Neighborhood-aware autoencoder for missing value imputation

Helena Aidos
LASIGE, Faculdade de Ciências,
Universidade de Lisboa
Lisbon, Portugal
haidos@ciencias.ulisboa.pt

Pedro Tomás
INESC-ID, Instituto Superior Técnico,
Universidade de Lisboa
Lisbon, Portugal
pedro.tomas@inesc-id.pt

***Abstract*—Missing values are a fundamental issue in many applications by constraining the application of different learning methods or by impairing the attained results. Many solutions have been proposed by relying on statistical or machine learning techniques. However, in most cases, the results are not yet satisfactory. Hence, motivated by the advent of deep learning, different solutions have also been proposed, such as by adopting autoencoders and adversarial training. However, in most of these solutions, the results are impaired by the network structure and training strategy, constraining the accuracy of missing value imputation. In this paper, we revisit autoencoder networks and show that through a careful selection of network structure and optimization strategy we outperform other deep learning solutions. We further study the impact of a previously proposed technique, stochastic corruption of inputs, to show that when the network is well designed and trained, it actually impairs the results.**

***Index Terms*—Missing values, deep learning, autoencoder architecture**

I. Introduction

Many real-world datasets contain missing values (e.g., medical data [1]–[3], social network [4], microarray and gene expression data [5]), which are a result of manual data entries, incorrect measurements, equipment errors, among others. The missing values problem is a major concern in many fields. In particular, most machine learning techniques are not well prepared to deal with missing data and require some sort of imputation to allow its application, or the deletion of the missing records. Furthermore, observations with missing values may have a significant impact on predictive analysis, as well as on descriptive and inferential statistics.

Hence, it is imperative to handle missing data either by listwise deletion or by replacing the missing values with estimations from the observed data via imputation procedures. However, ignoring observations with missing values (listwise deletion) may produce biased estimates of means, biased predictive results and standard errors that are correct for a reduced subset of the data, but are often larger with respect to all available data [6]. Therefore, the replacement of missing values through an imputation technique is a better solution.

Along the years, many solutions have been proposed, such as by replacing missing values with simple statistics (e.g., mean or mode); by adopting regression methods [7]; by relying on machine learning techniques (e.g., k-nearest neighbor [8], decision trees [9], dynamic programming [10], fuzzy c-means and genetic algorithm [11]); and more recently through deep learning (e.g., [12]–[14]). Missing values imputation based on deep learning have been proposed following different strategies, namely based on generative adversarial networks [14]–[16], autoencoders [13], [17]–[20], and variational autoencoders [21], [22]. For instance, Collaborative Generative Adversarial Network [14] has been proposed to tackle the problem of missing image data imputation. Datawig [12] was proposed to deal with non-numerical data, including unstructured text and categorical data, by using a long short-term memory, but requires the specification of the imputed attributes and the attributes that will be used to impute the missing values. GAIN [16] is based on generative adversarial networks, where the generator imputes the missing values conditioned to the observed values, while the discriminator determines which values were actually observed and which were imputed. Finally, MIDA [13] and ODAE [20] are both based on denoising autoencoders, where the inputs of the network are corrupted by setting random inputs to zero or adding random noise. However, despite the recent success in deep learning approaches, the proposed schemes often focus solely on problem formulation disregarding the impact of careful optimization of network parameters as well as on the optimization strategies.

In this paper we revisit autoencoder networks and show that through careful selection of network parameters, we attain state-of-the-art results. Hence, our solution is based on an autoencoder architecture that learns to minimize the error between the observed input and the reconstructed output (excluding missing values). Since neural networks layers do not naturally handle missing values, we introduce an input layer that takes into consideration the missing data point local neighborhood. It contrasts with other approaches which initially assume an imputed value of zero or of the feature mean. Hence, during training, the encoding part learns a latent

This work was partially supported by Portuguese national funds through Fundação para a Ciência e a Tecnologia (FCT) under project HAnDLE, ref. PTDC/EEI-HAC/30485/2017, the INESC-ID Research Unit, ref. UIDB/50021/2020, and the LASIGE Research Unit, ref. UIDB/00408/2020 and ref. UIDP/00408/2020.

space that is able to correctly encode the data samples, despite the existence of incomplete data. The decoding network learns to reproduce the samples based on the latent space. Together, they also extract information from the missing pattern, in order to guide the data imputation process.

In summary the contributions of our work are the following:

- We present an autoencoder architecture that, although simpler than other alternatives, provides more accurate imputations;
- We introduce an input layer that takes into consideration the local neighborhood to initialize the neural network function;
- We show that a state-of-the-art technique, stochastic corruption of the inputs [13], [20], do not provide significant benefits once the neural network is properly defined and optimized.

To validate the proposed approach we rely on five synthetic datasets of varying complexities. Each of these datasets describes different combinations of linear or non-linear functions, such as: affine, polynomial, rational, exponential, logarithmic and trigonometric functions. We also use five real datasets often used by state-of-the-art competitive approaches. To model the missing patterns, we rely on three common approaches [23]: missing completely at random (MCAR), missing not at random (MNAR) and missing at random (MAR). Results show that the proposed technique is simple, yet effective, providing better results than novel state-of-the-art deep learning approaches.

II. Neighborhood-aware imputation

A. *Problem Formulation*

Consider a dataset $\boldsymbol{y} = \{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n\} \in \mathcal{Y}^n$, with $\mathcal{Y}$ representing the space to which each data sample belongs (e.g., $\mathbb{R}^L$, with L representing the dimensionality of the data sample). Each data sample $\boldsymbol{y}_i$, is assumed to be extracted from an unknown process $\boldsymbol{f}$, which transforms an unknown (hidden) state $\boldsymbol{s}_i \in \boldsymbol{\mathcal{S}}$ to the observed sample $\boldsymbol{y}_i$, i.e., $\boldsymbol{f} : \mathcal{S} \rightarrow \mathcal{Y}$. For example, in a medical database, $\boldsymbol{s}_i$ represents the patient state, whereas $\boldsymbol{y}_i = [y_i^1, \cdots, y_i^L]$ are the observed variables by performing a series of medical exams. Unfortunately, the application of the transformation process $\boldsymbol{f}$ results in information losses, with an associated pattern of missing values $\boldsymbol{m}_i \in \{0, 1\}^L$ (e.g., exams not performed, exams only made to specific group(s) of patients, patients unable to complete exams). We assume that the missing pattern $\boldsymbol{m}_i$ is a result from a process $g : \mathcal{G} \rightarrow \{0, 1\}^L$ that randomly introduces information losses (MCAR), generates missing values given some pattern of the unknown state $\boldsymbol{s}_i$ (MAR), and/or drops information depending on the observed state $\boldsymbol{y}_i$ (MNAR).

To recover the information lost by sample $\boldsymbol{y}_i$ (through a missing pattern $\boldsymbol{m}_i$), we need (i) to recover the hidden state $\boldsymbol{s}_i$, by learning an inverse transformation $\boldsymbol{f}^{-1}$ as well as the influence of the missing pattern generator $\boldsymbol{g}$; and (ii) learn the transformation $\boldsymbol{f}$ to obtain the unknown values.

However, since the transformation $\boldsymbol{f}$ is unkown and generally impossible to estimate, we rely on an encoding network $\boldsymbol{E_\theta}$ to obtain an alternative approximate representation of $\boldsymbol{s}_i$, which we represent by $\boldsymbol{z}_i$. The objective of the encoding network is thus not only to estimate a latent space that correctly encodes the partially observed variable $\boldsymbol{y}_i$, but also the influence of the missing pattern generator $\boldsymbol{g}$. Naturally, since the objective is to obtain the missing information, we use a decoding network $\boldsymbol{D_\theta}$ to transform the latent space $\boldsymbol{z}_i$ into an observed estimated variable $\tilde{\boldsymbol{y}}_i$. Then (not in Fig. 1) we 11use the known information to obtain a better estimate of data sample $\boldsymbol{y}_i$, obtaining $\hat{\boldsymbol{y}}_i = \tilde{\boldsymbol{y}}_i \circ \boldsymbol{m}_i + \boldsymbol{y}_i \circ (\boldsymbol{1} - \boldsymbol{m}_i)$, with $\circ$ representing the Hadamard product and $m_i^k = 1$ indicating that variable k of data sample $\boldsymbol{y}_i$ (i.e., y_i^k) is missing.

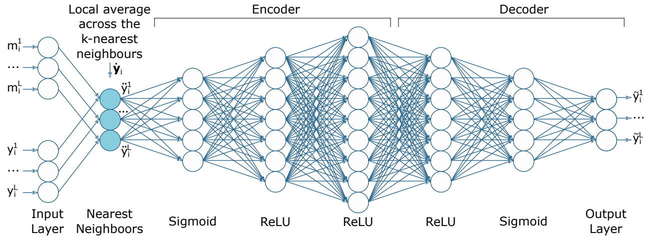

Fig. 1. Overview of the devised autoencoder architecture.

B. *Network architecture*

To obtain a latent space that is able to capture both the sample generation process $\boldsymbol{f}$ and the missing imputation pattern $\boldsymbol{g}$ we use fully-connected networks in an autoencoder structure (see Fig. 1), modified to take into account the data sample $\boldsymbol{y}_i$ feature values, its missing pattern $\boldsymbol{m}_i$, as well as the values of the known neighbours. Hence, for each missing feature j, an input value $\dot{y}_i^j$ is taken by considering the local average of its k-nearest neighbours (see Fig. 1) across the complete training set (Euclidean distance over the observed variables). The encoding part of the proposed deep neural network then takes the computed estimates $\ddot{\boldsymbol{y}}_i = \dot{\boldsymbol{y}}_i \circ \boldsymbol{m}_i + \boldsymbol{y}_i \circ (\boldsymbol{1} - \boldsymbol{m}_i)$ and constructs the latent space representation. For this, we adopt a autoencoder architecture where the number of neurons in successive encoding layers increases by α at each new layer. On the decoder side, we mirror the structure of the encoder, successively decreasing the number of neurons by α at each new layer. In this work we set $\alpha = L$. To correctly encode non-linear transformations, we use sigmoid activations at the first and last hidden layers of the encoder and decoder, respectively, and ReLU activations elsewhere. Although alternative solutions can be adopted, we found no significant difference between ReLU and SeLU (the observed root mean squared error differences were found to lie within the standard deviation range of the network). We also concluded that replacing the sigmoid by other activation functions (tanh, softsign) yielded worst results. Finally replacing any of the ReLU functions with a sigmoid (or an alternative function) results in a degradation of the quality of the imputation process.

C. *Model training*

To train our model, we use an Adam optimizer to minimize the mean squared error between the known variables in each batch of samples $\{\boldsymbol{y}_j, \boldsymbol{y}_{j+1}, \cdots\}$ and the estimated samples $\{\tilde{\boldsymbol{y}}_j, \tilde{\boldsymbol{y}}_{j+1}, \cdots\}$, i.e.,

$$\text{Loss} = \sum_j (\mathbf{1} - \boldsymbol{m}_j)^T ((\boldsymbol{y}_j - \tilde{\boldsymbol{y}}_j) \circ (\boldsymbol{y}_j - \tilde{\boldsymbol{y}}_j)) \ .$$

III. Experimental setup

A. Datasets

The experiments were conducted in two ways: synthetic datasets with different characteristics and increasing difficulty, and real-world datasets from the UCI Machine Learning Repository[1].

Each synthetic dataset is composed of 5000 samples with 12 attributes. Four of the 12 attributes correspond to random values extracted from a uniform distribution (range $[0, 1]$) and the remaining result from linear or non-linear transformations of the other attributes (including the random values). The datasets describe increasingly complex functions, such as:

- *Affine*: linear combinations of attributes;
- *Exponential* and *Logarithm*: application of exponential and logarithmic functions (respectively) over linear combinations of attributes;
- *Rational*: application of multiplications, divisions and polynomial functions of attributes to introduce high order effects in the transformation functions;
- *Trigonometric*: application of trigonometric functions, to evaluate the use of non-injective mappings that do not provide an exact inverse.

The used real world datasets correspond to breast cancer (683 observations / 9 attributes), glass (214/9), vehicle (846/16), satellite (6435/36) and letter recognition (20000/16).

To provide a common evaluation ground for all methods and guarantee that errors in different features have similar importance, we normalize all features to the range $[0, 1]$.

B. Imputation procedure

To describe how datasets are influenced by missing values, three different mechanisms are usually found in the literature [23], namely: (i) *Missing Completely At Random* (MCAR), where the probability of being missing do not depend on any variables in the data; (ii) *Missing At Random* (MAR), where the missing probability is independent on the actual observed data and is the same within groups of the observed data, but different across groups; and, (iii) *Missing Not At Random* (MNAR), where the missing mechanism is neither MCAR nor MAR. Hence, the existence of missing values is dependent on the hypothetical value of the data.

A multivariate missing data procedure [24] was applied to each dataset to generate missing values in multiple variables, with different missing proportions and underlying missingness mechanism. This procedure is implemented in R in the package called `ampute`. In this paper, missing proportions from 5% to 40%, with 5% increment, and all three missingness mechanisms (MCAR, MAR, and MNAR) were considered.

C. Alternative state-of-the-art approaches

The proposed imputation method will be compared with several state-of-the-art imputation methods, namely: mean, k-nearest neighbor (k-NN, with $k = 5$), least square (LS), MIDA [13], and GAIN[2] [16]. Median imputation and stochastic regression were also applied to the datasets, however the results were similar to the mean and LS imputations, respectively, and were therefore left out from the presented results. The proposed imputation method was implemented in Python 3.6 using TensorFlow 2.0. The state-of-the-art methods rely on standard Python packages, including `autoimpute`[3], or on specific Tensorflow implementations. Each experiment was conducted 10 times and within each a 5-fold cross-validation strategy was employed, where the data was split in train (80% of the data) and test (20%). The average root mean squared error (RMSE) was used to validate the imputation values and was computed over the missing values, by comparing the true observed values with the values imputed by each of the above mentioned approaches.

IV. Results

Figure 2 presents the results of the synthetic datasets for all the considered imputation methods, using an MCAR missigness mechanism (due to space constraints, the results for the MAR and MNAR mechanisms are only shown in Table I for a 20% missing values proportion). By analyzing this figure it can be observed that in the affine dataset, the LS is the second or third best imputation method (depends of the missingness proportion), since it correctly assumes a linear model between attributes. Nonetheless, the proposed approach significantly outperforms its results, attaining a lower RMSE.

Exponential, rational and trigonometric datasets introduce non-linear combinations of attributes, thus significantly requiring the use of non-linear methods. Hence, the best methods are the proposed one, GAIN and 5-NN, with the proposed one significantly outperforming the other two. The worst methods are LS and mean, since they model missing values through an often incorrect linear model and a constant value, respectively. In particular, in the exponential and trigonometric datasets, LS presents a higher RMSE value compared to the remaining methods, for all missingness proportions.

Finally, in the logarithm dataset, the proposed method presents the lowest score, followed by 5-NN and LS. Although the results of LS might seem surprising, they are a result of a transformation that reduces the dynamic range of variables (regarding the previous cases), thus allowing a linear approximation to make good predictions. Also, GAIN presents the worst in this dataset, especially for lower missingness proportions, where it attains relatively higher RMSE values.

To evaluate the impact of the other missigness mechanisms (MAR and MNAR), Table I presents the results for 20% of missing values (similar conclusions can be drawn for other

[1] https://archive.ics.uci.edu/ml/index.php

[2] The code used for this algorithm was downloaded from the authors github account: https://github.com/jsyoon0823/GAIN

[3] https://pypi.org/project/autoimpute/

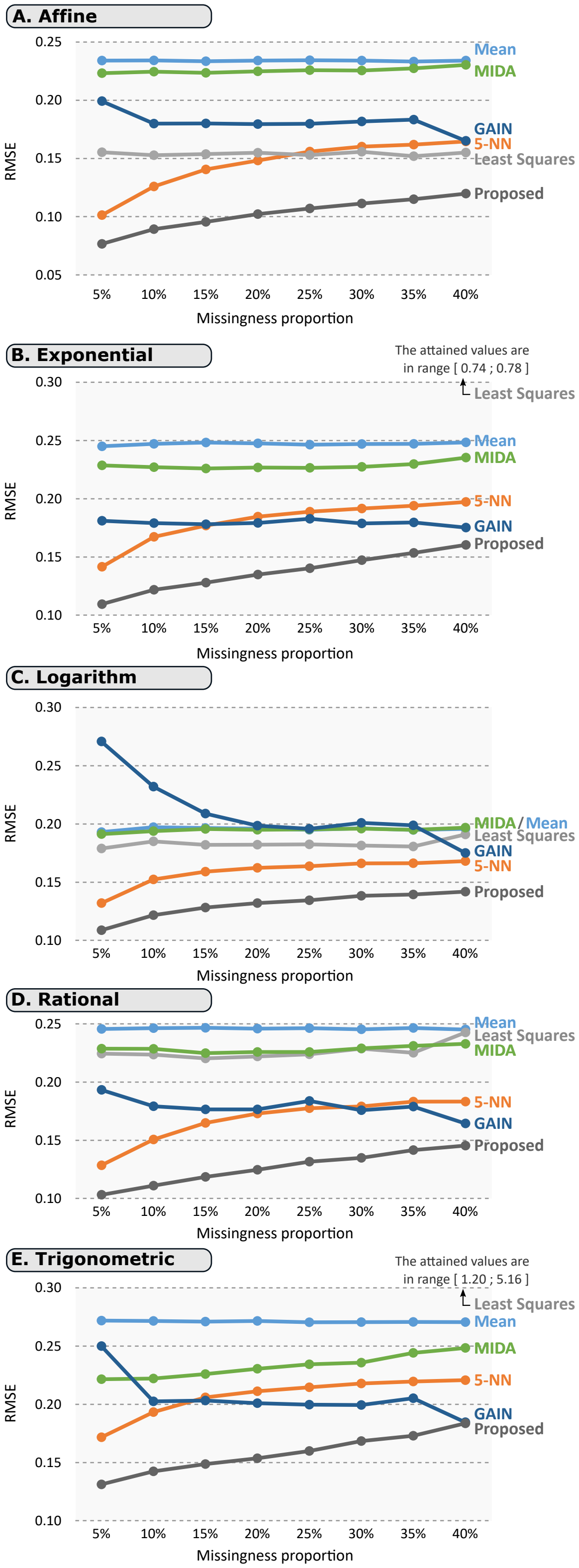

Fig. 2. Evaluation of the proposed imputation method for the synthetic datasets in a MCAR strategy.

missingness proportions). As can be seen, the RMSE for each method is similar for all imputation mechanism, except for MIDA which globally performs better in an MNAR scenario.

TABLE I

AVERAGE ROOT MEAN SQUARED ERROR (± STANDARD DEVIATION) OF THE PROPOSED IMPUTATION STRATEGY AND THE STATE-OF-THE-ART APPROACHES FOR DIFFERENT MISSING MECHANISMS (ROWS) AND 20% OF MISSSINGNESS PROPORTION, IN ALL FIVE SYNTHETIC DATASETS.

	Mean	5-NN	LS	MIDA	GAIN	Proposed
			AFFINE DATASET			
MCAR	0.234 (± 0.003)	0.148 (± 0.004)	0.155 (± 0.006)	0.225 (± 0.006)	0.179 (± 0.015)	**0.102** (± 0.004)
MAR	0.235 (± 0.003)	0.152 (± 0.003)	0.150 (± 0.005)	0.214 (± 0.007)	0.186 (± 0.018)	**0.105** (± 0.004)
MNAR	0.236 (± 0.003)	0.148 (± 0.004)	0.145 (± 0.005)	0.204 (± 0.005)	0.181 (± 0.015)	**0.101** (± 0.004)
			EXPONENTIAL DATASET			
MCAR	0.238 (± 0.003)	0.185 (± 0.004)	0.757 (± 0.029)	0.227 (± 0.007)	0.179 (± 0.014)	**0.135** (± 0.005)
MAR	0.255 (± 0.004)	0.187 (± 0.004)	0.797 (± 0.038)	0.218 (± 0.008)	0.183 (± 0.014)	**0.134** (± 0.007)
MNAR	0.262 (± 0.003)	0.186 (± 0.004)	0.796 (± 0.036)	0.213 (± 0.008)	0.191 (± 0.014)	**0.136** (± 0.006)
			LOGARITHM DATASET			
MCAR	0.196 (± 0.003)	0.162 (± 0.003)	0.182 (± 0.008)	0.195 (± 0.004)	0.199 (± 0.014)	**0.132** (± 0.004)
MAR	0.195 (± 0.003)	0.165 (± 0.005)	0.183 (± 0.008)	0.194 (± 0.004)	0.204 (± 0.020)	**0.136** (± 0.004)
MNAR	0.195 (± 0.003)	0.158 (± 0.004)	0.175 (± 0.009)	0.189 (± 0.004)	0.199 (± 0.025)	**0.127** (± 0.004)
			RATIONAL DATASET			
MCAR	0.246 (± 0.003)	0.173 (± 0.005)	0.222 (± 0.007)	0.226 (± 0.005)	0.176 (± 0.014)	**0.125** (± 0.006)
MAR	0.254 (± 0.004)	0.171 (± 0.004)	0.235 (± 0.008)	0.218 (± 0.007)	0.190 (± 0.018)	**0.123** (± 0.007)
MNAR	0.258 (± 0.004)	0.172 (± 0.004)	0.234 (± 0.008)	0.208 (± 0.007)	0.186 (± 0.017)	**0.123** (± 0.005)
			TRIGONOMETRIC DATASET			
MCAR	0.272 (± 0.004)	0.211 (± 0.004)	1.456 (± 0.435)	0.230 (± 0.006)	0.201 (± 0.020)	**0.154** (± 0.009)
MAR	0.271 (± 0.004)	0.214 (± 0.005)	1.522 (± 0.299)	0.233 (± 0.007)	0.202 (± 0.017)	**0.157** (± 0.010)
MNAR	0.269 (± 0.004)	0.212 (± 0.004)	1.553 (± 0.495)	0.219 (± 0.007)	0.196 (± 0.013)	**0.155** (± 0.009)

Also, all methods have a lower variability (lower standard deviation values), except LS in highly non-linear datasets (exponential and trigonometric) and GAIN which presents standard deviations 10x higher than the remaining methods (a consequence of the use of adversarial training). In general the best solutions amongst the state-of-the-art approaches are the linear models for affine dataset and GAIN for the remaining cases. However, the proposed imputation method attains significantly better RMSE values for all datasets and all three imputation mechanisms.

Finally, the proposed imputation method is compared with the remaining deep learning approaches (MIDA and GAIN) on real datasets. Figure 3 presents the results when amputing each dataset with a MCAR mechanism and 20% of missingness proportion. As can be seen, the proposed imputation method outperforms the remaining deep learning strategies, with MIDA attaining the worst results. Although not presented due to space limitations, similar conclusions were obtained for the MAR and MNAR missingness mechanisms.

V. DISCUSSION

Although MIDA [13] closely resembles the proposed approach, it attains significantly worst results (approximately half of the RMSE in almost all datasets). Our results suggest that this is due to multiple deficiencies in MIDA, namely: (1) MIDA uses the Nesterov's accelerated gradient, instead

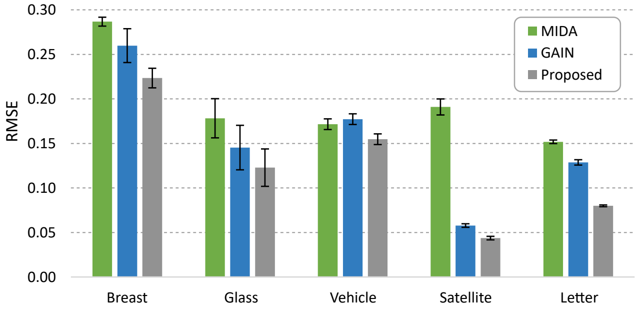

Fig. 3. Evaluation of the proposed imputation method for the real datasets in a MCAR strategy and for 20% of missingness proportion.

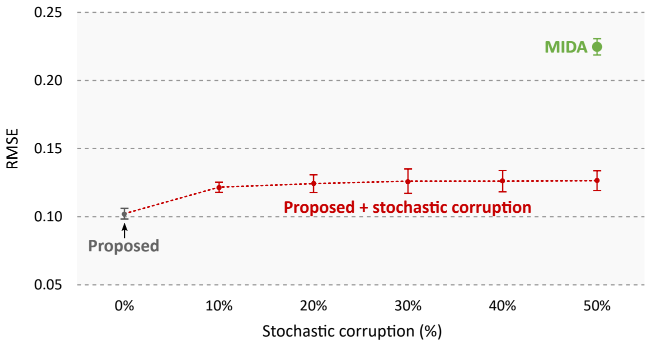

Fig. 4. Evaluation of the use of stochastic corruption over the proposed imputation method. Average RMSE and standard deviation for the affine dataset in a MCAR strategy and for 20% of missingness proportion.

of ADAM, which allows a significantly better tuning of the network parameters; (2) we use a better choice on the activation functions (a combination of sigmoids and ReLUs), whereas they solely use the tanh function, which provides worst results and is inconsistent with the data normalization applied by the authors (range $[0, 1]$); (3) the use of the feature mean value as input for the neural network, which in our case would result in an approximate 60% increase in RMSE value (a similar problem found in the method by [19]); and (4) the use of stochastic corruption of the inputs (also used by [19] and [20]), which sets a percentage of the inputs to zero.

In what concerns this last case, figure 4 presents the results of adding stochastic corruption to the inputs of proposed method. As can be observed adding even a small percentage of corruption leads to a significant degradation of the results (up to 24%).

VI. Conclusions

A deep learning approach to the missing value imputation problem is herein proposed. Although based on a simpler approach than competitive deep learning solutions, it relies on careful tuning of network parameters and training methodology, and on the introduction of an input layer that takes into consideration the k-nearest neighbours of each incomplete sample. As a consequence it attains competitive results against state-of-the-art methods, such as those based on deep learning.

References

[1] Yelipe, Porika, and Golla, “An efficient approach for imputation and classification of medical data values using class-based clustering of medical records,” *Computers and Electrical Engineering*, vol. 66, pp. 487–504, 2018.

[2] Lemos, Silva *et al.*, “Discriminating alzheimer’s disease from mild cognitive impairment using neuropsychological data,” *Age (M±SD)*, vol. 70, no. 8.4, pp. 73–3, 2012.

[3] Carreiro, Amaral *et al.*, “Prognostic models based on patient snapshots and time windows: Predicting disease progression to assisted ventilation in amyotrophic lateral sclerosis,” *Journal of biomedical informatics*, vol. 58, pp. 133–144, 2015.

[4] Krause, Huisman *et al.*, “Missing network data a comparison of different imputation methods,” in *IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*, 2018, pp. 159–163.

[5] Souto, Jaskowiak, and Costa, “Impact of missing data imputation methods on gene expression clustering and classification,” *BMC Bioinformatics*, vol. 16, no. 1, pp. 1–9, 2015.

[6] Pepinsky, “A note on listwise deletion versus multiple imputation,” *Political Analysis*, vol. 26, no. 4, pp. 480–488, 2018.

[7] García-Laencina, Sancho-Gómez, and Figueiras-Vidal, “Pattern classification with missing data: A review,” *Neural Computing and Applications*, vol. 19, no. 2, pp. 263–282, 2010.

[8] “Nearest neighbor imputation algorithms: A critical evaluation,” *BMC Medical Informatics and Decision Making*, vol. 16, pp. 197–208, 2016.

[9] Rahman and Islam, “Missing value imputation using decision trees and decision forests by splitting and merging records: Two novel techniques,” *Knowledge-Based Systems*, vol. 53, pp. 51–65, 2013.

[10] Nelwamondo, Golding, and Marwala, “A dynamic programming approach to missing data estimation using neural networks,” *Information Sciences*, vol. 237, pp. 49–58, 2013.

[11] Aydilek and Arslan, “A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm,” *Information Sciences*, vol. 233, pp. 25–35, 2013.

[12] Biessmann, Salinas *et al.*, “Deep learning for missing value imputation in tables with non-numerical data,” in *Int. Conf. on Information and Knowledge Management (CIKM)*, 2018, pp. 2017–2026.

[13] Gondara and Wang, “MIDA: Multiple imputation using denoising autoencoders,” in *Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD)*, 2018, pp. 260–272.

[14] Lee, Kim *et al.*, “CollaGAN : Collaborative GAN for Missing Image Data Imputation,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2487–2496.

[15] Li, Jiang, and Marlin, “MisGAN: Learning from Incomplete Data with Generative Adversarial Networks,” in *Int. Conf. on Learning Representations (ICLR)*, 2019, pp. 1–12.

[16] Yoon, Jordon, and Van Der Schaar, “GAIN: Missing data imputation using generative adversarial nets,” in *Int. Conf. on Machine Learning (ICML)*, vol. 13, 2018, pp. 9042–9051.

[17] Costa, Santos *et al.*, “Missing Data Imputation via Denoising Autoencoders: The Untold Story,” in *Int. Symposium on Intelligent Data Analysis (IDA)*, vol. 11191, no. November, 2018, pp. 87–98.

[18] Mattei and Freiisen, “MIWAE: Deep generative modelling and imputation of incomplete data sets,” in *Int. Conf. on Machine Learning (ICML)*, 2019, pp. 7762–7772.

[19] Abiri, Linse *et al.*, “Establishing strong imputation performance of a denoising autoencoder in a wide range of missing data problems,” *Neurocomputing*, vol. 365, pp. 137–146, 2019.

[20] Phung, Kumar, and Kim, “A deep learning technique for imputing missing healthcare data,” in *Int. Conf. of the IEEE Engineering in Medicine and Biology Society (EMBS)*. IEEE, 2019, pp. 6513–6516.

[21] McCoy, Kroon, and Auret, “Variational Autoencoders for Missing Data Imputation with Application to a Simulated Milling Circuit,” in *IFAC Workshop on Mining, Mineral and Metal Processing (MMM)*, vol. 51, no. 21. Elsevier B.V., 2018, pp. 141–146.

[22] Camino, Hammerschmidt, and State, “Improving Missing Data Imputation with Deep Generative Models,” *arXiv preprint*, p. arXiv:1902.10666, 2019.

[23] Van Buuren, *Flexible imputation of missing data*. Chapman and Hall/CRC, 2018.

[24] Schouten, Lugtig, and Vink, “Generating missing values for simulation purposes: a multivariate amputation procedure,” *Journal of Statistical Computation and Simulation*, vol. 88, no. 15, pp. 2909–2930, 2018.