Information Sciences 595 (2022) 278–293

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Handling missing data through deep convolutional neural network

Hufsa Khan [a], Xizhao Wang [a,b], Han Liu [a,b,*]

[a] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
[b] Guangdong Key Laboratory of Intelligent Information Processing, Shenzhen University, Shenzhen 518060, China

ARTICLE INFO

Article history:
Received 20 November 2021
Received in revised form 16 January 2022
Accepted 24 February 2022
Available online 1 March 2022

Keywords:
Missing value
Data imputation
Fuzzy clustering
Convolutional neural network

ABSTRACT

The presence of missing data is a challenging issue in processing real-world datasets. It is necessary to improve the data quality by imputing the missing values so that effective learning from data can be achieved. Recently, deep learning has become the most powerful type of machine learning techniques, which can be used for discovering the hidden knowledge that exists in a large dataset to make accurate predictions. In this paper, we propose an imputation method that involves using a convolutional neural network to impute the missing values. The missing value of each instance is imputed essentially by using a trained kernel. The weights of the kernel are determined by learning from the given data that are arranged spatially in the data matrix. The kernel carries out a weighted sum of neighboring elements in an array for imputing the missing values. In addition, in the absence of the true values with which the missing values are expected to be replaced, a loss function is designed without the need to know the true value. Our method is evaluated on UCI datasets in comparison with state-of-the-art methods. The experimental results show that the proposed approach performs closely to or better than other methods.

1. Introduction

In the current era, in different domains, a vast amount of data is generated gradually and the rapid increase in the data size has shown the recognition of the significance of big data analysis. It is necessary to make sure collected data are trustable and valuable because poor quality data cannot be used to produce reliable models. In practice, the existence of missing values in the collected data set is a common and unavoidable issue that can lead to ambiguity in data analysis. The presence of missing data can occur in various domains, such as gene expression [18], traffic control [3], industrial informatics [22], image processing [48], and software project [42]. Data analysis made without addressing the above-mentioned issue may produce misleading results. Therefore, it is necessary to improve the data quality by effectively handling missing values.

The traditional methods of missing data handling can be summarized into two categories. The first one is deletion, which is designed to eliminate all those instances that have some features with missing values. The second approach is imputation, which aims to replace the missing values with some reasonable values. There have also been different machine learning based imputation methods, e.g., k-nearest neighbors (KNN) [9], recurrent neural networks (RNN) [9,17,34,16], and generative adversarial imputation networks (GAIN) [47,14,44,29]. Also, in [25], a machine learning based missing data imputation approach has recently been proposed in which the missing value of each sample is imputed by selecting a shorter interval

* Corresponding author.
E-mail addresses: hufsakhan@email.szu.edu.cn (H. Khan), xizhaowang@ieee.org (X. Wang), han.liu@szu.edu.cn (H. Liu).

https://doi.org/10.1016/j.ins.2022.02.051
0020-0255/

(i.e., the interval is actually selected by taking a sub-interval of the domain of the corresponding feature) and then taking the average of the feature values that fall in this interval. This approach shows good performance of imputation but it does not take into account the feature correlation which can be used to improve further the imputation accuracy and robustness.

For instance, in [11,32], the concept of feature correlation was used to enhance the imputation performance. It is interesting to know that the proposed approach was designed in [32] to impute the missing values just by exploiting the feature correlation and some other useful patterns that exist in data and the imputation effectiveness can be improved by considering only those features that have high correlation in the setting of deep learning. In our proposed approach, to improve the imputation accuracy, we propose to identify the correlation that exists in the data matrix and make use of the correlation information to achieve an improvement of the prediction performance. Therefore, in this paper, we pay a special attention on the data sets that hold the non-linearity and spatial relationship. Here the term "spatial relationship" refers to the spatial arrangement of the data in the data matrix. In the above-mentioned data arrangement, the kernel can be learned more effectively by exploiting the correlation information that exists in the given data. Otherwise, the kernel may not be constructed sufficiently well. Furthermore, the true value with which the missing value is expected to be replaced is likely to be unknown in real life. Therefore, it is necessary to explore how to design a loss function without the need to know the true value.

In addition, in the last few years, deep learning has been extensively used in different fields, including missing data imputation, which has led to a significant improvement of the imputation performance through using a large amount of training data. Therefore, due to the remarkable success of generative adversarial network (GAN), a great attention has been paid towards applying them for missing value imputation. In [47], a GAN based imputation method was proposed but it requires hyper-parameters setting to adjust the effect of the MSE loss term as well as the rate of discriminator hint vectors. In [28], generator and discriminator networks are used separately to learn the structure and distribution of the missing data. Although these GAN based methods show state-of-the-art performance, it should be noted that the presence of additional loss terms may often bias the generated samples toward the mode of the distribution being modeled. Also, these methods are often too complicated to be applied in practical setups.

The objective of this paper is to impute the missing value and measure how robust our imputation approach is. It is also highly desirable to measure how the imputation impacts the classification accuracy. In this paper, we propose a convolutional neural network imputation (CNNI) approach to handle the missing values to improve the data quality. In particular, the relevant issues are addressed by developing new models and strategies for effective recovery of missing data in the setting of deep learning. The proposed approach (CNNI) is considered not only useful for handling large datasets but also helpful for generating reasonable values for the imputation of missing values.

The main findings of this paper include:

- A new imputation approach based on a well-known deep convolutional neural network architecture is proposed.
- The proposed approach is capable of improving the effectiveness of missing value imputation through training a convolutional kernel by exploiting the useful information that exists in the given data, such as the spatial relationships and the non-linearity.
- It is validated experimentally and statistically that the proposed CNNI shows improved performance on 7 datasets from the UCI repository [19], as compared to other state-of-the-art imputation methods.

The rest of the paper is organized as follows: Section 2 briefly reviews the related work on machine learning based imputation methods. The proposed approach is presented in Section 3. Section 4 presents the experimental setup, results and empirical analysis of the proposed approach. Finally, the concluding remarks and future work are presented in Section 5.

2. Related work

This section reviews some representative works that focus on the missing data handling methods. Since the presence of missing data is a common problem in many data-driven applications, many studies have been conducted to handle this problem. In particular, the imputation of the missing values generally requires specific assumptions about the data distribution, i.e., any inappropriate assumptions can bias the estimations of the imputed values. However, traditional approaches of missing data imputation can be categorized into two types - statistical methods and machine learning based ones. In many fields, to handle the missing data problem, different imputation techniques have been proposed, e.g., a class center based approach [41], KNN [13], fuzzy clustering [25], a bagging based method [2], auto-encoder neural networks [27,15], similarity rules [37], multiple imputation by chained equations (MICE) [46] and random forests [39]. In [41], a class center based missing value imputation approach was proposed to achieve more effective replacement of missing values, and it was considered a modified form of mean imputation. In this approach, the distance between each of data points within a class and the center of the class is calculated, and the threshold for the later imputation is defined by using the other observed data.

In [8], various machine learning based predictive models, such as KNN, support vector machines (SVM) and decision trees, have been used to impute the missing values by formulating the imputation problem as an optimization one. K-nearest neighbor imputation (KNNI) is considered one of the most popular techniques due to its simplicity and effectiveness compared to other approaches. In [13], purity k nearest neighbors imputation (PkNNI) was proposed as an extension of the traditional KNNI method, which is based on purity training and imputation. In this method, the purity of a record is computed

by aggregating the votes of records that are selected as their nearest neighbours. However, for a huge dataset, NNI can be computationally expensive, since it needs to go through all the instances in order to impute each incomplete record in a data set. As a result, the performance of NNI is limited in the presence of a large amount of missing data [35]. Moreover, finding the nearest neighbours and identifying the exact distance function could be a difficult task [4]. Furthermore, a random forest based imputation approach was proposed in [39] and it was revealed that the performance and robustness of random forest imputation could be improved with increasing feature correlation.

In MIAEC [46], the chain approach was used to mine all evidence of missing values which are relevant and set up the further estimation of the missing values. In addition, Tian et al. [40] applied the fuzzy c-mean (FCM) clustering technique to group the observed data for training the gray theory based classifier to impute the missing values. An iterative approach named "incremental attribute regression imputation" was proposed in [43], which aims to build a sequence of regression models to impute the missing values iteratively. In addition, the class label of each sample is used as a predictor variable in the setting of incremental attribute regression imputation. The traditional missing data imputation approaches mainly focus on using different probabilistic models or regression methods to impute the missing values, and they only take limited information as inputs. Thus, they cannot perform very accurate imputation especially for data with a high ratio of missing values. However, in the case of very large datasets, machine learning based imputation approaches (e.g., KNN, MICE and fuzzy clustering) require much computation time, which makes these approaches less practical in the task of imputing a large amount of missing data.

In recent years, with the increase in the size of data, deep learning has shown great potential in different areas including biology [45], image reconstruction [50,6], biomedical imaging [12] and genomics studies [10]. Also, some deep learning methods have also been proposed to specifically solve the missing data imputation problem in various contexts leading to promising results [5,24]. For instance, in [36], the multi-layer perceptron (MLP) network was used to impute the missing values and was investigated in terms of the impacts of different learning rules and model parameters on the final performance. In [27,15], the auto-encoder architecture was used for missing value imputation, which involves learning how to reconstruct the original input value by minimizing the reconstruction error. In [47], the GAN network was used to impute the missing values. For GAN based approaches, it is quite costly to train the model due to the requirement of high performance computing resources. A hybrid approach based on neural networks and genetic algorithm is used in [1] to impute the missing values for medical IoT implementations. The method benefits from the effectiveness of deep learning in imputing the missing data and the usefulness of genetic algorithm in optimizing the weights of the neural network. In [27], an imputation model was proposed by using the auto-encoder based architecture that reduces the complexity of the data. In addition, in MIDA, [23] proposed a model based on a complete deep denoising auto-encoder that can handle different types of data, missingness patterns and missingness proportions.

In [20], a deep learning based missing value imputation approach was proposed for handling traffic data. The deep learning based approach is capable of discovering the correlations present in the data by a layer-wise pre-training and can improve the imputation performance by conducting a fine tuning afterward. In [9], a deep model was used for imputing the missing values present in time series data, where the model captured the long term temporal dependencies of time series observations and utilized the missing patterns for improving the prediction performance by incorporating masking and time interval in a deep model. Moreover, in [49], a novel local similarity imputation method was proposed to estimate the values to impute the missing data based on fast clustering and top knearest neighbors. In particular, a two-layer stacked auto-encoder combined with distinctive imputation was applied to locate the principal features of a dataset for the clustering part.

The above methods aim at predicting the values to impute the missing data one by one and directly by clustering, regression or neural networks. In Section 3, we present a new approach, which involves the use of CNN to extract deep features by exploiting the spatial correlation that exists in the arranged data matrix, and imputation is achieved by using a kernel that is produced by learning the non-linearity that exists in the given data. To the best of our knowledge, this is the first time a CNN approach has been applied to impute missing data.

3. Methodology

In this section, we will discuss the proposed method of missing value imputation, which is designed in the setting of CNN. The basic idea of the proposed approach is to impute the missing values essentially by using a trained CNN kernel, whereas the weights of the kernel are determined by learning the non-linearity and spatial property that exist in the given data. In the proposed approach, first of all, we find the correlation between the features so that the most correlated features can lie close to each other. In this way, the kernel can be learned more effectively by exploiting the correlation information that exists in the given data. Otherwise, the kernel may not be constructed sufficiently well. Secondly, the FCM algorithm is applied to organize instances into various clusters. Later on, the instances are sorted in an ascending order on the basis of the membership values to which these instances belong to a cluster. Finally, the proposed CNNI approach is applied to impute the missing values present in the well-organized data. This study focuses on imputing the missing values present in data that carry non-linearity and spatial relationships. The detailed description of the proposed approach is provided in the following subsection.

Algorithm 1: Training of the proposed CNNI method

Input : Data set X having m instances and n features.
Output : A trained CNNI model.
Step 1:
Begin:
Divide X into training data ($X^t_{obs} \in \mathbb{R}^{m\times n}$) and testing data ($X^{t'}_{obs} \in \mathbb{R}^{m'\times n}$).
Take $X^t_{obs} \in \mathbb{R}^{m\times n}$ and randomly delete some values to create missing data $X^t_{miss} \in \mathbb{R}^{m\times n}$.
FCM is used to divide $X^t_{miss} \in \mathbb{R}^{m\times n}$ into k clusters, and the membership value $\mu_{i,j}$ to which instance x_i belongs to the j-th cluster is calculated by using Eq. (2).
Find the correlation between the features of $X^t_{miss} \in \mathbb{R}^{m\times n}$, and arrange the instances in an ascending order on the basis of the membership values to which these instances belong to a cluster.
end
Step 2:
Begin:
$X^t_{miss} \in \mathbb{R}^{m*n}$ passed to the CNN model.
The CNN model imputes the missing values of $X^t_{miss} \in \mathbb{R}^{m\times n}$.
Imputed data passed to fully connected (FC) layers to predict the labels of instances by using a softmax function.
Calculate the loss by using the designed loss function (based on cross entropy).
Repeat the procedure until the loss gets minimized.
Repeat
end
Step 3:
Begin:
Weights of the kernel are optimized and the model gets trained to minimize the loss.
Return the trained CNNI model.
end

3.1. Problem formulation

We have a dataset which is represented in the form of a $m \times n$ matrix, where m is the number of instances and n is the number of features. The original data set X is splitted into two parts, i.e., the training data ($X^t_{obs} \in \mathbb{R}^{m\times n}$) and the testing data ($X^{t'}_{obs} \in \mathbb{R}^{m'\times n}$), as shown in Fig. 1.

After data partitioning to obtain the training and test sets, we artificially created the missing values in $X^t_{obs} \in \mathbb{R}^{m\times n}$ and $X^{t'}_{obs} \in \mathbb{R}^{m'\times n}$, where these missing values are initially filled with zeros. After creating the missingness, we get new versions of training data and testing data, i.e., $X^t_{miss} \in \mathbb{R}^{m\times n}$ and $X^{t'}_{miss} \in \mathbb{R}^{m'\times n}$, respectively. Furthermore, the FCM algorithm is applied on $X^t_{miss} \in \mathbb{R}^{m\times n}$ and $X^{t'}_{miss} \in \mathbb{R}^{m'\times n}$ to divide the data into different clusters (as shown in Fig. 1). Our objective is to impute the missing values by exploiting the spatial relationships and the non-linearity information that exist in the data matrix. Therefore, the instances are arranged in an ascending order on the basis of the membership values to which these instances belong to a cluster so that instances with closer membership values can lie close to each other and vice versa. The objective of sorting the data is to achieve more effective learning of the non-linearity pattern from given data.

The FCM algorithm can effectively organize n instances $X = \{x_1, \ldots, x_n\}$ into a number of fuzzy clusters. For a given set of data, FCM returns k cluster centroids $C = \{c_1, \ldots, c_k\}$ and a partition matrix W, respectively, where each element μ_{ij} of W represents the degree to which instance x_i belongs to the j-th cluster and here we set the number of clusters $k = 3$. The objective of FCM is to minimize an objective function as shown in Eq. (1):

$$J = \sum_{i=1}^{n}\sum_{j=1}^{k}(\mu_{ij})^m \|x_i - c_j\|^2, \tag{1}$$

where $\|x_i - c_j\|$ is the Euclidean distance between the i-th instance and the j-th cluster centroid and $m \in \mathbb{R}$ denotes the fuzzification coefficient, i.e., the fuzzifier m determines how fuzzy the cluster will be, while $m \geqslant 1$. However, a greater value of m

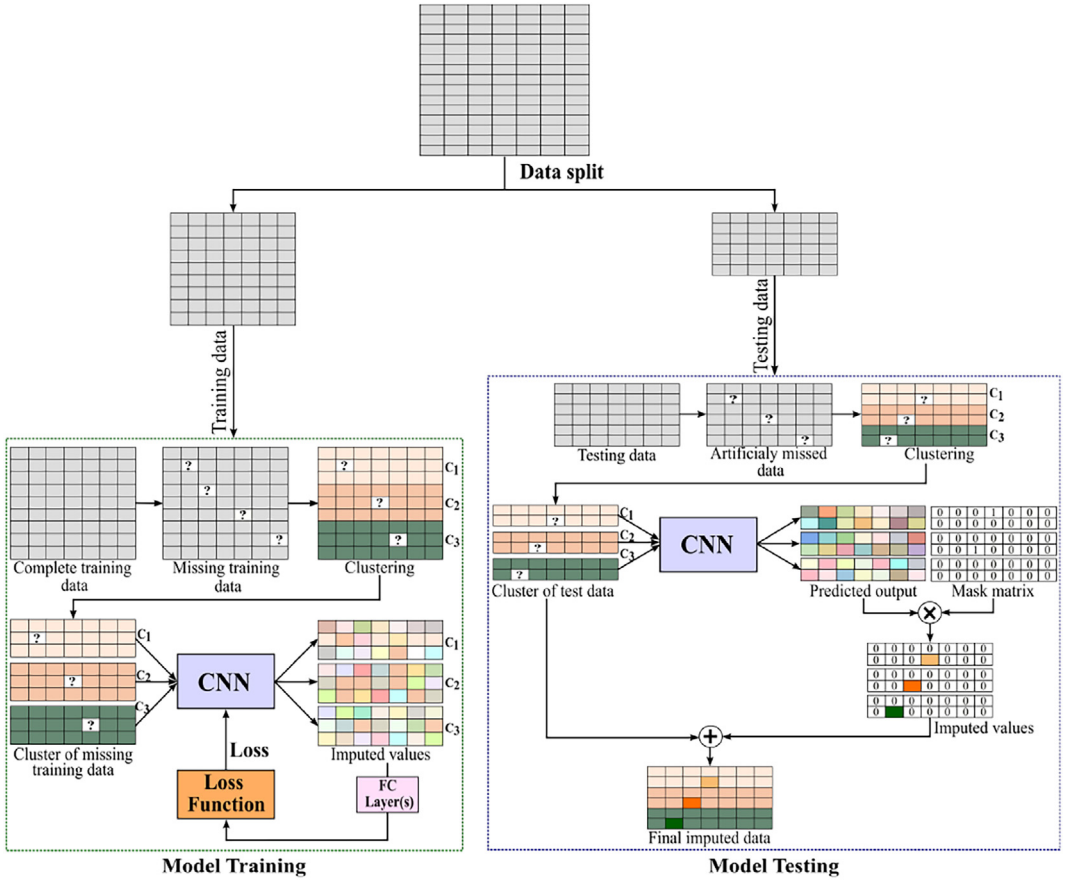


Fig. 1. Block diagram of the proposed CNNI imputation approach. One block represents model training and the other one shows the model testing.

results in a smaller membership value. In this study, we set its value m = 2. Furthermore, the membership degree to which instance x_i belongs to the j-th cluster is calculated by using Eq. (2).

$$\mu_{ij} = \frac{1}{\sum_{j\prime=1}^{k} \left(\frac{\|x_i - c_j\|}{\|x_i - c_{j\prime}\|} \right)^{\frac{2}{m-1}}}, \tag{2}$$

where c_j is the j-th cluster centroid and k is the number of clusters. Since identifying the centroid of a cluster is an iterative process, it continues until the termination condition is met.

$$c_j = \frac{\sum_{i=1}^{n} (\mu_{ij})^m x_i}{\sum_{i=1}^{n} (\mu_{ij})^m}, \forall j = 1, 2, \ldots, k, \tag{3}$$

In general, singularity is not caused in the denominator of Eq. (2) unless instance x_i is equal to the centroid of a cluster. According to Eq.(3), for the j-th cluster, instance x_i could be equal to centroid c_j when $\mu_{i,j}$=1 and $\forall\, i\prime \neq$ i: $\mu_{i\prime,j}$=0. However, this case is unlikely in real applications. Therefore, there is very little possibility to obtain zero in the denominator of Eq. (2) causing singularity.

We can see in Fig. 1 that one block represents the model training while the other one is used to show the model testing. In the training block, we have the clean training data ($X^t_{obs} \in \mathbb{R}^{m \times n}$) on which we artificially created the missing values, and these missing values are initially filled with zeros. After creating missingness, we get the polluted training data set which is

denoted as $X^{t}_{miss} \in \mathbb{R}^{m\times n}$, the Fuzzy c mean algorithm is applied on $X^{t}_{miss} \in \mathbb{R}^{m\times n}$. As can be seen in Fig. 1, where c_1, c_2, and c_3 represent the centroids of three clusters that contain instances with missing data, these clusters are passed to the CNN model to impute the missing values (see Algorithm 1).

On the other hand, in the testing phase, to create the missingness for the clean test data $X^{t'}_{obs} \in \mathbb{R}^{m'\times n}$, the same training procedure is repeated and we get the polluted test data $X^{t'}_{miss} \in \mathbb{R}^{m'\times n}$. After training the CNN model (see Algorithm 1), it is evaluated on $X^{t'}_{miss} \in \mathbb{R}^{m'\times n}$ to impute the missing values as shown in Fig. 1 and Algorithm 2. The missing values in $X^{t'}_{miss} \in \mathbb{R}^{m'\times n}$ can be imputed in the way illustrated in Eqs. (4) and (5).

$$x_{ij} = f_n(x^{t'}_{miss}), \tag{4}$$

$$f_n = f(\sum_i w_i x_i + b), \tag{5}$$

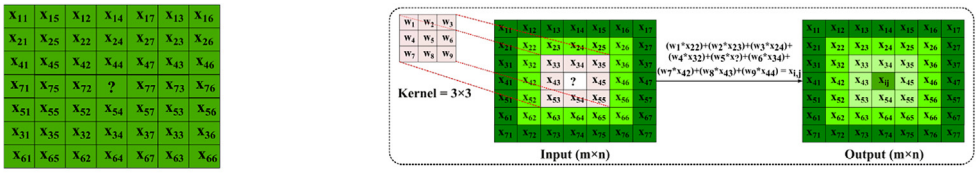
where $f_n(.)$ is the CNN network function that convolves over the data, x_i is the i-th instance, w_i is the weight vector of x_i, b is the bias term, and f is the activation function. We can notice that when the convolution operation is performed on the data it changes all the values of data. Similarly, in our case, after completing the imputation process, we get a new matrix of data in which all the values are changed due to the convolution operation. Therefore, to extract the imputed values, masking is applied on the newly generated data matrix as shown in Fig. 1. Later on, these masked values are merged with $X^{t'}_{miss} \in \mathbb{R}^{m'\times n}$ to obtain finally imputed data. At the last, after masking, we get a complete matrix with finally imputed values as shown in Fig. 1.

Furthermore, Fig. 2 represents an example of how a single value can be imputed by using a kernel for $m \times n$ input data. In the pre-processing stage, first, we identify the correlation that exists in the data matrix so that the most correlated feature values can lie close to each other. Fig. 2(a) represents the original data arrangement before identifying the spatial relationship between feature values, while we can see in Fig. 2(b) that different colors are used to show different extents to which different parts of these known values are spatially related to the missing value, after the data has been rearranged spatially through identifying the relationship between feature values. In the above rearrangement of data, the kernel can be learned more effectively by exploiting the correlation information that exists in the given data.

Let us have a kernel of size $M \times N$ (where M is the number of rows and N is the number of columns) and the value of a unit x_{ij} in the feature map is missing. The value of x_{ij} is calculated as a weighted sum of the inputs contained in a patch of size $M \times N$ as shown in Fig. 2(b). The weights of the kernel are learnable parameters that are estimated by exploiting the non-linearity information from the given data. Once the weights are optimized at the training stage, then they can be utilized for missing data imputation at the testing phase. Eq. (6) shows how a filter convolves over the data and imputes the missing value x_{ij}.

$$x_{ij} = (W * X)(i,j) = \sum_{m=1}^{M}\sum_{n=1}^{N} X_{i-m,j-n} W_m n, \tag{6}$$

where the symbol $*$ represents the convolution operation, m and n are used to represent the indexes of the row and column, respectively, i.e., m = 1 to M and n = 1 to N. In addition, i and j represent the indexes of the row and column in which the missing value is located. However, after completing the testing phase, we get a new matrix of data in which all the values are changed due to the CNN operation. The imputed data are taken out by applying the mask of missing parts on the final imputed data as shown in Fig. 1. Later on, these masked values are merged with the missing ones due to the case that missing values are replaced with newly imputed data and the rest of the data remain unchanged (see Fig. 1).



(a) Before spatial arrangement of data (b) After spatial arrangement of data

Fig. 2. An example of the process for missing value imputation. Fig. 2(a) represents the original data arrangement before identifying the spatial relationship between feature values. Fig. 2(b) shows the spatial arrangement of the data after identifying the spatial relationship between feature values, where different colors are used to show different extents to which different parts of these known values are spatially related to th.e missing value.

Algorithm 2: Imputation and testing procedure

Input : CNNI, test missing data ($X^{t'}_{miss} \in \mathbb{R}^{m' \times n}$) having m' instances and n features.
Output : Imputed data containing m' instances and n features.
Step 1:
Begin:
for i=1 to n do
Take testing data ($X^{t'}_{obs} \in \mathbb{R}^{m' \times n}$) and randomly delete some values to create test data with missing values ($X^{t'}_{miss} \in \mathbb{R}^{m' \times n}$).
Using FCM to organize $X^{t'}_{miss} \in \mathbb{R}^{m' \times n}$ into k clusters. The membership value $\mu_{i,j}$ to which instance x_i belongs to the j-th cluster is calculated by using Eq. (2).
Sort the instances in $X^{t'}_{miss} \in \mathbb{R}^{m' \times n}$ in an ascending order on the basis of the membership values to which these instances belong to a cluster.
end
Step 2:
Begin:
Impute $X^{t'}_{miss} \in \mathbb{R}^{m' \times n}$ by using Eq. (4)
Repeat the procedure until all missing values get imputed.
Repeat
end
Step 3:
Begin:
Apply a mask to get imputed values ($X^{t'}_{imputed} \in \mathbb{R}^{m' \times n}$) and merge them with $X^{t'}_{miss} \in \mathbb{R}^{m' \times n}$ to get the finally imputed data.
The designed loss function (based on cross-entropy) is used to calculate the loss.
Report Loss
end

3.2. Network architecture

The proposed method comprises of a particular family of neural network architectures known as CNN. Fig. 3 summarizes the architecture of our network and we can see the designed architecture consists of a dense convolutional neural network with a residual connection (CNN-DR). As the designed architecture carries a simple connectivity pattern, it aims to make sure that maximum information can flow between layers in the network. Therefore, we densely connected all layers directly to each other. However, to preserve the feed-forward nature, each layer obtains additional inputs from all its preceding layers and passes on its feature maps to all subsequent layers. In the dense connection, the concatenation is used to receive the collective knowledge from all preceding layers. A dense network is a type of convolutional neural networks, which utilizes the dense connection between the layers, where each layer is directly connected to every other layer in a feed-forward fashion. In contrast, in the traditional convolutional network, each layer is only connected to its subsequent layer, e.g., L layers have L sets of connections. The advantage of using this dense connection is that it solves the vanishing-gradient problem, strengthens the feature propagation, and encourages the feature re-usability property.

The dense and residual connections perform different operations, i.e., in a dense connection, we combine all the preceding feature maps through concatenation. On the other hand, in a residual connection, we combine feature maps through summation before they are passed into a specific layer. The advantage of having these two (residual and dense connection) networks is that we can use only one preceding feature map in a residual connection, whereas features obtained from all the preceding convolutional blocks can be used in a dense connection.

As can be seen in Fig. 3(a), the proposed CNN-DR architecture has six 2D Conv. layers with an increasing progressive number of filters that carry one residual connection. In Fig. 3(a), the solid line, which carries the input layer to the addition operator, is known as a residual connection. In particular, in the best-case scenario, including additional layers of the deep neural network can perform better than using the shallower counterpart network and also reduce the error by a significant margin. Therefore, we have included a residual block to achieve better performance than the one of the simple deep neural networks. The block is represented as shown in Eq. (7).

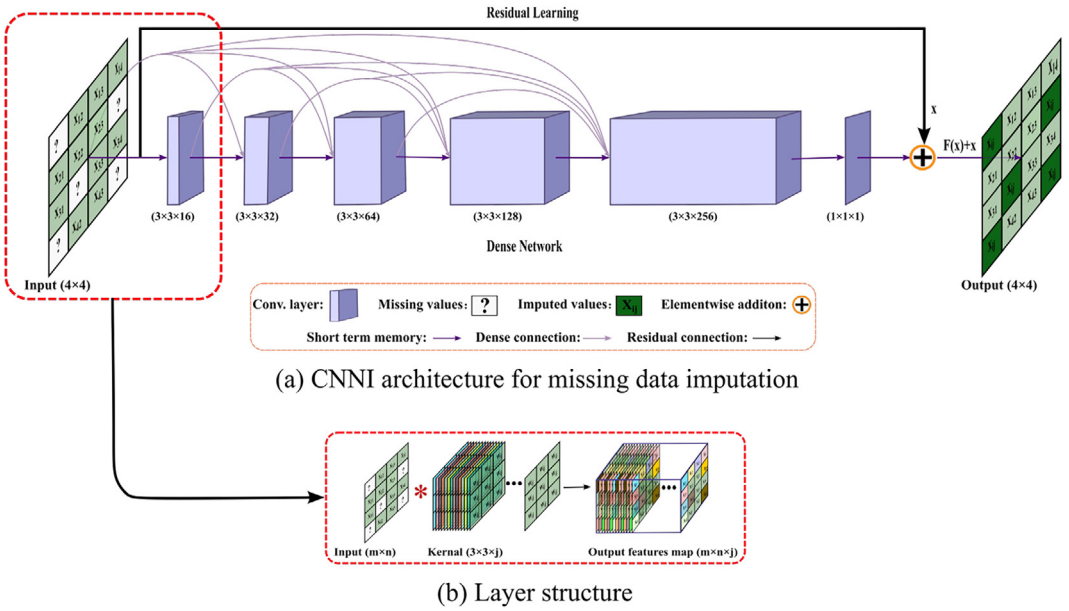


Fig. 3. CNNI architecture for missing data imputation is shown in Fig. 3(a). The layer structure of CNNI is shown in Fig. 3(b), which represents how each kernel (in the form of a 3×3 matrix) convolves on the input (in the form of a $m \times n$ matrix) and then a feature map (in the form of a $m \times n \times j$ tensor) is obtained as an output, where the symbol $*$ represents the convolution operation.

$$x_{i+1} = x_i + F(x_i, W_i), \tag{7}$$

$$H(x) = F(x) + x, \tag{8}$$

$$F(x) = relu(w_2 * (relu(w_1 x + b))), \tag{9}$$

where in Eq. (7), x_{i+1} and x_i are the output and input units of the network, F is a residual function, and W_i represents the parameters of the block, while Eq. (8) shows the feature transformation function where x is the input and $F(x)$ is the residual function. Similarly, Eq. (9) shows the residual function where relu is the activation function, w_1 and w_2 represent the weights in the first and second layers and b is used to represent the bias term. We use the residual block within the simple network because setting a skip connection in the residual network solves the vanishing gradient problem. In particular, this problem is resolved by allowing the alternate shortcut (via residual block) path for the gradient to flow through. The other way that this connection helps the model is to ensure that the higher layers will perform at least as good as the lower layers. With a residual block, the input can forward-propagate faster through the residual connection across layers.

We can see in Fig. 3(a) that the input data matrix has some randomly created missing values and there is a corresponding complete data matrix at the output. In order to avoid information loss and keep the data size consistent, padding is adopted to achieve that the output size of data remains the same as the input one. Moreover, in the designed architecture, the kernels have some weights which are the learnable parameters and the weight values form their surrounding which lies under a kernel. In addition, rectified linear unit (ReLU) is used as the activation function due to its effectiveness and popular applications in practice. The trainable parameters in a convolutional layer are estimated by Eq. (10).

$$(K_h * K_h * F_{i-1} + 1) * F_i, \tag{10}$$

where $K_h \times K_h$ is the convolutional kernel size, F_{i-1} represents the number of feature maps in the previous and current layers and + 1 shows a bias is added. Furthermore, Fig. 3(a) shows the imputation of a randomly missing value of a data sample going through a trained model. Once the model is trained, it is capable of imputing the missing values through the trained kernel.

3.3. Loss function

Our proposed CNNI approach is considered as an end-to-end joint learning scheme, where missing data imputation and classification are undertaken simultaneously. In other words, the proposed approach does not only focus on the imputation of missing values, considering that the main objective is to improve the classification performance and the imputed data are expected to be helpful in improving the effectiveness of classifier learning as pointed out in [21,25]. Also, due to the case that the true value with which the missing value is expected to be replaced is likely to be unknown in real life [30], a loss function

is designed without the need to know the true value, in order to measure how effectively the missing values are imputed in terms of promoting performance improvements in classification tasks.

In terms of the design of the proposed loss function, we adopt fully connected layers to undertake classification of the imputed data, where the softmax activation function is used to output a vector of class probabilities for each of the imputed instances. In this setting, the true labels ($X^L_{true} \in \mathbb{R}^{m\times n}$) and the predicted labels ($X_{pred} \in \mathbb{R}^{m\times n}$) are passed to a loss function to compute the loss, where the true labels are converted into one-hot vectors that can be directly compared with the predicted probability distributions. Specifically, the difference between $X^L_{true} \in \mathbb{R}^{m\times n}$ and $X^L_{pred} \in \mathbb{R}^{m\times n}$ is calculated by using categorical cross entropy shown in Eq. (11). To reduce the cross-entropy loss, the weights of the CNN kernal are changed towards reducing the difference between the predicted and true labels. This procedure is continued until the loss gets minimized (as shown in Fig. 1).

$$CrossEntropy = -\sum_{i=1}^{n} \overset{L}{X}_{true_{(i)}} log(\overset{L}{X}_{pred_{(i)}}), \tag{11}$$

where n is the total number of classes, $\overset{L}{X}_{true_{(i)}}$ is an indicator of whether the i-th class is the ground truth one, and $\overset{L}{X}_{pred_{(i)}}$ is the probability estimated for i-th class.

4. Experimental results and discussions

4.1. Experimental framework

In this section, we present the experimental performance of our proposed method and measure how the imputation performance superiority of the proposed approach is significant as compared to the performance of others. In particular, 7 datasets from the UCI machine learning repository are selected and their details are given in Section 4.2. Initially, random artificial missingness is created on the clean instances and these missing values are replaced with zeros. Later on, these missing values are imputed using the proposed CNNI approach and some other imputation techniques are selected for the sake of comparison, i.e., KNNI [7], MissForest [38], MICE [33], GAIN [47]. Finally, we confirm the stability of the proposed CNNI approach with the setting of different missing ratios (i.e., 10%, 20%, 30%, 40%, 50%, 60%, 70%). The performance evaluation of CNNI is undertaken by using five supervised learning algorithms (LR, LDA, KNN, NB and MLP) to train classifiers on imputed data. On each dataset, the experiment is executed 5 times in the setting of random data paritioning (80% for training and 20% for testing) and the average classification accuracy alongside the standard deviation is obtained, which is considered the final score of the imputation performance. Initially, 20% attribute values of all data points from the clean data are artificially removed. As a result, a dataset with missing values is generated.

4.2. Datasets

Table 1 gives the details of the selected datasets, such as the number of attributes, instances, and classes. We can see that none of the datasets has missing values, except the Ozone dataset. While in our case for evaluating the effectiveness of missing values imputation, the datasets should not contain missing values, such that a direct comparison with the performance obtained on clean data can be achieved. Therefore, during the data pre-processing stage, all the instances (records) that originally carry the missing values are removed. Instead, we artificially remove some feature values of these clean instances to create any missing values. Furthermore, we normalized each of the values into the range $[0, 1]$ to avoid the destructive effect of large scale values in the computation, i.e., normalization is taken to keep the domains of the features more consistent with each other.

4.3. Experimental setting

The proposed architecture based on CNN is shown in Fig. 3, where the network is composed of six 2D convolutional layers, which each has a successive number of kernels, i.e., 16, 32, 64, 128 and 256, respectively, with the size of 3×3, as shown

Table 1
The selected UCI datasets for experimental work.

Dataset	No.Rec.	No. Attr.	Classes	Missing data
Letter Recognition (LR)	20000	16	26	No
Optdigits (OD)	5620	64	10	No
Segmentation (Sg)	2310	19	7	No
Pendigits (PD)	10992	16	10	No
Parkinson(Pk)	5875	26	42	No
Ozone (Oz)	2536	73	2	Yes
Waveform (WF)	5000	21	3	No

in Fig. 3. The learning rate [26] is set to $1e^{-3}$, and the batch size is set to 4. The Adam optimizer is used with its default parameter setting to minimize the loss (considering the high efficiency of the optimizer) and a customized loss function (cross entropy) is used for the loss calculation at the output. In addition, to train the network, the number of epochs should be specified. Since having too few epochs often leads to under-fitting and a high computational cost and overfitting could result from having too many epochs, in our experiment, we set to involve 500 epochs. Interestingly, the convolution operation of CNN reduces the dimensionality of the input data and also there exists some boundary conditions for the elements that are close to the boundary line. Therefore, to address this problem, we take padding by adding zeros to the end of the input array and the stride size is set to 1. The advantage of adding this padding is that we ensure the output size to be the same as the input one. In addition, the ReLU activation function is used for each layer.

4.4. Performance evaluation criteria

In this paper, we selected classification accuracy as the metric to evaluate the proposed CNNI approach. The performance of imputation is evaluated by using five well known supervised learning algorithms, i.e., LR, LDA, KNN, NB and MLP. Classification accuracy is measured to specify the effectiveness of missing data imputation on the performance of classification. It is important to note that the objective of the experiments is to analyse how effectively missing value imputation helps improve the classification performance. Therefore, it is not a good choice to select the MAE and RMSE for performance evaluation, because a lower MAE or RMSE of imputation cannot directly indicates a higher score of classification accuracy. Due to this reason, we selected different algorithms of classifiers training to evaluate the performance in the absence of missing values and in the presence of missing values. We consider two aspects to evaluate the performance of the proposed CNNI approach. First, we want to analyse how good the performance of learning from the imputed data produced by CNNI is, in comparison with the performance of learning from the clean data. Second, we want to confirm the effectiveness of our method by comparing it with others. Therefore, for evaluation in these perspectives, the original data set is partitioned into 3 parts, namely, the clean training subset, the training subset with missing values and the testing set (the proportion of division is 60%, 20%, 20% respectively). We conduct the whole procedure of performance evaluation in two steps. In the first step, we use the aforementioned algorithms (LR, LDA, KNN, NB and MLP) to train classifiers on entirely clean data and perform the evaluation of each classifier on testing data (without missing values). In the second step, we use the same algorithms to train classifiers on the mixture of the clean instances and the instances with imputed values for evaluation using testing data (with some missing values). In Table 2, the results obtained through the two steps are shown.

4.5. Results and discussions

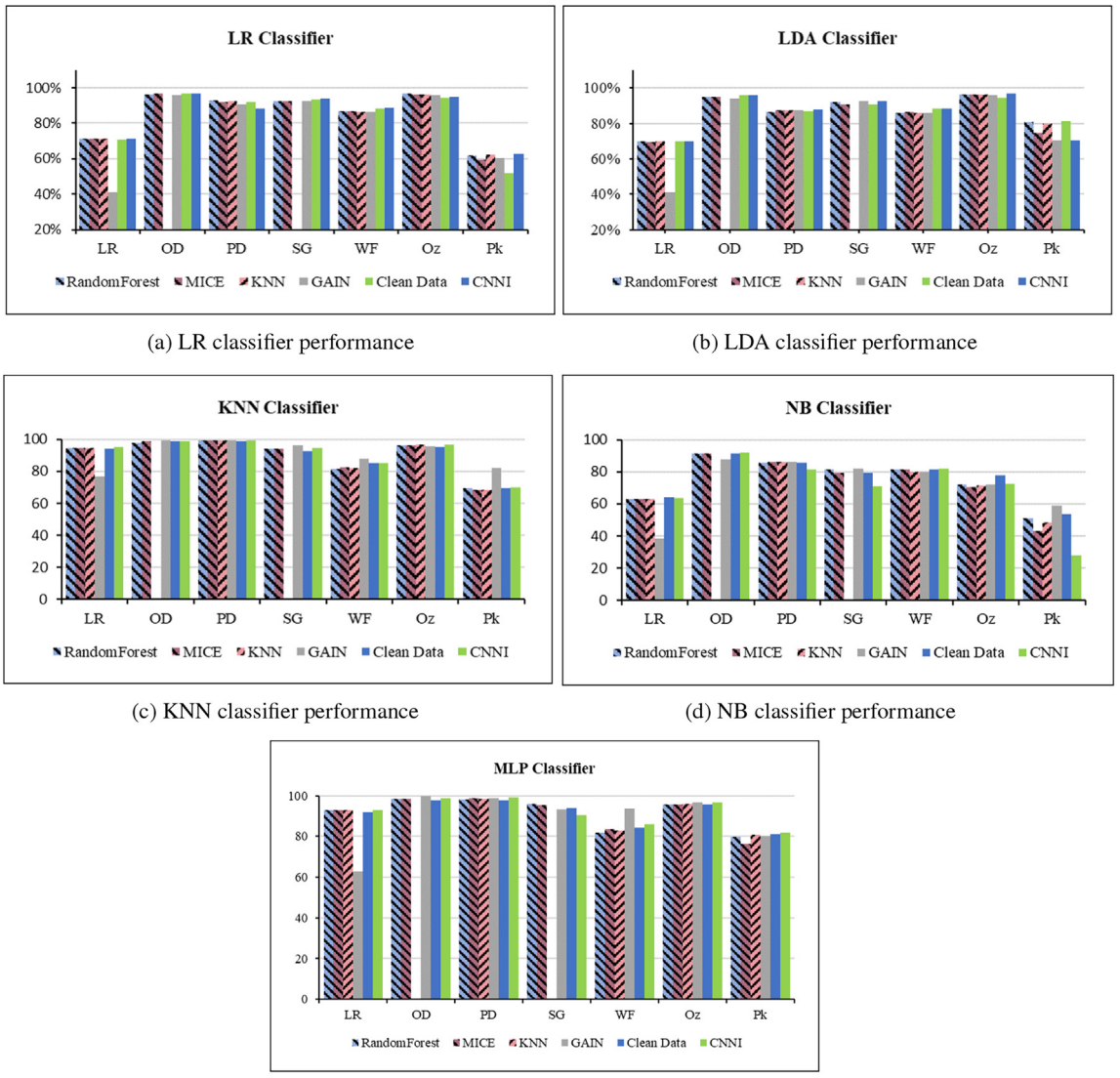
In this section, a detailed description of the comparison of the proposed method with other different imputation methods is provided. Table 2 shows the classification accuracy obtained using various methods over the randomly generated missing values. As it is shown in Table 2, there is a significant performance difference between the proposed method and each of the other ones. This difference can be visualized from Fig. 4, which is a graphical representation of Table 2. In Table 2, the best scores obtained on each dataset are shown in the bold format in a column and the second highest score is underlined. The experimental results obtained through varying missing ratios (i.e., 10%, 20%, 30%, 40%, 50%, 60%, 70%) are shown in Table 3 and are visualized in Fig. 5 and Fig. 6, which indicate that our imputation technique shows stable performance. The classification accuracy does not vary significantly with the change of the missing ratios in the data sets, and the phenomenon indicates that with the increase of the missing ratio in a data set the performance of the proposed approach will not become obviously worse. In addition, Table 3 shows the effectiveness and feasibility of the proposed CNNI approach while missing ratios vary. The results obtained on all the datasets show that the imputation based on the spatial relationship in the data matrix can result in an improvement of the classification performance.

Also, our proposed approach is compared with other state-of-the-art imputation methods. The performance of the proposed CNNI approach is compared with that of four different imputation methods, i.e., GAIN, RandomForest, KNNI, and MICE imputation approaches. In our experiments, we used the publicly available code in GitHub for implementing the GAIN approach and other imputation methods were implemented by using the Scikit-learn library [31], where the default parameter settings of various methods are used. We performed all our experiments on desktop computer intel(R) core(TM) i5-7400 CPU@3.00 GHz having 64-bit operating system.

We can see in Table 2 that the classification accuracy achieved on the data imputed by CNNI is better than the one achieved on clean data, which shows that the imputed values help improve the feature representation through using our proposed CNNI approach. Furthermore, it can be observed from the experimental results that in the majority of the cases for all the selected classifiers the performance produced by our CNNI approach is better than the one obtained by the other methods. However, the results indicate that for some classifiers trained on some datasets, the performance of our method remains marginally worse. For example, we can observe that the average accuracy of KNN and MLP classifiers trained on the optdigits and waveform data sets with missing value imputation by the GAIN method is a little bit higher than the one produced by CNNI. In addition, NB classifiers show better performance on pendigits and segmentation datasets, while using the GAIN method for missing value imputation. Random Forest shows better performance on pendigits and ozone datasets while using LR classifiers. Similarly, better performance can be obtained on the original (clean) versions of letter recognition and ozone datasets while using LR and NB classifiers, respectively, and the KNNI method shows better perfor-

Table 2
Classification accuracy obtained using various imputation approaches and supervised learning algorithms on 7 UCI datasets.

Dataset	Classifiers	RandomForest	MICE	KNNI	GAIN	Clean Data	CNNI
Pendigits (PD)	LR	**92.871 ± 0.077**	91.915 ± 0.469	92.700 ± 0.469	90.807 ± 0.550	91.859 ± 0.000	88.331 ± 0.307
	LDA	86.606 ± 0.027	87.174 ± 0.593	87.413 ± 0.703	87.234 ± 0.609	86.857 ± 0.000	**87.788 ± 0.262**
	KNN	99.488 ± 0.000	99.124 ± 0.253	99.215 ± 0.222	99.359 ± 0.205	98.908 ± 0.000	**99.617 ± 0.018**
	NB	85.730 ± 0.000	86.344 ± 0.821	86.048 ± 0.848	**86.431 ± 0.915**	85.629 ± 0.000	81.473 ± 0.068
	MLP	98.215 ± 0.151	98.811 ± 0.105	98.635 ± 0.450	98.975 ± 0.296	97.772 ± 0.000	**98.988 ± 0.231**
Optdigits (OD)	LR	96.450 ± 0.084	96.733 ± 0.388	–	95.647 ± 1.923	96.797 ± 0.000	**96.838 ± 0.222**
	LDA	95.063 ± 0.055	94.933 ± 1.138	-	93.867 ± 3.002	95.618 ± 0.000	**95.697 ± 0.267**
	KNN	98.019 ± 0.042	98.622 ± 0.499	–	**99.489 ± 0.62**5	98.833 ± 0.000	98.843 ± 0.000
	NB	91.290 ± 0.162	91.666 ± 0.810	–	87.815 ± 4.805	91.459 ± 0.000	**91.879 ± 0.297**
	MLP	98.489 ± 0.116	98.400 ± 0.268	–	**100.00 ± 0.000**	97.754 ± 0.000	98.884 ± 0.354
Letter Recognition (LR)	LR	71.085 ± 0.040	71.287 ± 0.684	71.206 ± 0.787	41.205 ± 0.870	70.825 ± 0.000	**71.465 ± 0.302**
	LDA	69.708 ± 0.014	69.556 ± 0.825	69.625 ± 0.970	41.232 ± 0.735	69.680 ± 0.000	**69.975 ± 0.188**
	KNN	94.684 ± 0.104	94.693 ± 0.825	94.631 ± 0.253	76.571 ± 1.474	94.225 ± 0.000	**94.930 ± 0.131**
	NB	63.246 ± 0.024	63.081 ± 0.465	63.250 ± 0.455	38.345 ± 0.919	**64.025 ± 0.000**	63.730 ± 0.452
	MLP	92.882 ± 0.311	92.950 ± 0.341	**92.987 ± 0.275**	62.923 ± 1.417	92.025 ± 0.000	92.850 ± 0.369
Segment (SG)	LR	92.571 ± 0.246	92.417 ± 1.363	–	92.566 ± 1.518	93.285 ± 0.000	**94.022 ± 0.407**
	LDA	92.046 ± 0.060	90.494 ± 0.992	–	92.372 ± 1.298	90.769 ± 0.000	**92.406 ± 0.291**
	KNN	94.274 ± 0.237	94.120 ± 0.788	–	**96.378 ± 1.326**	92.747 ± 0.000	94.659 ± 0.107
	NB	81.318 ± 0.000	79.230 ± 2.820	–	**81.988 ± 2.581**	79.560 ± 0.000	70.769 ± 2.103
	MLP	96.035 ± 0.759	95.439 ± 0.788	–	93.257 ± 1.579	94.065 ± 0.000	90.461 ± 2.017
Parkinson (Pk)	LR	61.930 ± 0.084	59.489 ± 0.918	62.383 ± 1.578	60.471 ± 1.729	52.068 ± 0.000	**62.978 ± 0.789**
	LDA	80.580 ± 0.063	74.680 ± 1.095	79.978 ± 1.533	70.222 ± 1.850	**81.191 ± 0.000**	70.127 ± 0.559
	KNN	69.481 ± 0.169	68.638 ± 0.827	68.638 ± 0.950	**81.947 ± 1.0385**	69.702 ± 0.000	69.868 ± 0.393
	NB	51.144 ± 0.045	43.212 ± 0.518	48.255 ± 1.266	**58.945 ± 2.466**	53.531 ± 0.000	28.187 ± 0.549
	MLP	79.639 ± 0.217	76.170 ± 1.926	80.808 ± 1.071	79.957 ± 0.993	81.276 ± 0.000	**81.863 ± 1.623**
Waveform (WF)	LR	86.595 ± 0.090	86.925 ± 0.430	86.325 ± 1.406	86.120 ± 0.577	88.400 ± 0.000	**88.920 ± 0.248**
	LDA	86.018 ± 0.070	86.350 ± 0.532	85.925 ± 1.130	85.780 ± 0.737	88.300 ± 0.000	**88.440 ± 0.372**
	KNN	81.668 ± 0.144	82.375 ± 0.079	81.850 ± 1.441	**87.689 ± 0.671**	85.400 ± 0.000	85.421 ± 0.000
	NB	81.224 ± 0.050	81.250 ± 0.684	80.000 ± 1.015	79.690 ± 1.073	81.500 ± 0.000	**81.851 ± 0.080**
	MLP	81.909 ± 0.746	83.550 ± 0.422	82.875 ± 0.932	**93.760 ± 1.475**	84.400 ± 0.000	85.860 ± 0.960
Ozone (Oz)	LR	**96.538 ± 0.145**	96.418 ± 0.344	96.213 ± 0.311	95.891 ± 2.537	94.594 ± 0.000	94.756 ± 0.366
	LDA	96.283 ± 0.000	96.148 ± 0.165	96.451 ± 0.211	95.891 ± 2.537	94.594 ± 0.000	**96.943 ± 0.216**
	KNN	96.113 ± 0.000	96.283 ± 0.302	96.666 ± 0.006	95.891 ± 2.537	95.405 ± 0.000	**96.880 ± 0.000**
	NB	71.959 ± 0.000	70.270 ± 1.282	71.556 ± 0.052	72.100 ± 9.997	**77.567 ± 0.000**	72.621 ± 0.556
	MLP	95.578 ± 0.469	95.743 ± 0.727	95.986 ± 0.631	96.761 ± 3.008	95.675 ± 0.000	**96.897 ± 0.132**



(a) LR classifier performance

(b) LDA classifier performance

(c) KNN classifier performance

(d) NB classifier performance

(e) MLP classifier performance

Fig. 4. The performance analysis of four imputation methods for five classifiers is shown. (**a**) LR classifier (**b**) LDA classifier (**c**) KNN classifier (**d**) NB classifier (**e**) MLP classifier. x-axis shows the datasets and y-axis shows the accuracy (%).

Table 3
Varying the ratio of missing values to achieve the stable accuracy.

Data Set	Classifiers	10%	20%	30%	40%	50%	60%,	70%
Optdigits	LR	96.902 ± 0.142	96.838 ± 0.222	95.998 ± 0.229	95.533 ± 0.330	95.338 ± 0.316	95.444 ± 0.294	95.373 ± 0.186
	LDA	95.755 ± 0.035	95.697 ± 0.267	94.933 ± 0.181	94.697 ± 0.090	94.644 ± 0.205	94.519 ± 0.261	94.608 ± 0.191
	KNN	98.843 ± 0.000	98.843 ± 0.000	98.833 ± 0.020	98.803 ± 0.100	98.841 ± 0.003	98.811 ± 0.000	98.801 ± 0.000
	MLP	98.624 ± 0.118	98.884 ± 0.354	98.669 ± 0.241	97.908 ± 0.342	97.758 ± 0.315	97.864 ± 0.368	97.918 ± 0.215
Waveform	LR	88.900 ± 0.451	88.920 ± 0.248	88.520 ± 0.519	87.020 ± 0.331	87.380 ± 0.318	87.300 ± 0.178	88.000 ± 0.792
	LDA	88.520 ± 0.097	88.440 ± 0.372	87.900 ± 0.360	87.520 ± 0.453	86.920 ± 0.530	86.840 ± 0.492	87.380 ± 0.430
	KNN	85.400 ± 0.000	85.421 ± 0.000	85.380 ± 0.000	85.400 ± 0.000	85.221 ± 0.000	85.400 ± 0.000	85.100 ± 0.000
	MLP	84.880 ± 0.722	85.860 ± 0.960	86.040 ± 0.233	85.700 ± 0.855	85.080 ± 0.563	85.500 ± 0.103	85.380 ± 0.847

mance on letter recognition dataset while using MLP to train the classifiers. The main reason for the good performance of these classifiers for some specific datasets is probably due to the suitability of some particular learning algorithms for the datasets. In other words, if the dataset fits well to the model structure of the classifier, then the classifier will perform better.

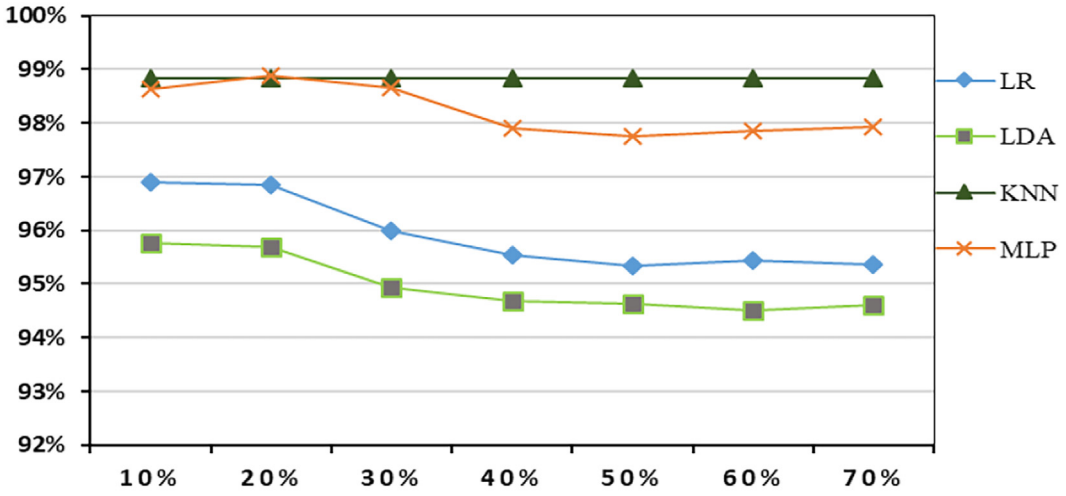


Fig. 5. Performance of CNNI at different missing ratios over the optdigits dataset. The x-axis and y-axis show the missing ratio and accuracy (%), respectively.

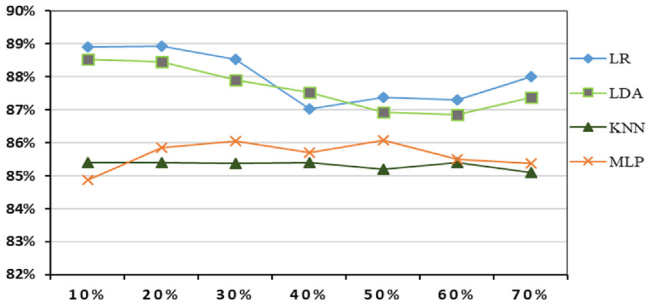


Fig. 6. Performance of CNNI at different missing ratio over the waveform dataset. The x-axis and y-axis show the missing ratio and accuracy (%), respectively.

According to the experimental results, our method shows significant superiority, where in most of the cases the performance of our method is close to or better than the one of the GAIN method. Also, the KNNI approach shows good performance but its computational efficiency is low due to the requirement of keeping track of all training instances for finding the neighbor nodes, whereas CNNI can more efficiently impute the data though using the convolutional kernel by considering very few parameters.

There are also some limitations of the proposed approach, e.g., in the proposed CNNI approach, those kernel weights are the learnable parameters which are determined by learning the non-linearity that exists in the given data and are updated iteratively. In such a case, if no spatial relationship can be discovered from the data, then it might be possible that the learning of the kernel goes through a wrong direction, leading to ineffective imputation. Since the paper is focusing only on the numerical datasets but many categorical datasets may also have missing values, it reveals the need to consider how to extend the proposed approach for handling missing values of categorical features.

4.6. Model analysis

An experiment on model analysis is conducted for the proposed CNNI approach and their results obtained using various parameter settings are shown in Table 4. We compare each specific parameter setting with the default one and show the effect of parameters tuning on the performance. Therefore, based on the results shown in Table 4, we finally selected the default parameters for the proposed CNNI approach as shown in Section 4.3. The model analysis is specifically conducted on the optdigits dataset, and the classification accuracy is measured by using two learning algorithms, i.e., MLP and LR, with different parameter settings.

Table 4
Model analysis. This table reports accuracy (%) in each condition for optdigits dataset.

Setting	LR	MLP
Our default setting	96.838 ± 0.222	98.884 ± 0.354
CNN-DR →Plain 2D ConvNet	95.355 ± 0.344	90.782 ± 1.314
CNN-DR → Residual connection	95.362 ± 0.313	97.998 ± 0.255
CNN-DR → Dense connection	96.022 ± 0.451	98.352 ± 0.322
6 Conv. 2D layer→5 Conv. 2D layer	96.492 ± 0.745	98.659 ± 0.156
6 Conv. 2D layer→7 Conv. 2D layer	96.805 ± 0.603	98.885 ± 0.711
Epoch 500→300	95.409 ± 0.153	91.270 ± 0.699
Epoch 500→600	96.851 ± 0.337	96.679 ± 0.557
Kernel size $3 \times 3 \rightarrow 5 \times 5$	96.800 ± 0.153	98.199 ± 0.699
ReLU →Leaky ReLU	95.960 ± 0.306	98.590 ± 0.670
ReLU →Tanh	95.213 ± 0.220	87.651 ± 0.661
k = 3 →k = 4	96.851 ± 0.339	98.881 ± 0.318
k = 3 →k = 5	96.813 ± 0.121	98.879 ± 0.361

In Table 4, the first row represents the accuracy (%) obtained using the default parameter setting of CNNI performing on optdigits dataset for LR and MLP classifiers. In Table 4, we can see how the setting of a sequence of controlled experiments can affect the evaluation process to show the impacts of different elements on the mentioned dataset.

Table 4 (the second, third, and fourth rows) represents the results obtained by replacing our default architecture (CNN-DR) with plain 2D Conv. net, Residual connection net, and CNN Dense net, respectively. In particular, the plain network consists of seven 2D Conv. layers with the Leaky ReLU activation function, the residual connection network carries the simple 2D Conv. net with a residual connection from the input layer to the output layer, and in the CNN Dense architecture, each layer is connected to its successive layer and ReLU is used as an activation function. It can be seen that the model trained with default parameter settings shows better performance as compared to those models trained using other settings of parameters.

In Table 4, the fifth and sixth rows represent the performance of replacing the 2D Conv. layer where by default we set to have 6 convolutional layers. However, when we increase the number of 2D Conv. layers from 6 to 8 in our default residual progressive CNN model, it can be observed that there is no significant effect on the accuracy score, which indicates that adding an extra layer does not significantly improve the accuracy for the selected data set, i.e., 6 layers are enough to achieve good performance.

The seventh and eighth rows of Table 4 show the effect of the number of epochs on the optdigits dataset. In the proposed CNNI, by default, we set 500 epochs, but we evaluated the proposed model with a tuned number of epochs. The results indicate that selecting 500 epochs is enough for the proposed CNNI as the increase of the number of epochs does not significantly impact on the performance but the influence is significant with the decrease of the number of epochs.

The ninth row of Table 4 shows the effect of the filter size on the performance. By default, we selected 3×3 as the kernel size, but we evaluated the proposed CNNI model by setting different kernel sizes. The results indicate that the increase of the kernel size makes the performance worse than that obtained by the default setting. Specifically, the results show that the increase of the kernel size results in an increased impact on determining the neighboring weights of a specific weight in the kernel and making the imputed value less confident. Therefore, a kernel with a smaller size is suitable for the experiment as compared to the one with a larger size. Also, we change the activation function from ReLU to Leaky ReLU or Tanh and the results show that better performance can be obtained using ReLU.

The last two rows of Table 4 show the effect of the number k of clusters on the performance. By default, we selected $k = 3$, but we evaluated the proposed CNNI approach by setting different values of k (i.e., $k = 4,5$). It can be seen in Table 4 that the accuracy score across all the defined values of k are close to each other, which indicates that the proposed approach is not so much sensitive to the number of clusters. In the case of having a larger number of clusters, there is no significant effect on the accuracy, but the computational complexity and execution time will be increased, so we set $k = 3$ for our experiments.

5. Conclusion

In this paper, we have proposed a CNN based missing data imputation approach, which aims to transform incomplete data into complete one. In the proposed approach, a CNN kernel is constructed by learning the spatial relationship and non-linearity that exist in the given data for imputing the missing values. Furthermore, the fuzzy c-mean clustering technique is applied to partition the data into different clusters, on the basis of their membership values, such that data can be organized to capture the strong correlation between feature values. The proposed CNNI approach has been compared with other state-of-the-art methods using different real-world datasets from the UCI respository and the results show the effectiveness of CNNI in the imputation of missing values of numeric features. Furthermore, the proposed approach shows stable performance while the missing ratio is varied from 10% to 70%. The proposed CNNI approach comes up with a strong alternative to traditional methods for imputing missing values in large datasets. In the future, we will explore the strategies of missing value imputation on more complex datasets such as multi-modal data.

CRediT authorship contribution statement

Hufsa Khan: Methodology, Validation, Writing - original draft. **Xizhao Wang:** Supervision, Writing - review & editing, Funding acquisition. **Han Liu:** Methodology, Supervision, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China, Guangdong province (No.2018A 0303130026) and National Natural Science Foundation of China (Grants 61976141 and 61732011).

References

[1] N. Al-Milli, W. Almobaideen, Hybrid neural network to impute missing data for IoT applications, in: 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), IEEE, 121–125, 2019..
[2] A. Andiojaya, H. Demirhan, A bagging algorithm for the imputation of missing values in time series, Expert Syst. Appl. 129 (2019) 10–26.
[3] B. Bae, H. Kim, H. Lim, Y. Liu, L.D. Han, P.B. Freeze, Missing data imputation for traffic flow speed using spatio-temporal cokriging, Transp. Res. Part C: Emerg. Technol. 88 (2018) 124–139.
[4] G.E. Batista, M.C. Monard, An analysis of four missing data treatment methods for supervised learning, Appl. Artif. Intell. 17 (5–6) (2003) 519–533.
[5] B.K. Beaulieu-Jones, J.H. Moore, P.R.O.-A.A.C.T. CONSORTIUM, Missing data imputation in the electronic health record using deeply learned autoencoders, in: Pacific Symposium on Biocomputing 2017, World Scientific, 207–218, 2017..
[6] C. Belthangady, L.A. Royer, Applications, promises, and pitfalls of deep learning for fluorescence image reconstruction, Nature Methods 16 (12) (2019) 1215–1225.
[7] L. Beretta, A. Santaniello, Nearest neighbor imputation algorithms: a critical evaluation, BMC Med. Inform. Decision Making 16 (3) (2016) 197–208.
[8] D. Bertsimas, C. Pawlowski, Y.D. Zhuo, From predictive methods to missing data imputation: an optimization approach, J. Mach. Learn. Res. 18 (1) (2017) 7133–7171.
[9] Z. Che, S. Purushotham, K. Cho, D. Sontag, Y. Liu, Recurrent neural networks for multivariate time series with missing values, Sci. Rep. 8 (1) (2018) 1–12.
[10] C.L. Chen, A. Mahjoubfar, L.-C. Tai, I.K. Blaby, A. Huang, K.R. Niazi, B. Jalali, Deep learning in label-free cell classification, Sci. Rep. 6 (1) (2016) 1–16.
[11] X. Chen, Z. Wei, Z. Li, J. Liang, Y. Cai, B. Zhang, Ensemble correlation-based low-rank matrix completion with applications to traffic data imputation, Knowledge-Based Syst. 132 (2017) 249–262.
[12] Y. Chen, Y. Li, R. Narayan, A. Subramanian, X. Xie, Gene expression inference with deep learning, Bioinformatics 32 (12) (2016) 1832–1839.
[13] C.-H. Cheng, C.-P. Chan, Y.-J. Sheu, A novel purity-based k nearest neighbors imputation method and its application in financial distress prediction, Eng. Appl. Artif. Intell. 81 (2019) 283–299.
[14] H.-S. Chiang, M.-Y. Chen, Y.-J. Huang, Wavelet-based EEG processing for epilepsy detection using fuzzy entropy and associative petri net, IEEE Access 7 (2019) 103255–103262.
[15] S.J. Choudhury, N.R. Pal, Imputation of missing data with neural networks for classification, Knowledge-Based Syst. 182 (2019) 104838.
[16] J. de Jesús Rubio, E. Lughofer, J. Pieper, P. Cruz, D.I. Martinez, G. Ochoa, M.A. Islas, E. Garcia, Adapting H-infinity controller for the desired reference tracking of the sphere position in the maglev process, Inform. Sci. 569 (2021) 669–686.
[17] J.J. de Rubio, Stability analysis of the modified Levenberg-Marquardt algorithm for the artificial neural network training, IEEE Trans. Neural Networks Learn. Syst..
[18] M.C. De Souto, P.A. Jaskowiak, I.G. Costa, Impact of missing data imputation methods on gene expression clustering and classification, BMC Bioinform. 16 (1) (2015) 1–9.
[19] D. Dua, C. Graff, UCI Machine Learning Repository, URL: http://archive.ics.uci.edu/ml, 2017.
[20] Y. Duan, Y. Lv, W. Kang, Y. Zhao, A deep learning based approach for traffic data imputation, in: 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), IEEE, 912–917, 2014..
[21] U. Garciarena, R. Santana, An extensive analysis of the interaction between missing data types, imputation methods, and supervised classifiers, Expert Syst. Appl. 89 (2017) 52–65.
[22] Z. Ge, Z. Song, S.X. Ding, B. Huang, Data mining and analytics in the process industry: The role of machine learning, IEEE Access 5 (2017) 20590–20616.
[23] L. Gondara, K. Wang, Mida: Multiple imputation using denoising autoencoders, in: Pacific-Asia conference on knowledge discovery and data mining, Springer, 260–272, 2018..
[24] N. Jaques, S. Taylor, A. Sano, R. Picard, Multimodal autoencoder: A deep learning approach to filling in missing sensor data and enabling better mood prediction, in: 2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII), IEEE, 202–208, 2017..
[25] H. Khan, X. Wang, H. Liu, Missing value imputation through shorter interval selection driven by Fuzzy C-Means clustering, Comput. Electr. Eng. 93 (2021) 107230.
[26] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980..
[27] X. Lai, X. Wu, L. Zhang, W. Lu, C. Zhong, Imputations of missing values using a tracking-removed autoencoder trained with incomplete data, Neurocomputing 366 (2019) 54–65.
[28] S.C.-X. Li, B. Jiang, B. Marlin, Misgan: Learning from incomplete data with generative adversarial networks, arXiv preprint arXiv:1902.09599..
[29] A. López-González, J.M. Campaña, E.H. Martínez, P.P. Contro, Multi robot distance based formation using Parallel Genetic Algorithm, Appl. Soft Comput. 86 (2020) 105929.
[30] N. McCombe, X. Ding, G. Prasad, D.P. Finn, S. Todd, P.L. McClean, K. Wong-Lin, Predicting feature imputability in the absence of ground truth, arXiv preprint arXiv:2007.07052..
[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine Learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[32] M.G. Rahman, M.Z. Islam, Fimus: A framework for imputing missing values using co-appearance, correlation and similarity analysis, Knowledge-Based Syst. 56 (2014) 311–327.
[33] A. Rubinsteyn, S. Feldman, fancyimpute: An Imputation Library for Python, URL: https://github.com/iskandr/fancyimpute, 2016..
[34] J. d. J. Rubio, Y. Pan, J. Pieper, M.-Y. Chen, J.H. Sossa Azuela, Advances in Robots Trajectories Learning via Fast Neural Networks, Front. Neurorobot. 15 (2021) 29..

[35] A.M. Sefidian, N. Daneshpour, Missing value imputation using a novel grey based fuzzy c-means, mutual information based feature selection, and regression model, Expert Syst. Appl. 115 (2019) 68–94.
[36] E.-L. Silva-Ramírez, R. Pino-Mejías, M. López-Coello, Single imputation with multilayer perceptron and multiple imputation combining multilayer perceptron and k-nearest neighbours for monotone patterns, Appl. Soft Comput. 29 (2015) 65–74.
[37] S. Song, Y. Sun, A. Zhang, L. Chen, J. Wang, Enriching data imputation under similarity rule constraints, IEEE Trans. Knowl. Data Eng. 32 (2) (2018) 275–287.
[38] D.J. Stekhoven, P. Bühlmann, MissForest–non-parametric missing value imputation for mixed-type data, Bioinformatics 28 (1) (2012) 112–118.
[39] F. Tang, H. Ishwaran, Random forest missing data algorithms, Statistical Analysis and Data Mining: The ASA, Data Sci. J. 10 (6) (2017) 363–377.
[40] J. Tian, B. Yu, D. Yu, S. Ma, Missing data analyses: a hybrid multiple imputation algorithm using gray system theory and entropy based on clustering, Appl. Intell. 40 (2) (2014) 376–388.
[41] C.-F. Tsai, M.-L. Li, W.-C. Lin, A class center based approach for missing value imputation, Knowledge-Based Syst. 151 (2018) 124–135.
[42] J. Van Hulse, T.M. Khoshgoftaar, Incomplete-case nearest neighbor imputation in software measurement data, Inform. Sci. 259 (2014) 596–610.
[43] B. van Stein, W. Kowalczyk, An incremental algorithm for repairing training sets with missing values, in: International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Springer, 175–186, 2016..
[44] D.M. Vargas, Superpixels extraction by an Intuitionistic fuzzy clustering algorithm, J. Appl. Res. Technol. 19 (2) (2021) 140–152.
[45] S. Webb, Deep learning for biology, Nature 554 (7693)..
[46] X. Xu, W. Chong, S. Li, A. Arabo, J. Xiao, MIAEC: Missing data imputation based on the evidence chain, IEEE Access 6 (2018) 12983–12992.
[47] J. Yoon, J. Jordon, M. Schaar, Gain: Missing data imputation using generative adversarial nets, in: International Conference on Machine Learning, PMLR, 5689–5698, 2018..
[48] Q. Zhang, Q. Yuan, C. Zeng, X. Li, Y. Wei, Missing data reconstruction in remote sensing image with a unified spatial–temporal–spectral deep convolutional neural network, IEEE Trans. Geosci. Remote Sens. 56 (8) (2018) 4274–4288.
[49] L. Zhao, Z. Chen, Z. Yang, Y. Hu, M.S. Obaidat, Local similarity imputation based on fast clustering for incomplete data in cyber-physical systems, IEEE Syst. J. 12 (2) (2016) 1610–1620.
[50] B. Zhu, J.Z. Liu, S.F. Cauley, B.R. Rosen, M.S. Rosen, Image reconstruction by domain-transform manifold learning, Nature 555 (7697) (2018) 487–492.