Sánchez-Morales, A., Sancho-Gómez, J. L., Martínez-García, J. A. & Figueiras-Vidal, A. R. (2020). Improving deep learning performance with missing values via deletion and compensation. In: *Neural Computing and Applications (Special Issue on Green and Human Information Technology 2019//International Work conference on the Interplay between Natural and Artificial Computation (IWINAC) 2015, 32*(17), Sept. 2020, pp. 13233–13244.

DOI: https://doi.org/10.1007/s00521-019-04013-2

Improving deep learning performance with missing values via deletion and compensation

Adrián Sánchez-Morales[1] · José-Luis Sancho-Gómez[1] · Juan-Antonio Martínez-García[1] · Aníbal R. Figueiras-Vidal[2]

Abstract
Missing values in a dataset is one of the most common difficulties in real applications. Many different techniques based on machine learning have been proposed in the literature to face this problem. In this work, the great representation capability of the stacked denoising auto-encoders is used to obtain a new method of imputating missing values based on two ideas: deletion and compensation. This method improves imputation performance by artificially deleting values in the input features and using them as targets in the training process. Nevertheless, although the deletion of samples is demonstrated to be really efficient, it may cause an imbalance between the distributions of the training and the test sets. In order to solve this issue, a compensation mechanism is proposed based on a slight modification of the error function to be optimized. Experiments over several datasets show that the deletion and compensation not only involve improvements in imputation but also in classification in comparison with other classical techniques.

Keywords Missing values · Imputation · Classification · Deep learning

1 Introduction

In recent years, data processing is being an extensively exploited field. Great efforts are being carried out in order to develop mechanisms to obtain useful information from data, a task which is getting harder due to the amount of information which is daily produced. Through literature, researchers have shown the wide range of drawbacks which can appear when handling real datasets. One of the most studied is the presence of missing values that arise in almost every real-world application [1–3].

Different effects of incomplete datasets in classification performance appear in accordance with the way of dealing with missing values. A first approach is to train classifiers only with complete samples. In this case, useful information is discarded and the classification of new incomplete instances is not possible. There are also embedded procedures which can directly deal with unknown input values without imputation, such as decision trees [4] and fuzzy neural networks [5, 6]. However, the more extended procedures are those that impute the missing values. This is because most decision-making tools cannot be directly used for incomplete data classification. Besides, imputation also extracts additional information (imputed values) which can be used to enhance the classification performance.

The most popular imputation procedures are based on statistical models for the class distributions, such as the well-known Gaussian mixture model (GMM) with an expectation–maximization (EM) formulation for

✉ Adrián Sánchez-Morales
adrian.sanchezm@edu.upct.es

José-Luis Sancho-Gómez
josel.sancho@upct.es

Juan-Antonio Martínez-García
juan.antonio.mtnez@gmail.com

Aníbal R. Figueiras-Vidal
arfv@tsc.uc3m.es

[1] Departamento de Tecnologías de la Información y las Comunicaciones, Universidad Politécnica de Cartagena, Plaza del Hospital, 1. Edificio Cuartel de Antigones, 30202 Cartagena, Murcia, Spain

[2] Departamento de Teoría de la Señal y Comunicaciones, Universidad Carlos III de Madrid, Avda Universidad, 30, 28911 Leganés, Madrid, Spain

maximum-likelihood (ML) estimation of both the mixture parameters and the missing values [7–9]. Other reasonable imputation schemes take into account the proximity or similarity of the incomplete instances to other complete examples, such as the *K*-nearest neighbors (KNN) [10–12] and the self-organizing maps (SOMs) [13–15] imputation methods. Auxiliary learning machines, such as multilayer perceptrons (MLPs) [16, 17] or support vector machines (SVMs) [18], can also be used to estimate missing values. Nevertheless, although all these approaches effectively use the available information, they do not directly consider that, in many cases, the final objective is just to obtain a good classification performance. In this context, some works like [10, 19] propose methods for imputation focused on the improvement of the classification.

In 2006, deep neural networks (DNN) gained popularity due to works like [20, 21]. They show that the difficulty of training a simple DNN (for example, an MLP with several hidden layers) could be overcome by training these networks layer by layer in an unsupervised manner, followed by a supervised learning and possibly a final tuning of the stacked network. By learning multiple levels of representations that depict different levels of abstraction of the data, deep learning algorithms learn a hierarchy of concepts [22, 23]. There are different types of deep learning architectures such as convolutional neural networks (CNN), convolutional deep belief networks (CDBN), deep neural networks (DNN), deep belief networks (DBN), stacked (denoising) auto-encoders (SAE/SDAE) and deep/stacked restricted Boltzmann machines (DBM).

There is not in the literature a large variety of imputation techniques based on deep learning. In [24], an imputation method based on AEs is presented. There, the missing and observed values are leveraged by optimizing a modified version of cross-entropy. Moreover, [25] presents a multiple imputation technique based on sparse SDAEs which is able to exploit data with different distributions and missing rates. Simultaneously, in [26] an imputation method with artificial values deletion is presented.

In this work, two new procedures are introduced during the training phase of an imputation technique which is based on SDAE. These are called deletion and compensation, and we will demonstrate how they improve the final performance. This paper is an extension of work [26] where the concept of deletion was introduced. Here, this concept is further developed and the efficiency of the compensation between the imbalanced training and test sets is studied. Although it is possible to use deletion without compensation [26], compensation supposes a clear improvement element as we prove now for the very first time. Therefore, the pair "deletion–compensation" is the novelty and the compensation has proven to be a key to improve the final performance of the imputation results.

Furthermore, the advantages obtained through these procedures in the imputation technique are used to introduce a new method of improving classification with a deep network.

This paper is organized as follows. The next section presents the proposed method, making a distinction between the use of deep networks to impute and to classify inputs with missing values. Deletion and compensation concepts are presented and applied in both cases. In the Experiments section, these techniques are used to solve several problems, reviewing the pre-imputation techniques that are used in this work. Finally, the paper is closed with the conclusions and possible future research directions.

2 Proposed method

According to previous studies in imputation techniques, most of them are based on the use of complete patterns from the input dataset to obtain missing values of incomplete samples. However, these techniques do not take into account any known values from incomplete samples, a fact that can be relevant in the learning process in order to impute more accurately.

In this section, a procedure which exploits all available information from the input data will be presented and its great capacity for classifying incomplete samples will be discussed. The method exploits the high representation capabilities of deep learning machines, and it is based on three concepts: pre-imputation, deletion and compensation.

More specifically, stacked denoising auto-encoders (SDAEs) are used in this paper for the following reason. When a neural network is trained with an input dataset which presents unknown values in some samples, it is very common to impute with numeric values. In fact, this is usually an attempt to get values as similar to the real ones as possible in order to avoid somehow the distortion of the input distribution. In this work, the initial imputation will be considered as a pre-imputation phase, and the values obtained will be treated as a noisy version of the real ones. This idea leads us to think about SDAEs as an appropriate option to construct an efficient imputation method.

2.1 SDAE in imputation

An auto-encoder (AE) is a neural network whose architecture is similar to that of an MLP with one hidden layer. It is trained to replicate at its output the same pattern presented to its input. In this way, hidden weights act as encoders of the input patterns, whereas output weights serve as decoders between the hidden and the output layers. Thus, an alternative representation of the input set is achieved by the hidden layer. This representation may be

compressive—if the hidden dimension is lower than input—or sparse—otherwise.

On the other hand, a denoising auto-encoder (DAE) is a variation in which the input is corrupted by artificially inserting some noise [27]. The concept of *denoising* consists in cleaning partially corrupted (or in short *noisy*) inputs, i.e., teaching the network to clean a noisy version $\tilde{\mathbf{x}}$ of the input $\mathbf{x}$. The main goal is to increase the robustness of the AE to face noisy inputs as well as to permit expensive hidden layers without leading to identity transformation. The type of noise used for training the network depends on the application.

A deep extension of the above mentioned DAE is the Stacked DAE (SDAE) [28]. Together with DBN [21], SDAEs are part of the widely known representation learning, a set of techniques characterized by the ability to automatically discover hierarchical representations of the input features. Basically, an SDAE is a deep network sequentially constructed by training and stacking several DAEs. It has been widely used due to its high representation capability as well as its robustness when noisy input datasets are dealt with.

In this work, the design and training of deep networks have been carried out according to [29]. In order to build each hidden layer of the deep architecture, every DAE will be trained as an expansive network. After being verified the performance over several datasets, an increment of 75% of the size of the previous layer has been established. Therefore, a DAE is firstly trained by using stochastic gradient and its hidden weights (with size $h_1 = 1.75 \times D$, where D is the dimension of input data) are used to set the first layer of the deep network. Then, this process is repeated twice obtaining hidden coders with sizes $h_2 = 1.75 \times h_1$ and $h_3 = 1.75 \times h_2$. For every individual training of each DAE, input data are corrupted with noise to increase the deep network robustness (Fig. 1).

In practice, one of the most commonly used techniques to train DAEs is stochastic gradient descent (SGD), where the optimal solution of the error function is reached through a stochastic approach. In order to exploit all available information from data during the training stage, a modified version of SGD has recently been proposed [26]. Thus, if we consider a set of data $\{\mathbf{x}_n\}_{n=1}^{N}$ with $\mathbf{x}_n$ as the nth-sample, another set of vectors with the same dimension is defined as $\{\mathbf{m}_n\}_{n=1}^{N}$ in order to determine the presence or absence of missing values. Thus

$$m_{\text{nd}} = \begin{cases} 1, & \text{if } x_{\text{nd}} \text{ is a known value} \\ 0, & \text{if } x_{\text{nd}} \text{ is missing} \end{cases} \quad (1)$$

where d, $1 \leq d \leq D$, denotes components. Furthermore, each stacked DAE in the final deep network is trained to

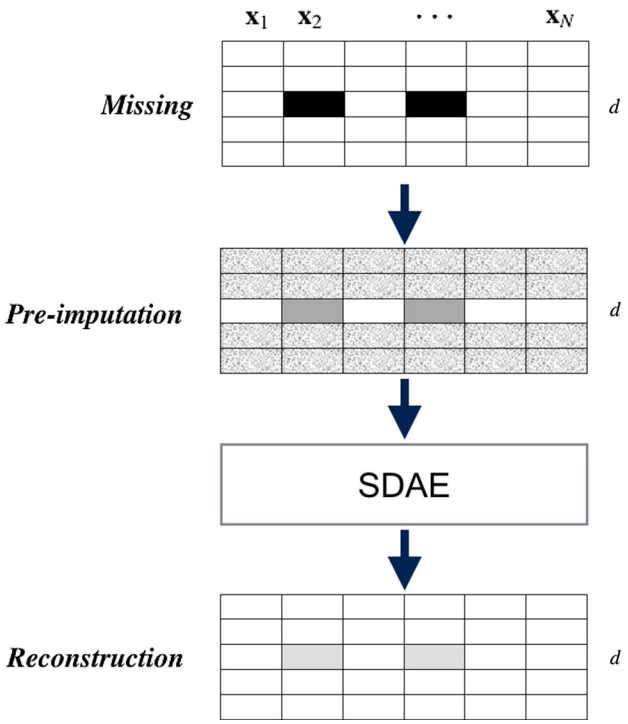

Fig. 1 SDAE for imputation. The process of how a SDAE imputes missing values from a pre-imputed version of the incomplete input dataset is shown. Black represents missing values in the original dataset. Without loss of generality and for the sake of clarity, missing is considered only in feature d. In the pre-imputation stage, it is included the pre-imputation values provided by the pre-imputation method (dark gray color); note that the partially corrupted (noisy) samples required by the SDAE are also included in this phase (dotted). Finally, samples are reconstructed, including original missing which is represented by light gray color

minimize the sum squared error (SSE) function with the form:

$$E = \sum_{n=1}^{N} \| (\mathbf{z}_n - \mathbf{x}_n) \odot \mathbf{m}_n \|^2 \quad (2)$$

where $\mathbf{z}_n$ is the output of the SDAE and $\odot$ represents the Hadamard product. Thanks to this formulation, all available data are used to train the network. These are all the values from complete samples and all the observed values from incomplete samples (samples with missing values).

During the training of a DAE, for each input sample $\mathbf{x}_n$, all hidden weights are updated. However, to train the output weights, the values of $\mathbf{m}_n$ must be taken into account, updating the weights corresponding to the values of $\mathbf{m}_n$ equal to 1.

As it has been mentioned above, the proposed imputation method with SDAEs is based upon three concepts we will explain in next sections: pre-imputation, deletion and compensation.

2.1.1 Pre-imputation

Replacing missing values by numeric values using a specific procedure will be called pre-imputation in this work. Samples with pre-imputed values can be considered as noisy samples. Under this idea, the use of an SDAE is justified to obtain an efficient imputation procedure.

Therefore, the first step of the method will be the pre-imputation, where unknown values are filled in. Then, the pre-imputed set will be used as training data for an imputation SDAE, with the objective of getting imputed values obtained by the deep network closer to the real data than the pre-imputed ones.

There are a lot of techniques in the literature that can be applied in the pre-imputation phase [30, 31]. One of the most simple methods consists in filling in missing values with zeros. However, if the pre-imputation method is able to achieve better results in the first step, it seems reasonable to expect better results with the whole process. In this work, in order to evaluate the performance of the proposed technique, several pre-imputation techniques will be applied with different levels of complexity. More specifically, three methods will be used: auxiliary MLPs, singular value decomposition and MICE.

2.1.2 Deletion

Once the pre-imputation phase has been carried out, the pre-imputed and the observed values are used during the training of the first DAE as inputs. Nevertheless, only the known values are used as targets, according to (2). Obviously, if we used pre-imputed values to train the network, this would reach sub-optimal solutions due to the fact that the training would try to reproduce them.

Deletion is connected with the possibility of explicitly helping the network to learn the reconstruction of missing values. This is possible by artificially forcing some other missing values. Let us assume that a dataset has missing values in the feature d with a percentage expressed as a decimal μ $(0 \leq \mu \leq 1)$. This missing percentage can be artificially increased in a value ε with $0 \leq \varepsilon \leq 1 - \mu$, in such a way that $(\mu + \varepsilon)N$ will be the effective missing rate over the total number of samples N. The network is hence trained with N input samples with $(\mu + \varepsilon)N$ incomplete samples in feature d, and N target samples with μN unknown values in the same feature. While μ is a constraint of the problem to solve, ε will be a design parameter which can be set by cross-validation. In this way, it is hoped this inserted missing values, whose real values are used as targets, help the neural network to learn how to reconstruct feature d more accurately.

The deletion process is shown in Fig. 2. Here, the μN missing values are represented in black and εN deleted samples are crossed out. After pre-imputation and before the deep learning process, the darker gray samples mean that their real values are actually known and, hence, they can be used as targets during training. All these values are reconstructed at the output, that is, not only missing but also deleted values.

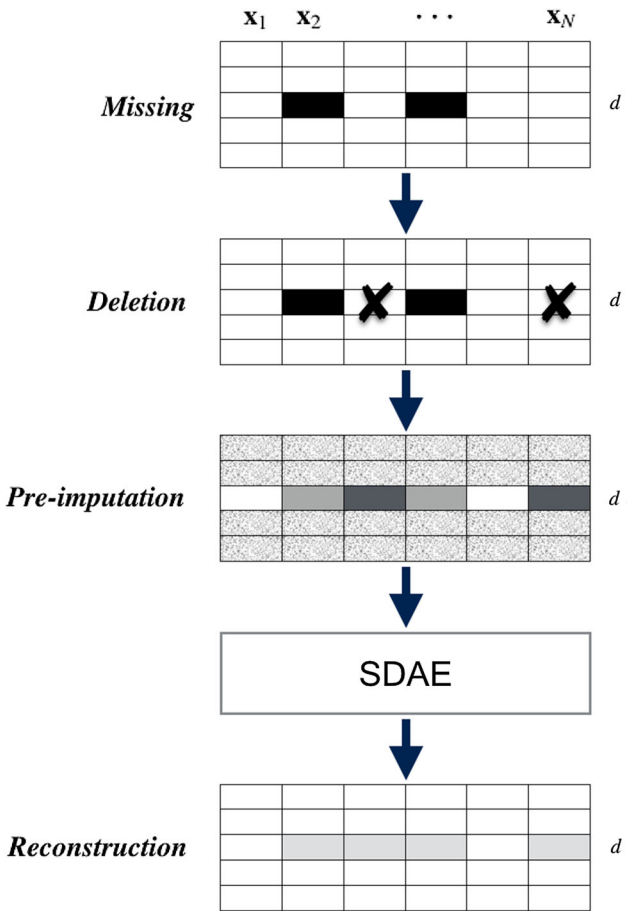

Fig. 2 Deletion process. Missing values in feature d are shown in black, and deletion is represented with a cross. Light and dark gray colors in the pre-imputation stage are used to show pre-imputed values for missing and deleted samples, respectively. The known values of deleted samples will be used as targets during training. Finally, the lightest gray samples represent reconstructed values by the deep network

Although this idea of deleting values to improve the imputation performance was previously presented in [26], the goal of this work is to go deeper into this concept, completing and improving the method with the compensation procedure which is introduced below.

2.1.3 Compensation

Despite the advantages that deletion can involve for imputation, there exists a main drawback due to the fact that both training and test sets get imbalanced during the process, because the rate of unavailable values is different.

Thus, by means of deletion, initial missing rate μ and complete rate $1-\mu$ are turned into $\mu+\varepsilon$ missing and $1-(\mu+\varepsilon)$ complete rates. However, the test set remains with the same missing rate μ and the complete rate $1-\mu$. In this scenario, if the value of ε is small, the imbalancing between sets is usually of little significance, but otherwise, it may appear a performance degradation that can be handled with a suitable compensation.

Next, the exact compensation is analyzed and an effective and practical compensation procedure is proposed.

Before introducing the deletion process, the error function SSE can be expressed as

$$E_1 = E_C + E_I \tag{3}$$

where E_C and E_I are the SSE of complete and incomplete samples, respectively. This expression can be written in terms of the missing rates as

$$E_1 = (1-\mu)N\overline{E}_C + \mu N\overline{E}_I \tag{4}$$

where $\overline{E}_C$ and $\overline{E}_I$ are the mean squared error (MSE) of the complete and incomplete samples, respectively. The first term represents the squared error contribution of the $(1-\mu)N$ complete samples, and the second term represents the corresponding contribution of the μN samples with missing values.

After the deletion process, the distribution of training data changes, while that of the test data does not. At the same time, the terms of the SSE will change because the missing rate has been modified. Therefore,

$$E_2 = E'_C + E'_I \tag{5}$$

where the two terms of E incorporate the additional εN missing values according to the following expression

$$E_2 = (1-(\mu+\varepsilon))N\overline{E'}_C + (\mu+\varepsilon)N\overline{E'}_I \tag{6}$$

where $\overline{E'}_C$ and $\overline{E'}_I$ are MSE of the complete and incomplete samples after deletion, respectively.

With the aim of treating both training and test sets equally, compensating the difference introduced in theirs distributions, a mechanism to balance them is proposed. In order to obtain an exact balance after deletion, training should be performed with an effective rate of $1-\mu$ complete samples and μ incomplete samples, i.e., as in (4).

According to our experiments, the exact re-balance of datasets is not always the best option, so a range of values for the weighting process must be explored. Thus, given μ and ε, the next convex function is proposed to be optimized

$$E = \alpha E'_C + (1-\alpha)E'_I \tag{7}$$

where α is the balancing parameter between $\alpha = 0$ (only errors of complete samples are optimized) and $\alpha = 1$ (only errors of incomplete samples). Note that for $\alpha = 0.5$ both errors are equally balanced, that is, (7) is equivalent to (5). The optimal value for α is obtained via cross-validation according to the accuracy of the network on a validation set without deletion. The objective is to find the optimal weighting that produces the best performance on a validation set similar to the test set (with the missing rate of the original problem).

After the deletion process, the number of samples with missing values increases and the number of complete samples decreases. To compensate this change, it is expected to obtain optimal values of α greater than 0.5 to pay more attention to errors of complete samples (which have decreased) than to those of the incomplete samples (which have increased). As it will be seen, the experimental results in Sect. 3 support this assumption.

2.2 Classification of incomplete patterns

The method presented so far has been focused on imputation. However, the SDAEs are frequently used to solve classification and regression problems. It has been demonstrated in the literature that the potential of an SDAE relies on the unsupervised pre-training of individual stacked DAEs used to create the deep network. This pre-training is used to initialize the hidden weights, helping to solve the optimization problem. Moreover, making noisy inputs helps the model to avoid the identity solution and, thus, to be more robust to distortions.

Training an SDAE consists of two main phases. First, several DAEs are designed and trained in an unsupervised way as it has been shown in Sect. 2.1. Secondly, a classifier above the output of the stacked network (output hidden layer of the last AE) is trained to solve the problem. This training can be carried out in two different ways: on the one hand, the hidden weights obtained in the first step are fixed and those related to the classifier are updated. In this case, a global fine tuning is necessary. On the other hand, after the unsupervised pre-training, both the classifier and the hidden weights can be trained together.

In this paper, the performance of SDAEs that are used for imputation is analyzed when they face classification problems with missing values in some features, in order to show how the classification performance is improved when SDAEs are trained using deletion and compensation.

3 Experiments

In this section, the impact of deletion and compensation will be analyzed when SDAEs face imputation and classification tasks. Firstly, the effects of these two techniques on the imputation will be analyzed. To do this, several

classical imputations techniques are compared to the SDAE with deletion and compensation. Thus, missing values are pre-imputed through one of the classical imputation methods already mentioned. This allows us to verify the improvement achieved by the technique proposed here. Secondly, the performance of this kind of SDAE is analyzed when it classifies unseen incomplete patterns.

Six datasets have been selected in this study: *Magic Gamma Telescope*, *Pima Indians Diabetes*, *Sensorless Drive Diagnosis*, *Gas Sensor Array Drift*, *Activity Recognition system based on Multisensor data fusion (AReM)* and *Twonorm*. The first five datasets can be found in the UCI repository [32], whereas *Twonorm* is available in [33]. The specific details of each set are shown in Table 1. Only complete datasets have been selected in order to artificially insert missing values and thus verify the correct behavior of the proposed algorithms. Incomplete values will be inserted in a randomly selected feature for every dataset except for AReM. To expand the scope of the study, four features have been randomly selected in that case. As it has been mentioned in the previous sections, there are several missing categories that can affect the efficiency of the selected imputation technique. In this work, the strictest missing type will be taken into account, that is, missing completely at random (MCAR), where missing values are produced completely at random and they have no relationship with observed values.

During training, each dataset is split into training (80%) and test (20%) sets. As it has been mentioned in Sect. 2.1, all deep networks are designed to have 3 hidden expansive layers with an incremental expansion of 75% of the size of the previous layer. Both to find the optimum value of the weights and the hyper-parameters in a particular range of possible values, a fivefold CV is performed for each value. It consists of dividing the training set into 5 subsets of the same size, using 4 of them for training and one for validation. For each one of the 5 possible scenarios (different validation datasets), the network is trained 50 times averaging the 50 validation errors. After completing the process, a global average error of validation is obtained, which is used to select the optimum value of the hyper-parameter.

Table 1 Datasets (main features)

	Samples		
	Total	Classes	Features
MAGIC Gamma telescope (M)	19,020	2	11
Pima Indians diabetes (P)	768	2	8
Twonorm (T)	7400	2	20
Sensorless drive diagnosis (S)	58,509	11	48
Gas sensor array drift (G)	13,910	6	128
AReM (A)	42,240	7	6

Then, using this value, the network is trained using the whole training set and, finally, its performance is evaluated with the test set.

In addition, different scenarios have been simulated to test the performance of each technique, dealing with several missing μ and deleted samples ε percentages. In particular, all possible combinations have been tested with $\mu = \{0.1, 0.2, 0.3\}$ and $\varepsilon = \{0.25, 0.5, 0.75\}$.

3.1 Imputation results

There exist a lot of imputation techniques that have been studied in the literature. Among them, three representative methods with different complexity and capabilities have been selected. These procedures are implemented to pre-impute missing values of the data which will be used to train the SDAE. We will refer to them from now on as pre-imputation methods. These methods are:

- *Multilayer Perceptron imputation (MLP)* The MLP imputation mechanism is a prediction model, that is, a predictive model which estimates values that will replace the missing data. It is considered as a regression imputation since the missing components are filled in by the predicted values from a regression analysis. As it is shown in [16], an MLP is trained only using the complete cases and considering the characteristics with missing values as targets. Once trained, incomplete patterns are processed and MLP outputs are used to impute unknown values. When several attributes are missing, some MLP schemes have to be designed. In this work, a fivefold CV process has been used to set the number of hidden nodes in these MLP architectures.
- *Singular value decomposition (SVD)* This method, based on the work [12], approximates missing data through patterns obtained by a set of linearly combined expressions. These patterns, identical to the principal components, can be cleverly selected so that most of the variance data is preserved. It is demonstrated that only several significant patterns are sufficient to do it. It is important to note that SVD is only for complete arrays, that is, first missing values have to be substituted by the mean of the feature.
- *Multiple imputation by chained equations (MICE)* MICE imputation has emerged in the last years as one of the most effective methods in imputation. In general, multiple imputation techniques have a number of advantages over single imputation ones. These methods involve filling in unknown values several times so that uncertainty in the imputations is taken into account and the error variances are reduced. These multiple imputations are implemented by a simple technique such as attributing average values, and a regression model is

implemented in order to iteratively improve the results. However, although multiple imputations will generally produce better results than a simple imputation, in many cases they are not the best option, becoming time-intensive when applied to large datasets [34]. More details about the MICE technique implemented in this work can be obtained in [35].

In Table 2, the imputation results for each algorithm and dataset are shown. Errors between imputed and real values are presented according to (2) and (7) for methods without and with compensation, respectively. Results achieved by the SDAEs that are designed and trained with pre-imputed data as well as results for selected pre-imputation methods directly applied over the missing data are presented. Three cases are distinguished for results based on deep networks: SDAE without deletion nor compensation, SDAE with only deletion, and SDAE with deletion and compensation. The optimal values for ε and α are presented inside parentheses, the best results for the pre-imputation group are shown in italics and the global best result for missing percentage is in bold. Two results are considered statistically different if $\phi_1 - \phi_2 \geq \frac{\sigma_1+\sigma_2}{2}$, where ϕ represents the mean and σ the standard deviation of the result.

3.2 Classification results

In Table 3, classification results are presented. For each algorithm and dataset, results are shown in terms of mean and standard deviation of classification accuracy. All of them are obtained with 50 independent runs. It is important to note that both presentation and comparison of results are done in the same way as in the previous section as well as that results for pre-imputation are obtained through an MLP trained with cross-validation.

For each experiment, deep networks are designed using a linear classifier as output of the SDAE. A linear classifier has been chosen because, in this case, it offers similar results to other more complex networks like an MLP. This means to accept that SDAEs successfully disentangles the classes [36], thus allowing the use of a simple classifier in the output. Once initialized the hidden weights through the first phase of the algorithm, the global network is jointly trained, that is, the linear output classifier is trained and the hidden weights are fine tuned.

3.3 Discussion

Analyzing the imputation results, it can be concluded:

- The pre-imputation results are always improved by a solution based on a SDAE regardless of what pre-imputation is implemented.
- The better pre-imputation the better final imputation accuracy with the SDAE.
- SDAE networks with deletion always improve the performance of SDAEs without deletion.
- SDAE networks with deletion and compensation improve the performance of SDAEs only with deletion in the majority of the cases. This is a clear evidence of the usefulness of compensation.
- According to the optimal values of ε, it can be concluded that a high deletion rate provides better results. This permits to conclude that deletion is useful to learn a better reconstruction of missing values.
- Most of the optimal values of α are slightly above 0.5, so that we can conclude that giving more importance to complete samples implies an improvement of the imputation process.
- Great results are achieved in the case of AReM dataset, when missing is inserted in several random features. This means the method generalizes well in different types of scenarios and situations of missing values, which is not a very surprising result due to the high representation capability of auto-encoders together with the compensation mechanisms introduced.

Analyzing the classification results, it can be concluded that:

- Deletion (with or without compensation) gets better results in most cases, that is, not only improves imputation results but also helps to get better performance of the SDAE classifier.
- In Table 4, the effect of compensation is presented showing the number of times it improves, degrades or equals the results with deletion. As it can be seen, results are worse only 1 time (2% of total scenarios), are better 24 times (45%) and considered statistically equal the other 29 times (53%).
- As we have already seen, although compensation always produces an improvement of the imputation results, in some cases it does not obtain a significant improvement in the classification performance. This means that, even though the obtained output values are closer to the desired ones, this improvement is not enough for allowing to differentiate the classes.
- It is very interesting to observe the results obtained when the multiple imputation MICE algorithm is applied. In this case, while the improvement obtained by our method is very moderate in imputation tasks, the improvement in classification is really significant. Thus, it can be concluded that the great representation capability of SDAEs is key to solve classification tasks, if a good pre-imputation is applied.
- Finally, it can be seen that classification results are improved with deletion and compensation even in

Table 2 Imputation results for each experiment carried out

Procedure	μ		
	0.1	0.2	0.3
M-*MLP*	0.5 ± 0.05	0.4 ± 0.05	0.4 ± 0.09
M-*MLP*-SDAE	0.2 ± 0.06	0.4 ± 0.02	0.2 ± 0.06
M-*MLP*-SDAE (ε)	0.1 ± 0.02 (0.75)	0.2 ± 0.06 (0.75)	**0.1 ± 0.02 (0.5)**
M-*MLP*-SDAE (ε, α)	**0.05 ± 0.01 (0.75, 0.6)**	**0.1 ± 0.07 (0.75, 0.3)**	**0.08 ± 0.006 (0.75, 0.7)**
M-*SVD*	0.7 ± 0.05	0.6 ± 0.05	0.7 ± 0.04
M-*SVD*-SDAE	0.3 ± 0.03	0.5 ± 0.06	0.3 ± 0.02
M-*SVD*-SDAE (ε)	0.15 ± 0.01 (0.5)	0.1 ± 0.01 (0.5)	0.25 ± 0.03 (0.75)
M-*SVD*-SDAE (ε, α)	**0.06 ± 0.01 (0.75, 0.2)**	**0.08 ± 0.01 (0.5, 0.7)**	0.11 ± 0.01 (0.75, 0.5)
M-*MICE*	**0.04 ± 0.004**	**0.08 ± 0.005**	**0.1 ± 0.01**
M-*MICE*-SDAE	**0.05 ± 0.008**	0.15 ± 0.01	**0.1 ± 0.02**
M-*MICE*-SDAE (ε)	**0.04 ± 0.005 (0.5)**	**0.1 ± 0.005 (0.5)**	**0.08 ± 0.03 (0.75)**
M-*MICE*-SDAE (ε, α)	**0.05 ± 0.005 (0.75, 0.2)**	**0.09 ± 0.01 (0.5, 0.7)**	**0.1 ± 0.01 (0.75, 0.5)**
P-*MLP*	0.5 ± 0.02	0.5 ± 0.01	0.6 ± 0.02
P-*MLP*-SDAE	0.2 ± 0.05	0.7 ± 0.02	0.6 ± 0.08
P-*MLP*-SDAE (ε)	0.1 ± 0.01 (0.5)	0.2 ± 0.02 (0.5)	0.09 ± 0.002 (0.5)
P-*MLP*-SDAE (ε, α)	**0.03 ± 0.005 (0.75, 0.7)**	*0.08 ± 0.01 (0.75, 0.6)*	**0.05 ± 0.006 (0.5, 0.8)**
P-*SVD*	0.65 ± 0.02	0.7 ± 0.008	0.7 ± 0.02
P-*SVD*-SDAE	0.4 ± 0.04	0.35 ± 0.04	0.4 ± 0.004
P-*SVD*-SDAE (ε)	0.2 ± 0.01 (0.75)	0.15 ± 0.02 (0.5)	0.25 ± 0.01 (0.75)
P-*SVD*-SDAE (ε, α)	*0.05 ± 0.008 (0.75, 0.3)*	**0.07 ± 0.01 (0.75, 0.7)**	**0.06 ± 0.004 (0.75, 0.5)**
P-*MICE*	**0.02 ± 0.008**	0.07 ± 0.004	**0.05 ± 0.008**
P-*MICE*-SDAE	0.04 ± 0.005	0.07 ± 0.005	0.08 ± 0.01
P-*MICE*-SDAE (ε)	**0.02 ± 0.005 (0.75)**	**0.05 ± 0. (0.5)**	**0.06 ± 0.004 (0.75)**
P-*MICE*-SDAE (ε, α)	**0.02 ± 0.004 (0.75, 0.7)**	**0.06 ± 0.004 (0.5, 0.4)**	**0.05 ± 0.005 (0.75, 0.7)**
T-*MLP*	0.9 ± 0.01	0.9 ± 0.05	0.8 ± 0.03
T-*MLP*-SDAE	0.6 ± 0.09	0.8 ± 0.1	0.5 ± 0.05
T-*MLP*-SDAE (ε)	0.08 ± 0.01 (0.5)	0.1 ± 0.002 (0.5)	0.08 ± 0.01 (0.75)
T-*MLP*-SDAE (ε, α)	**0.05 ± 0.006 (0.75, 0.6)**	**0.04 ± 0.004 (0.75, 0.7)**	**0.05 ± 0.008 (0.75, 0.7)**
T-*SVD*	0.75 ± 0.03	0.8 ± 0.05	0.8 ± 0.05
T-*SVD*-SDAE	0.5 ± 0.02	0.2 ± 0.04	0.4 ± 0.05
T-*SVD*-SDAE (ε)	*0.14 ± 0.02 (0.5)*	0.2 ± 0.03 (0.75)	0.2 ± 0.02 (0.75)
T-*SVD*-SDAE (ε, α)	*0.12 ± 0.01 (0.5, 0.2)*	**0.05 ± 0.01 (0.75, 0.5)**	*0.12 ± 0.01 (0.75, 0.4)*
T-*MICE*	0.04 ± 0.01	**0.04 ± 0.001**	0.07 ± 0.005
T-*MICE*-SDAE	0.07 ± 0.006	0.07 ± 0.003	0.08 ± 0.004
T-*MICE*-SDAE (ε)	0.07 ± 0.002 (0.5)	0.05 ± 0.005 (0.5)	**0.05 ± 0.003 (0.5)**
T-*MICE*-SDAE (ε, α)	**0.05 ± 0.003 (0.5, 0.8)**	**0.04 ± 0.002 (0.5, 0.7)**	**0.05 ± 0.01 (0.75, 0.6)**
S-*MLP*	2.65 ± 0.08	2.82 ± 0.1	2.89 ± 0.07
S-*MLP*-SDAE	2.36 ± 0.12	2.53 ± 0. 07	2.45 ± 0.08
S-*MLP*-SDAE (ε)	1.73 ± 0.07 (0.75)	1.9 ± 0.08 (0.5)	1.85 ± 0.02 (0.5)
S-*MLP*-SDAE (ε, α)	*1.24 ± 0.08 (0.5, 0.4)*	*1.71 ± 0.05 (0.75, 0.6)*	*1.56 ± 0.04 (0.5, 0.6)*
S-*SVD*	2.45 ± 0.08	2.57 ± 0.05	2.6 ± 0.05
S-*SVD*-SDAE	2.23 ± 0.1	2.34 ± 0.08	2.4 ± 0.05
S-*SVD*-SDAE (ε)	1.84 ± 0.06 (0.5)	*1.88 ± 0.02 (0.5)*	2.12 ± 0.02 (0.75)
S-*SVD*-SDAE (ε, α)	*1.57 ± 0.05 (0.75, 0.6)*	*1.94 ± 0.1 (0.5, 0.8)*	*1.93 ± 0.07 (0.75, 0.7)*
S-*MICE*	**0.82 ± 0.04**	**0.81 ± 0.05**	**0.92 ± 0.03**
S-*MICE*-SDAE	1.47 ± 0.1	1.24 ± 0.02	1.36 ± 0.05
S-*MICE*-SDAE (ε)	1.2 ± 0.06 (0.5)	**0.97 ± 0.1 (0.5)**	1.15 ± 0.06 (0.75)

Table 2 (continued)

Procedure	μ		
	0.1	0.2	0.3
S-*MICE*-SDAE (ε, α)	**0.91 ± 0.08 (0.75, 0.7)**	**0.85 ± 0.08 (0.75, 0.6)**	**0.87 ± 0.04 (0.5, 0.6)**
G-*MLP*	2.65 ± 0.05	2.85 ± 0.04	3.12 ± 0.03
G-*MLP*-SDAE	2.34 ± 0.07	2.54 ± 0.06	2.74 ± 0.06
G-*MLP*-SDAE (ε)	1.78 ± 0.05 (0.5)	*1.74 ± 0.07 (0.75)*	2.54 ± 0.04 (0.75)
G-*MLP*-SDAE (ε, α)	*1.56 ± 0.08 (0.75, 0.7)*	*1.82 ± 0.08 (0.75, 0.6)*	*2.37 ± 0.03 (0.75, 0.6)*
G-*SVD*	2.3 ± 0.04	2.64 ± 0.02	2.57 ± 0.03
G-*SVD*-SDAE	2.45 ± 0.05	2.51 ± 0.05	2.72 ± 0.1
G-*SVD*-SDAE (ε)	2.06 ± 0.03 (0.5)	*2.32 ± 0.05 (0.5)*	*2.35 ± 0.06 (0.5)*
G-*SVD*-SDAE (ε, α)	*1.83 ± 0.06 (0.5, 0.7)*	*2.25 ± 0.04 (0.75, 0.6)*	*2.32 ± 0.05 (0.75, 0.6)*
G-*MICE*	**1.12 ± 0.02**	**1.1 ± 0.02**	**0.92 ± 0.04**
G-*MICE*-SDAE	1.46 ± 0.04	1.34 ± 0.05	1.54 ± 0.06
G-*MICE*-SDAE (ε)	1.23 ± 0.06 (0.5)	1.19 ± 0.04 (0.5)	1.17 ± 0.08 (0.75)
G-*MICE*-SDAE (ε, α)	1.2 ± 0.04 (0.5, 0.5)	**0.92 ± 0.07 (0.75, 0.6)**	**0.95 ± 0.05 (0.75, 0.6)**
A-*MLP*	0.018 ± 0.002	0.021 ± 0.002	0.021 ± 0.002
A-*MLP*-SDAE	0.017 ± 0.001	0.019 ± 0.002	0.020 ± 0.002
A-*MLP*-SDAE (ε)	0.015 ± 0.001 (0.5)	0.019 ± 0.001 (0.25)	**0.017 ± 0.001 (0.5)**
A-*MLP*-SDAE (ε, α)	**0.012 ± 0.001 (0.5, 0.6)**	*0.018 ± 0.002 (0.5, 0.7)*	**0.017 ± 0.002 (0.5, 0.8)**
A-*SVD*	0.07 ± 0.001	0.05 ± 0.002	0.1 ± 0.002
A-*SVD*-SDAE	0.04 ± 0.001	0.03 ± 0.001	0.05 ± 0.002
A-*SVD*-SDAE (ε)	*0.016 ± 0.001 (0.5)*	*0.018 ± 0.001 (0.75)*	0.02 ± 0.002 (0.5)
A-*SVD*-SDAE (ε, α)	*0.015 ± 0.002 (0.75, 0.7)*	*0.019 ± 0.001 (0.75, 0.6)*	0.02 ± 0.001 (0.75, 0.6)
A-*MICE*	0.016 ± 0.002	0.019 ± 0.001	0.018 ± 0.001
A-*MICE*-SDAE	0.016 ± 0.001	0.019 ± 0.002	0.017 ± 0.002
A-*MICE*-SDAE (ε)	**0.013 ± 0.002 (0.75)**	**0.016 ± 0.001 (0.5)**	**0.015 ± 0.001 (0.5)**
A-*MICE*-SDAE (ε, α)	**0.012 ± 0.001 (0.75, 0.7)**	**0.016 ± 0.001 (0.5, 0.7)**	**0.016 ± 0.001 (0.5, 0.6)**

SSE described in (2) and (7) for methods without and with compensation, respectively, is presented. Different percentages of missing values μ in the input dataset are shown in columns. Several procedures are presented in rows according to the following nomenclature: first initial of the dataset—*pre-imputation method*—deep procedure *(deletion rate, compensation rate)*. For each dataset, the best result of a family of pre-imputation methods is shown in italics and the global minimum for a missing percentage in bold. Results without statistically difference are considered equal

AReM, a dataset with missing completely at random in several features.

4 Conclusions

The presence of missing values in a dataset is one of the most common difficulties when we face real applications. This work has been motivated by the idea of taking advantage of the great representation capability of SDAEs in imputating missing values as well as in classification tasks.

We have introduced a technique capable of improving imputation of missing samples by artificially deleting input values. Although this technique has been demonstrated to be efficient, it may cause an imbalance between distributions of training and test sets. In order to solve this, a compensation mechanism is proposed. It is based on a slight modification of the error function to be optimized. It is important to emphasize that, although some input values are deleted in this procedure, their corresponding known values are used as output targets during training, what produces an improvement in imputation.

Deletion and compensation are also useful mechanisms to improve classification performance. It has been shown that an SDAE followed by a linear classifier produces better results when deletion is considered in the training phase. Nevertheless, while the best option in imputation includes compensation (along with deletion), it does not always get a better performance in classification tasks. From the results obtained on different datasets, it is observed that improvements only occur in approximately 45% of the cases. In the rest, although the proposed method

Table 3 Classification accuracy for every experiment carried out

Procedure	μ		
	0.1	0.2	0.3
M-*MLP*	89.3 ± 1.6	87.3 ± 0.7	88.6 ± 0.4
M-*MLP*-SDAE	89.8 ± 0.7	89.7 ± 0.2	87.4 ± 0.2
M-*MLP*-SDAE (ε)	**92 ± 0.2 (0.75)**	**91.6 ± 0.3 (0.5)**	*90.9 ± 0.5 (0.75)*
M-*MLP*-SDAE (ε, α)	**92.3 ± 0.2 (0.5, 0.6)**	**91.3 ± 0.2 (0.75, 0.6)**	*90.8 ± 0.4 (0.75, 0.7)*
M-*SVD*	88.5 ± 0.4	87.1 ± 0.2	88.2 ± 0.2
M-*SVD*-SDAE	90.1 ± 0.6	89.4 ± 0.5	87.5 ± 0.6
M-*SVD*-SDAE (ε)	91.2 ± 0.3 (0.5)	**91.3 ± 0.2 (0.5)**	88.4 ± 0.4 (0.75)
M-*SVD*-SDAE (ε, α)	**91.9 ± 0.2 (0.5, 0.5)**	**91.5 ± 0.4 (0.5, 0.7)**	**91.3 ± 0.1 (0.75, 0.4)**
M-*MICE*	89.7 ± 0.6	90.3 ± 0.2	89.4 ± 0.2
M-*MICE*-SDAE	89.6 ± 0.8	**90.8 ± 0.4**	89.3 ± 0.5
M-*MICE*-SDAE (ε)	91.2 ± 0.4 (0.5)	**90.4 ± 0.5 (0.75)**	90.2 ± 0.3 (0.75)
M-*MICE*-SDAE (ε, α)	**92.3 ± 0.6 (0.75, 0.7)**	**91.2 ± 0.4 (0.75, 0.5)**	**91.4 ± 0.5 (0.75, 0.6)**
P-*MLP*	72.4 ± 1.7	70.3 ± 0.8	74.6 ± 1.3
P-*MLP*-SDAE	77.3 ± 2.8	72.8 ± 3.4	75 ± 1.2
P-*MLP*-SDAE (ε)	76.7 ± 1.6 (0.5)	**78 ± 1.3 (0.5)**	**79.8 ± 2.0 (0.5)**
P-*MLP*-SDAE (ε, α)	**80.8 ± 1.2 (0.75, 0.7)**	**79.4 ± 1.5 (0.75, 0.3)**	**80.5 ± 1.1 (0.5, 0.7)**
P-*SVD*	74.6 ± 0.4	74.3 ± 0.8	73.2 ± 0.4
P-*SVD*-SDAE	75.3 ± 0.6	76.2 ± 0.5	75.3 ± 0.7
P-*SVD*-SDAE (ε)	78.6 ± 0.4 (0.25)	**79.5 ± 0.4 (0.75)**	78.1 ± 0.2 (0.75)
P-*SVD*-SDAE (ε, α)	**79.5 ± 0.7 (0.5, 0.5)**	78.3 ± 0.2 (0.75, 0.6)	**79.2 ± 0.5 (0.75, 0.7)**
P-*MICE*	77.4 ± 0.7	77.8 ± 0.3	77.5 ± 0.4
P-*MICE*-SDAE	78.3 ± 0.2	78.1 ± 0.4	78.1 ± 0.5
P-*MICE*-SDAE (ε)	79.5 ± 0.3 (0.5)	**79.4 ± 0.4 (0.25)**	78.7 ± 0.1 (0.75)
P-*MICE*-SDAE (ε, α)	**80.7 ± 0.4 (0.5, 0.3)**	**79.5 ± 0.2 (0.5, 0.7)**	**79.3 ± 0.3 (0.75, 0.6)**
T-*MLP*	91.8 ± 1.3	91.2 ± 0.6	90 ± 1.4
T-*MLP*-SDAE	92.2 ± 0.2	95.5 ± 0.2	94.7 ± 1.4
T-*MLP*-SDAE (ε)	**98.3 ± 0.6 (0.5)**	**97.6 ± 0.8 (0.5)**	**97.6 ± 0.4 (0.75)**
T-*MLP*-SDAE (ε, α)	**97.8 ± 0.5 (0.75, 0.6)**	**97.1 ± 0.2 (0.75, 0.6)**	**98.1 ± 0.2 (0.75, 0.6)**
T-*SVD*	90.3 ± 0.4	90.4 ± 0.2	89.4 ± 0.5
T-*SVD*-SDAE	93.5 ± 0.5	92.7 ± 0.4	91.2 ± 0.4
T-*SVD*-SDAE (ε)	97.1 ± 0.3 (0.5)	96.3 ± 0.3 (0.75)	96.5 ± 0.7 (0.75)
T-*SVD*-SDAE (ε, α)	**97.9 ± 0.4 (0.5, 0.2)**	**97.4 ± 0.6 (0.75, 0.7)**	**97.8 ± 0.4 (0.75, 0.8)**
T-*MICE*	96.3 ± 0.2	95.8 ± 0.3	95.4 ± 0.2
T-*MICE*-SDAE	**97.5 ± 0.4**	96.2 ± 0.4	96.4 ± 0.5
T-*MICE*-SDAE (ε)	**98.2 ± 0.3 (0.5)**	**97.2 ± 0.2 (0.75)**	**97.3 ± 0.2 (0.75)**
T-*MICE*-SDAE (ε, α)	**97.9 ± 0.3 (0.5, 0.5)**	**97.7 ± 0.3 (0.75, 0.6)**	**98.3 ± 0.3 (0.75, 0.7)**
S-*MLP*	94.5 ± 0.3	93.2 ± 0.3	94.3 ± 0.5
S-*MLP*-SDAE	95.6 ± 0.6	94.5 ± 0.3	95.2 ± 0.3
S-*MLP*-SDAE (ε)	97.3 ± 0.3 (0.75)	*97.7 ± 0.6 (0.75)*	97.4 ± 0.2 (0.75)
S-*MLP*-SDAE (ε, α)	**98.7 ± 0.4 (0.5, 0.8)**	*97.3 ± 0.4 (0.75, 0.7)*	**98.2 ± 0.6 (0.75, 0.6)**
S-*SVD*	95.3 ± 0.4	95.3 ± 0.4	94.8 ± 0.4
S-*SVD*-SDAE	95.7 ± 0.6	94.6 ± 0.5	96.3 ± 0.5
S-*SVD*-SDAE (ε)	97.7 ± 0.4 (0.75)	97.5 ± 0.4 (0.75)	97.4 ± 0.3 (0.75)
S-*SVD*-SDAE (ε, α)	*97.9 ± 0.4 (0.75, 0.6)*	**98.4 ± 0.6 (0.75, 0.7)**	**98.4 ± 0.2 (0.5, 0.6)**
S-*MICE*	97.8 ± 0.1	97.6 ± 0.2	97.7 ± 0.4
S-*MICE*-SDAE	98.2 ± 0.2	**98.4 ± 0.4**	98.2 ± 0.1
S-*MICE*-SDAE (ε)	**99.1 ± 0.3 (0.5)**	**99.1 ± 0.2 (0.75)**	**98.7 ± 0.3 (0.75)**

Table 3 (continued)

Procedure	μ		
	0.1	0.2	0.3
S-*MICE*-SDAE (ε, α)	**99.2 ± 0.2 (0.5, 0.6)**	**98.6 ± 0.4 (0.75, 0.6)**	**98.9 ± 0.2 (0.75, 0.6)**
G-*MLP*	96.4 ± 0.4	95.7 ± 0.4	95.0 ± 0.4
G-*MLP*-SDAE	97.6 ± 0.4	97.1 ± 0.2	96.8 ± 0.4
G-*MLP*-SDAE (ε)	**98.6 ± 0.5 (0.75)**	97.7 ± 0.3 (0.75)	98.1 ± 0.2 (0.75)
G-*MLP*-SDAE (ε, α)	**98.8 ± 0.4 (0.75, 0.5)**	*98.4 ± 0.2 (0.5, 0.4)*	**98.8 ± 0.4 (0.75, 0.4)**
G-*SVD*	97.1 ± 0.2	97.3 ± 0.4	96.8 ± 0.3
G-*SVD*-SDAE	97.5 ± 0.2	98.1 ± 0.2	96.6 ± 0.6
G-*SVD*-SDAE (ε)	*98.2 ± 0.2 (0.5)*	**98.7 ± 0.2 (0.75)**	97.6 ± 0.2 (0.75)
G-*SVD*-SDAE (ε, α)	*98.6 ± 0.1 (0.5, 0.4)*	**98.6 ± 0.3 (0.75, 0.6)**	**98.7 ± 0.4 (0.75, 0.6)**
G-*MICE*	98.2 ± 0.2	97.8 ± 0.3	98.0 ± 0.2
G-*MICE*-SDAE	98.5 ± 0.1	98.2 ± 0.4	98.6 ± 0.2
G-*MICE*-SDAE (ε)	**99.1 ± 0.2 (0.5)**	98.6 ± 0.2 (0.75)	**99.2 ± 0.1 (0.75)**
G-*MICE*-SDAE (ε, α)	**99.4 ± 0.2 (0.5, 0.6)**	**99.1 ± 0.1 (0.75, 0.8)**	**99.1 ± 0.1 (0.75, 0.6)**
A-*MLP*	86.7 ± 0.4	84.7 ± 0.2	86.8 ± 0.1
A-*MLP*-SDAE	87.1 ± 0.2	85.2 ± 0.2	87.3 ± 0.3
A-*MLP*-SDAE (ε)	87.5 ± 0.2 (0.25)	*87.4 ± 0.3 (0.5)*	88.7 ± 0.1 (0.5)
A-*MLP*-SDAE (ε, α)	*89.2 ± 0.1 (0.75, 0.6)*	*87.6 ± 0.1 (0.25, 0.5)*	*89.2 ± 0.2 (0.75, 0.6)*
A-*SVD*	88.3 ± 0.1	87.3 ± 0.2	87.1 ± 0.2
A-*SVD*-SDAE	88.6 ± 0.2	88.1 ± 0.2	87.6 ± 0.2
A-*SVD*-SDAE (ε)	88.1 ± 0.2 (0.5)	*89.5 ± 0.1 (0.5)*	*88.7 ± 0.2 (0.5)*
A-*SVD*-SDAE (ε, α)	**91.1 ± 0.2 (0.75, 0.7)**	*89.4 ± 0.3 (0.5, 0.7)*	*88.9 ± 0.3 (0.75, 0.6)*
A-*MICE*	89.2 ± 0.3	88.4 ± 0.1	88.8 ± 0.3
A-*MICE*-SDAE	90.1 ± 0.3	88.5 ± 0.3	89.9 ± 0.2
A-*MICE*-SDAE (ε)	**90.7 ± 0.4 (0.75)**	**90.1 ± 0.2 (0.5)**	**90.3 ± 0.1 (0.5)**
A-*MICE*-SDAE (ε, α)	**91.3 ± 0.2 (0.75, 0.7)**	**90.4 ± 0.3 (0.75, 0.4)**	**90.1 ± 0. 3(0.5, 0.7)**

Different percentages of missing values in the input dataset are shown in columns. Several procedures are presented in rows according to the following nomenclature: first initial of the dataset—*pre-imputation method*—deep procedure *(deletion rate, compensation rate)*. For each dataset, the best result of a family of pre-imputations is shown in italic and the global minimum for a missing percentage in bold. Results without statistically difference are considered ties

Table 4 Effects of compensation on deletion in classification tasks

Dataset	Worse	Equal	Better
MAGIC Gamma telescope (M)	0	5	4
Pima Indians diabetes (P)	1	3	5
Two norms (T)	0	6	3
Sensorless drive diagnosis (S)	0	4	5
Gas sensor array drift (G)	0	5	4
AReM (A)	0	6	3

Numbers show how many times results with both deletion and compensation are worse, equal or better than those obtained when only deletion is applied. As it can be seen, worse results are achieved only 1 time (2%), improved in 24 cases (45%) and considered statistically equal in 29 cases (53%)

makes more precise imputations when it includes compensation, this is not translated into a better classification with the exception of using powerful pre-imputation techniques, such as MICE. It seems that the improvement is not enough for the network to select the correct class.

To improve the obtained results, the possibility of separately weighting the errors for complete, deleted and incomplete samples is a clear alternative, as well as considering the classification targets in the imputation process. Extending this work to regression tasks is also an interesting research avenue. Finally, the procedure we propose could also be used to deal with sample manipulations in adversarial learning [37, 38], after adapting it to the different situations that appear in this context.

Acknowledgements The work of A. R. Figueiras-Vidal has been partly supported by Grant Macro-ADOBE (TEC 2015-67719-P, MINECO/FEDER&FSE). The work of J.L. Sancho-Gómez has been partly supported by Grant AES 2017 (PI17/00771, MINECO/FEDER).

References

1. Sharpe PK, Solly RJ (1995) Dealing with missing values in neural network-based diagnostic systems. Neural Comput Appl 3(2):73–77. https://doi.org/10.1007/BF01421959
2. Little R, Rubin D (2002) Statistical analysis with missing data, 2nd edn. Wiley, London
3. García-Laencina PJ, Sancho-Gómez JL, Figueiras-Vidal AR (2010) Pattern classification with missing data: a review. Neural Comput Appl 19(2):263–282. https://doi.org/10.1007/s00521-009-0295-6
4. Quinlan JR (1993) C4.5: programs for machine learning. Morgan-Kaufmann, Burlington
5. Lim CP, Leong JH, Kuan MM (2005) A hybrid neural network system for pattern classification tasks with missing features. IEEE Trans Pattern Anal Mach Intell 27:648–653. https://doi.org/10.1109/TPAMI.2005.64
6. Del Castillo PR, Cardeosa J (2012) Fuzzy min–max neural networks for categorical data: application to missing data imputation. Neural Comput Appl 21(6):1349–1362. https://doi.org/10.1007/s00521-011-0574-x
7. Delalleau O, Courville A, Bengio Y (2008) Gaussian mixtures with missing data: an efficient EM training algorithm. In: Proceeding of the computing research association conference, Snowbird, p 155
8. Ghahramani Z, Jordan MI (1994) Supervised learning from incomplete data via an EM approach. In: Cowan JD, Tesauro G, Alspector J (eds) Advances in neural information processing systems, vol 6. Morgan-Kaufmann, Burlington, pp 120–127
9. Zio MD, Guarnera U, Luzi O (2007) Imputation through finite Gaussian mixture models. Comput Stat Data Anal 51(11):5305–5316. https://doi.org/10.1016/j.csda.2006.10.002
10. García-Laencina PJ, Sancho-Gómez JL, Figueiras-Vidal AR, Verleysen M (2009) K nearest neighbours with mutual information for simultaneous classification and missing data imputation. Neurocomputing 72(7–9):1483–1493. https://doi.org/10.1016/j.neucom.2008.11.026
11. Batista GE, Monard MC (2003) An analysis of four missing data treatment methods for supervised learning. Appl Artif Intell 17(5–6):519–533. https://doi.org/10.1080/713827181
12. Troyanskaya O, Cantor M, Sherlock G, Brown P, Hastie T, Tibshirani R, David Botstein D, Altman RB (2001) Missing value estimation methods for DNA microarrays. Bioinformatics 17(6):520–525
13. Fessant F, Midenet S (2002) Self-organising map for data imputation and correction in surveys. Neural Comput Appl 10(4):300–310. https://doi.org/10.1007/s005210200002
14. Peng H, Zhu S (2007) Handling of incomplete data sets using ICA and SOM in data mining. Neural Comput Appl 16(2):167–172. https://doi.org/10.1007/s00521-006-0058-6
15. Latif BA, Mercier G (2010) Self-organizing maps. https://doi.org/10.5772/9178
16. Gupta A, Lam MS (1996) Estimating missing values using neural networks. J Oper Res Soc 47:229–238. https://doi.org/10.2307/2584344
17. Nishanth KJ, Ravi V, Ankaiaha N, Bose I (2012) Soft computing based imputation and hybrid data and text mining: the case of predicting the severity of phishing alerts. Expert Syst Appl 39(12):10583–10589. https://doi.org/10.1016/j.eswa.2012.02.138
18. Smola AJ, Vishwanathan SVN, Hofmann T (2005) Kernel methods for missing variables. In: Proceedings of the 10th international workshop on artificial intelligence and statistics, pp 325–332
19. García-Laencina PJ, Sancho-Gómez JL, Figueiras-Vidal AR (2013) Classifying patterns with missing values using multi-task learning perceptrons. Expert Syst Appl 40(4):1333–1341. https://doi.org/10.1016/j.eswa.2012.08.057
20. Bengio Y, Lecun Y (2007) Scaling learning algorithms towards AI. MIT Press, Cambridge
21. Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput Appl 18(7):1527–1554. https://doi.org/10.1162/neco.2006.18.7.1527
22. Bengio Y (2009) Learning deep architectures for AI. Found Trends Mach Learn 2(1):1–127. https://doi.org/10.1561/2200000006
23. Deng L, Yu D (2014) Deep learning: methods and applications. Found Trends Signal Process 7(3–4):197–387. https://doi.org/10.1561/2000000039
24. Beaulieu-Jones BK, Moore JH (2017) Missing data imputation in the electronic health record using deeply learned autoencoders. World Scientific, Singapore, pp 207–218. https://doi.org/10.1142/9789813207813021
25. Gondara L, Wang K (2017) Multiple imputation using deep denoising autoencoders. arXiv:1705.02737v2
26. Sánchez-Morales A, Sancho-Gómez JL, Figueiras-Vidal AR (2017) Values deletion to improve deep imputation processes. In: International work-conference on the interplay between natural and artificial computation, IWINAC 2017, Coruna, pp 240–246. https://doi.org/10.1007/978-3-319-59773-7-25
27. Vincent P, Larochelle H, Bengio Y, Manzagol PA (2008) Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on machine learning, ICML'08. ACM, New York, pp 1096–1103. https://doi.org/10.1145/1390156.1390294
28. Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol PA (2010) Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res 11:3371–3408
29. Alvear-Sandoval RF, Figueiras-Vidal AR (2018) On building ensembles of stacked denoising auto-encoding classifiers and their further improvement. Inf Fusion 39:41–52. https://doi.org/10.1016/j.inffus.2017.03.008
30. Little RJA, Rubin DB (1986) Statistical analysis with missing data. Wiley, London
31. Schafer JL (1997) Analysis of incomplete multivariate data. Chapman & Hall, London
32. Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml
33. Delve: data for evaluating learning in valid experiments. https://www.cs.toronto.edu/~delve/data/datasets.html
34. Schmitt P, Mandel J, Guedj M (2015) A comparison of six methods for missing data imputation. J Biomet Biostat 6:224. https://doi.org/10.4172/2155-6180.1000224
35. Azur MJ, Stuart EA, Frangakis C, Leaf PJ (2011) Multiple imputation by chained equations: what is it and how does it work? Int J Methods Psychiatr Res 20(1):40–49. https://doi.org/10.1002/mpr.329
36. Brahma PP, Wu D, She Y (2016) Why deep learning works: a manifold disentanglement perspective. IEEE Trans Neural Netw Learn Syst 27(10):1997–2008. https://doi.org/10.1109/tnnls.2015.2496947
37. Goodfellow I, McDaniel P, Papernot N (2018) Making machine learning robust against adversarial inputs. Commun ACM 61(6):56–66. https://doi.org/10.1145/3134599
38. Vorobeychik Y, Kantarcioglu M (2018) Adversarial machine learning. Synth Lect Artif Intell Mach Learn 12(3):1–169