. The recombination training data denoted as X_{remix}, is employed during the latter part of the training stage to create a consolidated model. The parameters for applying LTR-SU method in AE training experiments were determined as follows.

• Substitution Ratio (P_{SR})
Similar to STR-IU method.

• Recombination Ratio (P_{RR})
P_{RR} is to determine the proportion of the best-fit reconstructions that are recombined with the original training data once the best-fit reconstruction process is completed. P_{RR} influence how much emphasis is placed on the recombined data during training.

• Nnumber of epochs for the best-fit training (N_{best_fit}).
N_{best_fit} is similar to N_{period}, It defines the duration of the training period. However, the reconstruction training period is longer, and it is trained only once.Please refer to Table 2 for a description of the LTR-SU method algorithm.

Table2. the algorithm of LTR-SU methods

Algorithm 2: LTR-SU method
Require: Recombination Ratio $\boldsymbol{P_{RR}}$, Substitution Ratio $\boldsymbol{P_{SR}}$, Number of training epochs for fit reconstruction $\boldsymbol{N_{best\_fit}}$, Number of general training epochs $\boldsymbol{N}$,

Stage1:get fit reconstruction dataset y_{best}
for j in 1: N_{best_fit}do
Train AE with X to produce back itting reconstruction $y_{bestfit}$
when $\text{err}(X, y_{j+1}) \leq \text{err}(X, y_j), y_{bestfit} = (1 - P_{SR})y_{bestfit} + P_{SR}y_{j+1}$
end for
return fit reconstruction dataset $y_{bestfit}$
Stage 2: replace X to $X_{remix}$and start training
take X as$F(X)$,by $F_{j+1}(X) = F_{i-1}(x) + \rho_i h(x;\theta_i)$, and$X_{remix} = (1 - P_{RR})X + P_{RR}y_{bestfit}$
for j in N_{best_fit} + 1: N do
Train AE with X_{remix}
end for

The LTR-SU approach similarly aims to reconstruct training data from the original missing data. However, the core aspect of this method involves its execution over an extended period to achieve better reconstruction for training, followed by continuous training without altering the training data. Please refer to Figure 3 to visualize the AE learning process of LTR-SU method

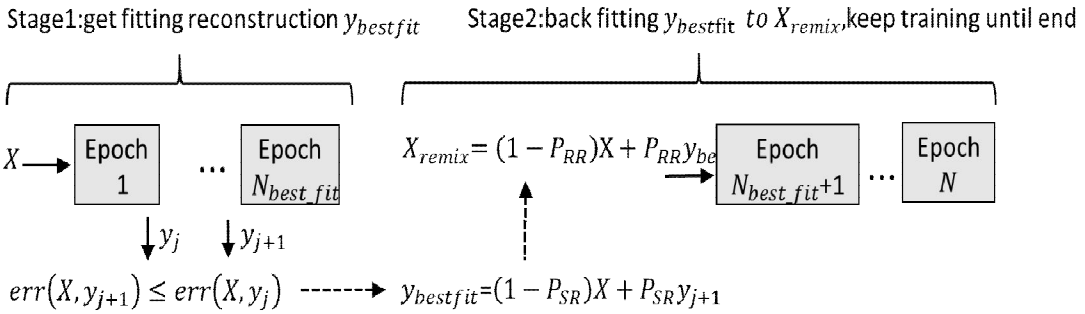


Figure 3. LTR-SU method in AE Learning process

4. EXPERIMENTS

4.1. DATASET DESCRIPTION

For the purpose of expedited and straightforward comparison between the experimental and control groups, we have opted for a dataset comprising exclusively numerical content. Simultaneously, to investigate whether the number of attributes and instances affects the training improvement outcomes, we aim to identify datasets with significant differences in these aspects. Finally, to confirm whether our proposed method is influenced by dataset characteristics and exhibits effectiveness solely for specific data types, we intend to select datasets from diverse application domains.

After conducting a thorough search and comparison. As shown in Table 1, our proposed approach is evaluated on 3 datasets sourced from the UCI Machine Learning Repository. The datasets used include the following:

• Cardiotocography Dataset

This dataset consists of measurements of fetal heart rate (FHR) and uterine contraction (UC) features on cardiotocograms, which were classified by expert obstetricians.

• Parkinson's Telemonitoring Dataset

This dataset includes a range of biomedical voice measurements from 42 individuals with early-stage Parkinson's disease who participated in a six-month trial of a telemonitoring device for remote symptom progression monitoring.

• Statlog (Landsat Satellite) Dataset

This dataset contains multi-spectral values of pixels in 3x3 neighborhoods in a satellite image. The attributes in this dataset are numerical, ranging from 0 to 255.

We have compiled certain data characteristics pertinent to the experiments in Table 3 below.

Table 3. Datasets

Dataset	Cardiotocography	Parkinsons Telemonitoring	Statlog(Landsat Satellite)
Instances	2126	5875	6435
Attributes	21	19	36
Attribute Type	Real	Integer, Real	Integer
Has Missing value	N	N	N
Offering Testing Dataset	N	N	Y

These datasets originate from diverse application domains, spanning from medical measurements to satellite imagery. As a result, each dataset possesses unique characteristics tailored to its specific domain of use. This diversity in data nature, ranging from healthcare-related measurements to satellite image features, provides a valuable set of challenges and opportunities for assessing the impact of dataset characteristics on the proposed methodology.

In addition, the datasets can be classified into two distinct groups based on the number of attributes: 23, 22, and 37. Similarly, the number of instances falls into two categories: 2126, and 5875, 6435. This classification serves as a structured basis for professional and academic comparison.

4.2. MODELS FOR COMPARISON

In our research, we primarily investigate the impact of updating input data during AE training, with a deliberate focus on avoiding extensive discussions related to optimization algorithms and loss functions. To assess the influence of modifying

input data, we employ DAE models and examine how different optimization mechanisms affect the performance of our proposed methods. We choose DAE over VAE due to VAE's inclusion of a KL Divergence term, which remains unaffected by changes in input data.

In our iterative model enhancement processes$F_i(x) = F_{i-1}(x) + \rho_i h(x;\theta_i)$, the primary focus is on improving $F_i(x)$. Within the realm of AE, there exist numerous optimization algorithms well-suited to address the optimization of $\rho_i h(x;\theta_i)$. As part of our evaluation methodology, we have selected two classical optimization algorithms: Stochastic Gradient Descent (SGD) and Adam (Adaptive Moment Estimation) to serve as baseline models. These choices align with the rigorous standards of evaluation in machine learning and deep learning research. SGD and Adam are widely recognized and respected optimization techniques, and their use as benchmarks will provide a robust basis for assessing the performance of our proposed approach.

In order to streamline the subsequent figure presentations, abbreviated references for the relevant comparison methods have been established, as delineated in Table 4 below.

Table 4. Models for comparison

Model Description	Abbreviation
DAE with Original Stochastic Gradient Descent optimizer	SGD
Applying STR-IU method in DAE with SGD	STR-IU +SGD
Applying LTR-SU method in DAE with SGD	LTR-SU +SGD
DAE with Original Adaptive Moment Estimation optimizer	Adam
Applying STR-IU method in DAE with Adam	STR-IU + Adam
Applying LTR-SU method in DAE with Adam	LTR-SU + Adam

4.3. EXPERIMENTAL SETUP

After randomizing the original dataset, we partitioned it into training and testing datasets according to a specified ratio. Specifically, the first 70% of the dataset were allocated to the training data, while the remaining 30% constituted the testing data. Given that the original dataset is devoid of any missing or erroneous values, a subset of the original values within the training dataset is randomly substituted with -10 to signify missing values by a specified ratio. The choice of -10 is based on the values observed in three experimental datasets, thus resembling an outlier.

For the experiments, we used different corruption levels (5%, 15%, and 25%) on three datasets. The number of input dimensions was determined based on the number of attributes, and the encoding dimension was fixed at 12. We initialized weights and biases with random values. Testing models were implemented using PyTorch, and data were scaled to a specific range between 0 and 1, based on the minimum and maximum values in the dataset.

The number of training epochs for the models ranged from 50 to 600, with intervals of 50. For network parameter tuning, we employed both the Adam optimizer and the SGD optimizer. In the case of SGD, we set a learning rate of 0.1, a momentum of 0.9, a batch size of 256, and used a MSE loss function. For Adam, we utilized a learning rate of 0.001, with other parameters set to their default values.

The RMSE metric is a commonly used measure for assessing model prediction accuracy, quantifying the square root of the mean of the squared differences between actual and predicted values. Lower RMSE values indicate better model performance. The calculation of RMSE is defined as follows: Given x_i as the actual value and y_i as the predicted value.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{N}(x_i - y_i)^2} \quad (13)$$

5. RESULTS AND DISCUSSION

5.1. APPLYING STR-IU METHOD IN DAE TRAINING

These parameter settings were chosen based on careful experimentation and observations, aiming to strike a balance between various factors influencing the STR-IU method AE learning process.

- Parameter Setting for Substitution Ratio (P_{SR})

The experimental results indicated that setting P_{SR} to either 1e-02 or 1e-04 had no significant effect. This lack of impact can be attributed to the fact that STR-IU method directly replaces a portion of the original data with the reconstruction data

produced during training, aiming to enhance the quality of the training data for improved training performance. STR-IU method relies on the quality of reconstruction data obtained at each iterations, which tends to be less stable. Therefore, a large-scale replacement may lead to interference and counterproductive results. As a result, we explored the range of P_{SR} values such as 1e-05, 1e-06, and 1e-07 to find suitable settings.

• Parameter Setting for Difference Threshold (P_{DT})

Determining an appropriate value for P_{DT} is a critical consideration. If the value is too small, it may not effectively achieve the threshold effect. Conversely, if the value is too large, it might result in a reduction in the number of replacements, which may also not be conducive to effective training. To address this, experiments were conducted on each of the three datasets using SGD, initially training for 200 epochs. The differences in RMSE between consecutive epochs were recorded and sorted in ascending order to observe the variations in numerical differences. The training was then repeated for 400 and 600 epochs, allowing for a comparison of how the numerical differences evolved with different epoch durations and datasets. A range of threshold values from 1e-04 to 3e-04 was adopted as they strike a balance between not being too large (resulting in too few replacements) and still exhibiting significant variability for observing the threshold effect. These values consistently ranked within the top 35-45% and 10-25% across different datasets and training durations.

• Parameter Setting for Number of Epochs in Iteration (N_{period})

In our experiments, we set values of 10 and 25 for these periods.

5.2. APPLYING LTR-SU METHOD IN DAE TRAINING

• Parameter Setting for Substitution Ratio (P_{SR})

Experiments were conducted on three different datasets, and effective P_{SR} values were found to be approximately in the range of 1e-02, 1e-04, and 1e-06.

• Parameter Setting for Recombination Ratio (P_{RR})

Similarly, after testing on three different datasets, P_{RR} values were found to be approximately in the range of 1e-04, and 1e-06.

• Parameter Setting for Number of Epochs for Best Fit Training (N_{best_fit})

To ensure the quality of reconstruction data, it is essential to set N_{best_fit} when the RMSE values has roughly converged. Referring to the RMSE convergence variations during the training of DAE for 200, 400, and 600 epochs, N_{best_fit} was set to 30, 40, and 50. This choice of N_{best_fit} helps to achieve higher-quality reconstruction and contributes to the overall training process of the LTR-SU method.

The experimental parameters involve the application of the STR-IU and LTR-SU methods to three datasets, outlined as Table 5 and Table 6.

Table 5. Parameters for 3 Datasets in applying STR-IU method in DAE

	Cardiotoco		Parkinsons		Statlog	
	SGD	Adam	SGD	Adam	SGD	Adam
P_{SR}	1e-06	1e-06	1e-07	1e-05	1e-06	1e-06
P_{DT}	1e-04	3e-04	2e-04	3e-04	2e-04	1e-04
N_{period}	10	10	10	25	25	10

Table 6. Parameters for 3 Datasets in applying LTR-SU method in DAE

	Cardiotoco		Parkinsons		Statlog	
	SGD	Adam	SGD	Adam	SGD	Adam
P_{SR}	1e-02	1e-06	1e-02	1e-02	1e-06	1e-04
P_{RR}	1e-04	1e-04	1e-04	1e-06	1e-06	1e-06
N_{best_fit}	40	30	30	30	40	40

5.3 ISSUES ASSOCIATED WITH FINDING OPTIMAL PARAMETER COMBINATIONS.

In practical implementations, both SGD and Adam algorithms typically avoid fixing the random seed. While this practice contributes to improved learning performance in models, it introduces a challenge: even with identical initial values, the non-fixed nature of the random seed may result in slight numerical variations upon repeated executions. Although these differences may be negligible, they can become inconspicuous, especially during the process of searching for optimal parameters where the inherent variability of result values may not be apparent. In such circumstances, the fluctuation of random seeds can introduce subtle interference, potentially leading to misjudgments in the evaluation.

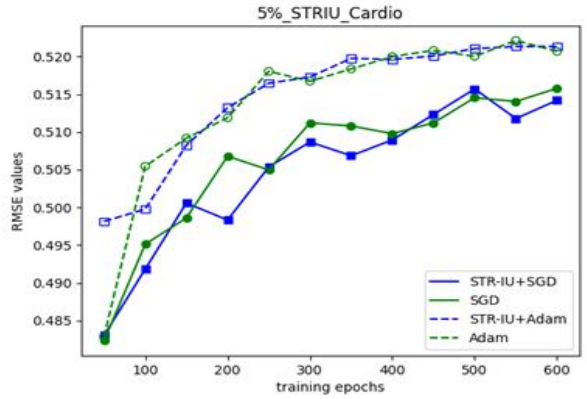
To mitigate this situation, we practice to execute multiple consecutive runs using the same initial values and subsequently compute the average of these runs. This approach aims to reduce the impact of random seed variability on the results, ensuring greater stability and minimizing the potential for misinterpretation caused by stochastic influences.

Moreover, during the sequential search for optimal parameter combinations, there may be instances in which a specific parameter demonstrates favorable performance with two or three distinct values. While one approach is to consider all these values for comparison, doing so may significantly increase the time spent. Therefore, it is common to evaluate by observing the results of adjacent values. After all, the impact of parameters on the model is unlikely to be highly drastic. If a particular value performs well, the performance of adjacent values should also be relatively good. Thus, at times, comparing the performance of neighboring values is used to determine the optimal parameter value.

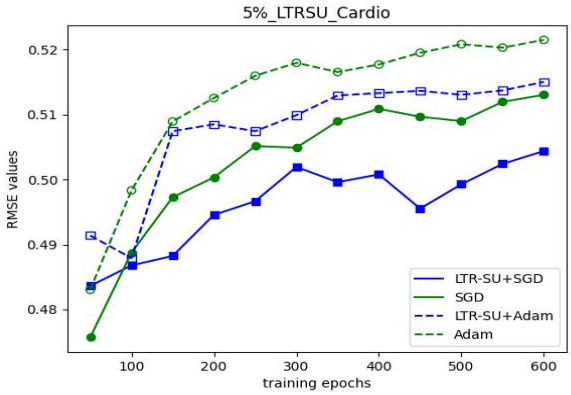
5.4 THE EXPERIMENT INVOLVES COMPARING 3 DATASETS USING THE RMSE METRIC

To compare the effectiveness of the STR-IU and LTR-SU methods, we divided Figure into two categories: (a) series depicting the differences between STR-IU and the SGD and Adam methods, while (b) series illustrating the disparities between LTR-SU and the SGD and Adam methods. We represented STR-IU+SGD and LTR-SU+SGD with solid squares connected by solid lines, SGD with solid circles connected by solid lines, STR-IU+Adam and LTR-SU+Adam with empty squares connected by dashed lines, and Adam with empty circles connected by dashed lines. Based on the experimental results of the three datasets at different missing data percentages, we generated a total of nine figures. First, the results for Cardio, Parkin, and Statio datasets with missing data at 5% are depicted in Figures 4, 5, and 6, respectively. Subsequently, the results for missing data at 15% are shown in Figures 7, 8, and 9, while the results for missing data at 25% are displayed in Figures 10, 11, and 12.

5.4.1 RMSE Value Variations for Each Dataset with 5% Missing Data

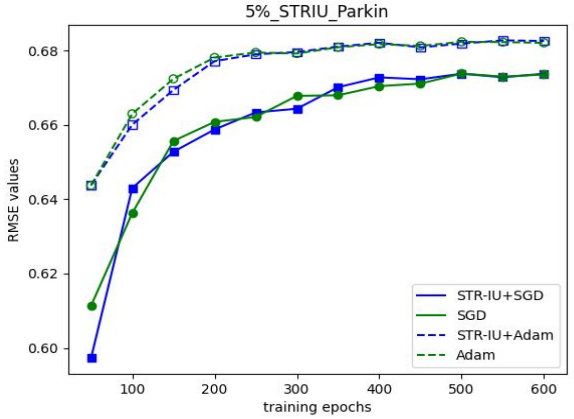


(a)

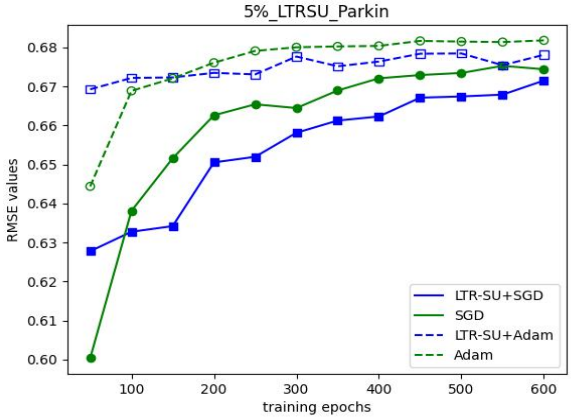


(b)

Figure 4. Comparison with SGD and Adam on Cardio Dataset with 5% Missing Data; (a) STR-IU Method; (b) LTR-SU Method

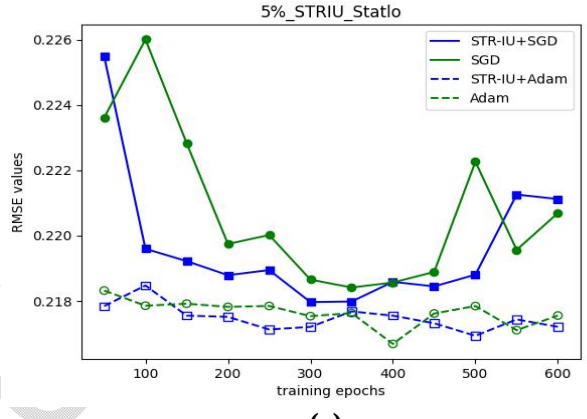


(a)

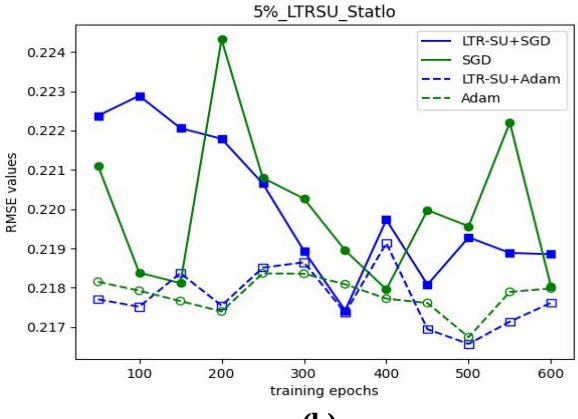


(b)

Figure 5. Comparison with SGD and Adam on Parkin Dataset with 5% Missing Data; (a) STR-IU Method; (b) LTR-SU Method



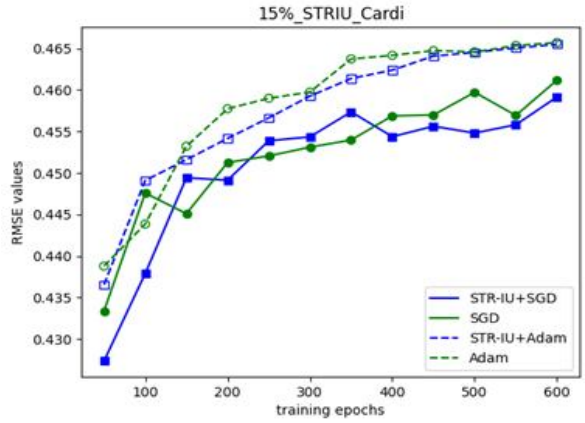
(a)



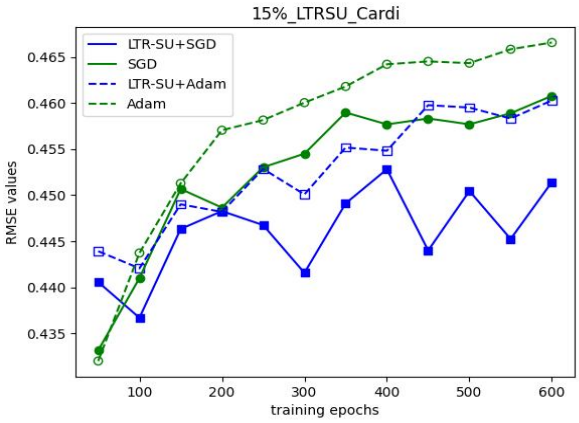
(b)

Figure 6. Comparison with SGD and Adam on Statio Dataset with 5% Missing Data; (a) STR-IU Method; (b) LTR-SU Method

5.4.2 RMSE Value Variations for Each Dataset with 15% Missing Data

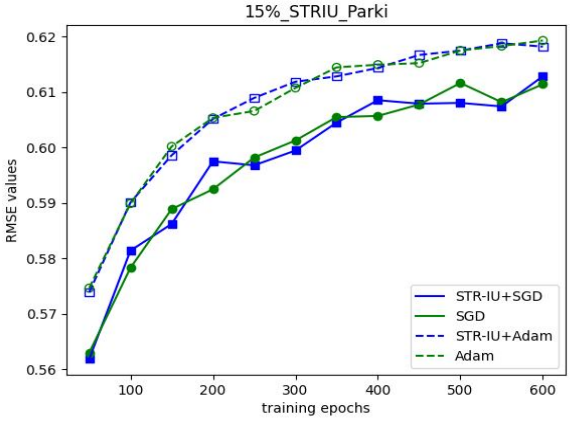


(a)

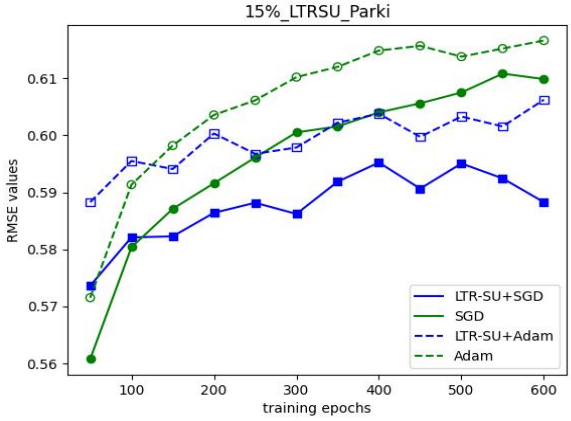


(b)

Figure 7. Comparison with SGD and Adam on Cardio Dataset with 25% Missing Data; (a) STR-IU Method; (b) LTR-SU Method

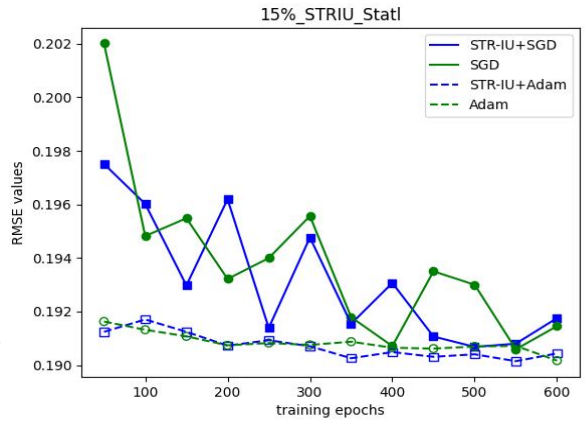


(a)

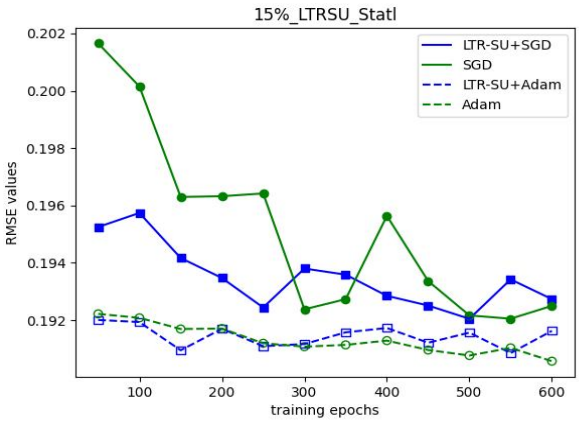


(b)

Figure 8. Comparison with SGD and Adam on Parkin Dataset with 25% Missing Data; (a) STR-IU Method; (b) LTR-SU Method



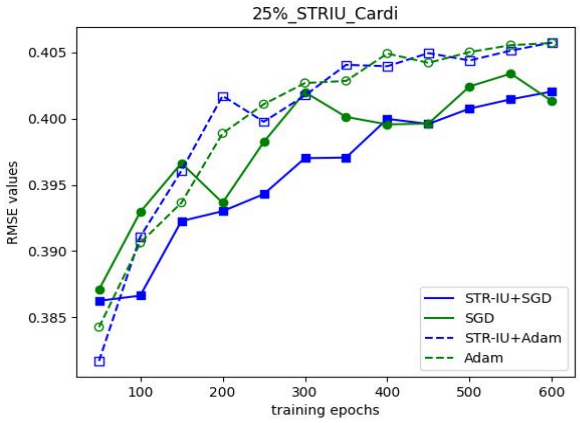
(a)



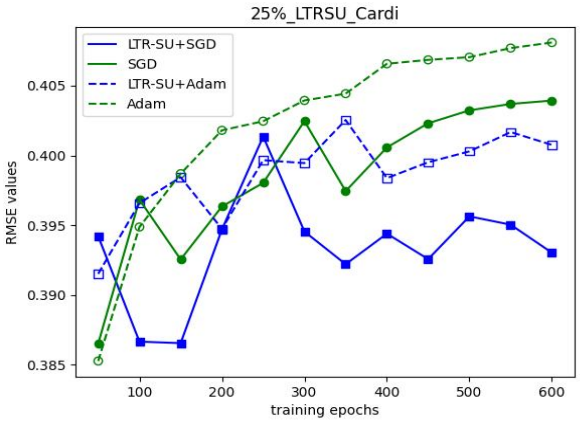
(b)

Figure 9. Comparison with SGD and Adam on Statio Dataset with 25% Missing Data; (a) STR-IU Method; (b) LTR-SU Method

5.4.3 RMSE Value Variations for Each Dataset with 25% Missing Data

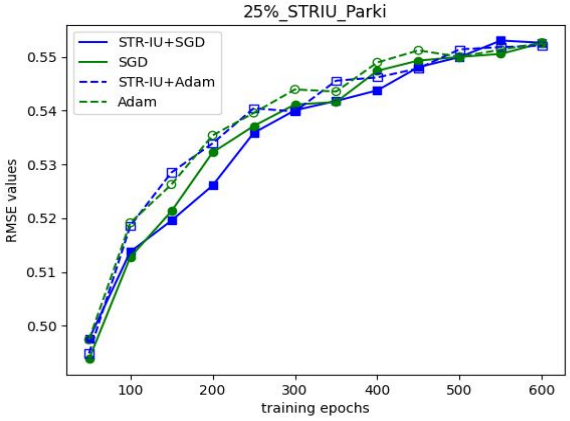


(a)

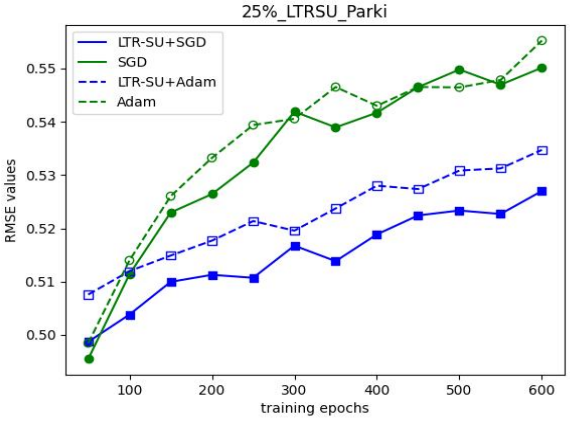


(b)

Figure 10. Comparison with SGD and Adam on Cardio Dataset with 25% Missing Data; (a) STR-IU Method; (b) LTR-SU Method



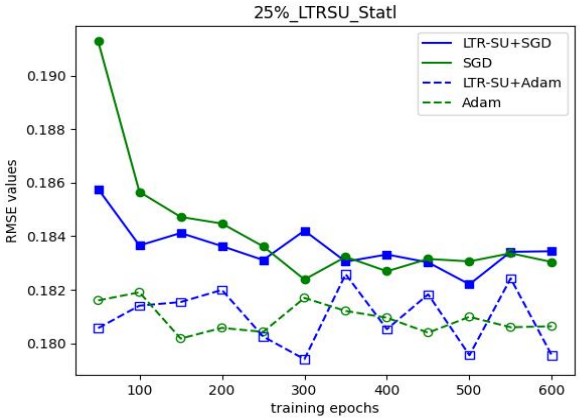
(a)



(b)

Figure 11. Comparison with SGD and Adam on Parkin Dataset with 25% Missing Data; (a) STR-IU Method; (b) LTR-SU Method

(a)



(b)

Figure 12. Comparison with SGD and Adam on Statio Dataset with 25% Missing Data; (a) STR-IU Method; (b) LTR-SU Method

In this experiment, LTR-SU consistently outperforms STR-IU, regardless of whether SGD or Adam is employed as the optimizer. Additionally, when comparing the use of SGD and Adam optimizers, it becomes evident that the improvements achieved with Adam are less pronounced compared to the enhancements observed with SGD.

In summary, LTR-SU maybe outperforms STR-IU, especially when SGD is employed as the optimizer. This outcome can be rationalized by the fact that the reconstruction data refined by LTR-SU undergoes continuous optimization over an

extended period, leading to enhanced stability and accuracy when compared to the reconstruction data generated during the STR-IU iteration.

STR-IU's frequent and rapid data replacements can potentially compromise its stability. In the STR-IU approach, replacements occur whenever there is a reduction in the error of the reconstruction data, without taking into account its overall stability. Conversely, LTR-SU executes a single replacement with thoroughly optimized reconstruction data and allocates half of the epoch time for further training, which contributes to its improved performance.
When comparing the use of SGD and Adam optimizers, it is evident that, in comparison to the improvements seen with SGD, the enhancements achieved with Adam are less pronounced. This observation can be attributed to Adam 's rapid convergence facilitated by its adaptive learning rates, which dynamically adjust based on parameter updates. Additionally, Adam 's momentum term aids in accelerating the convergence process. While Adam generally performs well across various datasets, the advantages of training data optimization are not as prominent in this context.

Additionally, we observed that, in contrast to the Statio dataset, higher numbers of epochs correspond to lower RMSE values, indicating better training effectiveness. However, for the cardio and parkin datasets, higher numbers of epochs are associated with higher RMSE values. This discrepancy may stem from incomplete attribute distribution analysis during the sampling and creation of training and testing datasets for these two datasets. It is possible that the distribution characteristics of the training and testing datasets do not align perfectly, leading to the observed trend. Despite this potential limitation, our research method still demonstrates superior performance within the same dataset.

6. CONCLUSION

DAEs and VAEs represent distinct types of AEs, each with unique training objectives and architectural characteristics. DAEs are primarily designed for data denoising and feature learning tasks, whereas VAEs are focused on probabilistic modeling and generative capabilities.

In our research, we primarily explore the implications of updating input data during AE training while avoiding extensive discussions on optimization algorithms and loss functions. To assess the impact of altering input data, we utilize DAE models and examine how various optimization mechanisms affect our proposed methods. We conduct comprehensive comparisons between SGD and the Adam optimization algorithm. We opt for DAE over VAE due to VAE's incorporation of a KL Divergence term, which remains unaffected by changes in input data.

Therefore, for our experiments, we selected the DAE architecture. We also considered two commonly used optimizers, namely SGD and Adam. We conduct comprehensive comparisons between SGD and the Adam optimization algorithm. Through our experiments, we have demonstrated that both LTR-SU and STR-IU approaches can significantly enhance the training quality of the base model. LTR-SU involves accumulating training results over stages and then performing a single replacement, while STR-IU replaces original training data directly based on each training iteration's results. Both approaches have proven to be effective.

In our experiments, we transformed three distinct datasets into synthetic datasets with varying levels of missing data (5%, 15%, 25%). The summarized results indicate that, while performance may not consistently excel across all training epoch settings, there is a noticeable overall improvement when updating input data, whether using SGD or Adam. Additionally, LTR-SU outperforms STR-RU, and models with DAE using SGD exhibit greater optimization compared to DAE using Adam.

While it is acknowledged that parameter selection often necessitates experimentation and fine-tuning, empirical evidence from our research indicates that parameter ranges suitable for addressing missing data do not exhibit substantial variations across distinct datasets and optimization techniques. This observation underscores the robustness and transferability of the proposed method. Consequently, the task of identifying appropriate parameter configurations does not pose a formidable challenge.

The significance of this finding lies in its potential to streamline the process of handling missing data effectively. It offers a standardized approach that can be applied with confidence across diverse datasets and analytical contexts. By simplifying parameter selection, this research contribution enhances the efficiency and reliability of missing data treatment, benefiting a wide range of data analysis and machine learning applications.

In our future research, we aim to explore other potentially valuable parameters. For instance, we plan to investigate whether excluding training data during the early stages of AE training, when convergence is still substantial, can improve subsequent replacement outcomes. Additionally, even within the same dataset, the training epoch at which the best performance is achieved may vary. Thus, identifying the optimal training epoch remains a subject of interest for future study

REFERENCES

1. BROWN, Marvin L.; KROS, John F. Data mining and the impact of missing data. Industrial management & data systems, 2003, 103.8: 611-621. Author 1 AB, Author 2 CD. Title of the chapter. In: Title of the Book, 2nd ed. Publisher name; 2023. pp. 102–144.

2. FAN, Jianqing; HAN, Fang; LIU, Han. Challenges of big data analysis. National science review, 2014, 1.2: 293-314.

3. GRAHAM, John W.; CUMSILLE, Patricio E.; ELEK-FISK, Elvira. Methods for handling missing data. Handbook of psychology, 2003, 87-114.

4. ZHANG, Zhongheng. Missing values in big data research: some basic skills. Annals of translational medicine, 2015, 3.21.

5. KWAK, Sang Kyu; KIM, Jong Hae. Statistical data preparation: management of missing values and outliers. Korean journal of anesthesiology, 2017, 70.4: 407.

6. BARALDI, Amanda N.; ENDERS, Craig K. An introduction to modern missing data analyses. Journal of school psychology, 2010, 48.1: 5-37.

7. RUBIN, Donald B. Inference and missing data. Biometrika, 1976, 63.3: 581-592.

8. LITTLE, Roderick JA; RUBIN, Donald B. Statistical analysis with missing data. John Wiley & Sons, 2019.

9. OSMAN, Muhammad S.; ABU-MAHFOUZ, Adnan M.; PAGE, Philip R. A survey on data imputation techniques: Water distribution system as a use case. IEEE Access, 2018, 6: 63279-63291.

10. MADLEY-DOWD, Paul, et al. The proportion of missing data should not be used to guide decisions on multiple imputation. Journal of clinical epidemiology, 2019, 110: 63-73.

11. SCHAFER, Joseph L. Multiple imputation: a primer. Statistical methods in medical research, 1999, 8.1: 3-15.

12. ALICE, M. Imputing missing data with R; MICE package 2015. 2018.

13. CHEEMA, Jehanzeb R. Some general guidelines for choosing missing data handling methods in educational research. Journal of Modern Applied Statistical Methods, 2014, 13.2: 3.

14. PEREIRA, Ricardo Cardoso, et al. Reviewing autoencoders for missing data imputation: Technical trends, applications and outcomes. Journal of Artificial Intelligence Research, 2020, 69: 1255-1285.

15. VINCENT, Pascal, et al. Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning. 2008. p. 1096-1103.

16. RYU, Seunghyoung; KIM, Minsoo; KIM, Hongseok. Denoising autoencoder-based missing value imputation for smart meters. IEEE Access, 2020, 8: 40656-40666.

17. LU, Haw-minn; PERRONE, Giancarlo; UNPINGCO, José. Multiple imputation with denoising autoencoder using metamorphic truth and imputation feedback. arXiv preprint arXiv:2002.08338, 2020.

18. ZHANG, Jianye; YIN, Peng. Multivariate time series missing data imputation using recurrent denoising autoencoder. In: 2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). IEEE, 2019. p. 760-764.

19. KINGMA, Diederik P.; WELLING, Max. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

20. REZENDE, Danilo Jimenez; MOHAMED, Shakir; WIERSTRA, Daan. Stochastic backpropagation and approximate inference in deep generative models. In: International conference on machine learning. PMLR, 2014. p. 1278-1286.

21. ROSKAMS-HIETER, Breeshey; WELLS, Jude; WADE, Sara. Leveraging variational autoencoders for multiple data imputation. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Cham: Springer Nature Switzerland, 2023. p. 491-506.

22. XIE, Ruimin, et al. Supervised variational autoencoders for soft sensor modeling with missing data. IEEE Transactions on Industrial Informatics, 2019, 16.4: 2820-2828.

23. MAKHZANI, Alireza, et al. Adversarial autoencoders. arXiv preprint arXiv:1511.05644, 2015.

24. TOLSTIKHIN, Ilya, et al. Wasserstein auto-encoders. arXiv preprint arXiv:1711.01558, 2017.

25. VINCENT, Pascal, et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research, 2010, 11.12.

26. NG, Andrew, et al. Sparse autoencoder. CS294A Lecture notes, 2011, 72.2011: 1-19.

27. GAO, Fuchang. Ae^{2}I: A Double Autoencoder for Imputation of Missing Values. arXiv preprint arXiv:2301.06633, 2023.

28. LIU, Xin; ZHANG, Zijun. A two-stage deep autoencoder-based missing data imputation method for wind farm SCADA data. IEEE Sensors Journal, 2021, 21.9: 10933-10945.

29. LU, Xugang, et al. Ensemble modeling of denoising autoencoder for speech spectrum restoration. In: Interspeech. 2014. p. 885-889.

30. WANG, Yanxia, et al. Missing data imputation with OLS-based autoencoder for intelligent manufacturing. IEEE Transactions on Industry Applications, 2019, 55.6: 7219-7229.

31. MIOK, Kristian, et al. Multiple imputation for biomedical data using monte carlo dropout autoencoders. In: 2019 E-Health and Bioengineering Conference (EHB). IEEE, 2019. p. 1-4.

32. YE, Yongchao; ZHANG, Shuyu; JAMES, J. Q. Traffic data imputation with ensemble convolutional Autoencoder. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). IEEE, 2021. p. 1340-1345.

33. FRIEDMAN, Jerome H. Greedy function approximation: a gradient boosting machine. Annals of statistics, 2001, 1189-1232.

34. HE, Zhiyuan, et al. Gradient boosting machine: a survey. arXiv preprint arXiv:1908.06951, 2019.

35. DONG, Manqing, et al. GrCAN: gradient boost convolutional autoencoder with neural decision forest. arXiv preprint arXiv:1806.08079, 2018.

36. SEYFIOĞLU, Mehmet Saygın; ÖZBAYOĞLU, Ahmet Murat; GÜRBÜZ, Sevgi Zubeyde. Deep convolutional autoencoder for radar-based classification of similar aided and unaided human activities. IEEE Transactions on Aerospace and Electronic Systems, 2018, 54.4: 1709-1723.

37. RUSHIN, Gabriel, et al. Horse race analysis in credit card fraud—deep learning, logistic regression, and Gradient Boosted Tree. In: 2017 systems and information engineering design symposium (SIEDS). IEEE, 2017. p. 117-121.

38. VO, Minh Thanh; NGUYEN, Trang; LE, Tuong. Robust head pose estimation using extreme gradient boosting machine on stacked autoencoders neural network. IEEE Access, 2019, 8: 3687-3694.

39. JANG, Jong-Hwan, et al. Unsupervised feature learning for electrocardiogram data using the convolutional variational autoencoder. PLoS One, 2021, 16.12: e0260612.

DEFINITIONS, ACRONYMS, ABBREVIATIONS

Here is the Definitions section. This is an optional section.
Term: Definition for the term