Missing Features Reconstruction Using a Wasserstein Generative Adversarial Imputation Network

Magda Friedjungová$^{1[0000-0002-3363-294X]}$, Daniel Vašata$^{1[0000-0003-0616-4340]}$, Maksym Balatsko$^{1[0000-0001-7573-0188]}$, and Marcel Jiřina$^{1[0000-0002-6686-1511]}$

Faculty of Information Technology, Czech Technical University in Prague
Prague, Czech Republic
{magda.friedjungova, daniel.vasata, balatmak, marcel.jirina}@fit.cvut.cz

Abstract. Missing data is one of the most common preprocessing problems. In this paper, we experimentally research the use of generative and non-generative models for feature reconstruction. Variational Autoencoder with Arbitrary Conditioning (VAEAC) and Generative Adversarial Imputation Network (GAIN) were researched as representatives of generative models, while the denoising autoencoder (DAE) represented non-generative models. Performance of the models is compared to traditional methods k-nearest neighbors (k-NN) and Multiple Imputation by Chained Equations (MICE). Moreover, we introduce WGAIN as the Wasserstein modification of GAIN, which turns out to be the best imputation model when the degree of missingness is less than or equal to 30%. Experiments were performed on real-world and artificial datasets with continuous features where different percentages of features, varying from 10% to 50%, were missing. Evaluation of algorithms was done by measuring the accuracy of the classification model previously trained on the uncorrupted dataset. The results show that GAIN and especially WGAIN are the best imputers regardless of the conditions. In general, they outperform or are comparative to MICE, k-NN, DAE, and VAEAC.

Keywords: Imputation Methods · Feature Reconstruction · Missing Data · Generative Models · Autoencoders · Wasserstein GAN

1 Introduction

When working with real-world datasets one of the standard problems that needs solving as part of the data preprocessing phase is dealing with missing data. The missingness can be represented by either individual missing data randomly located in instances or by the absence of entire features.

To our best knowledge, not much attention is paid to the second scenario where entire features are missing, i.e., there are no clear answers to questions such as how to face the situation, how the standard imputation method will perform or if there is a need to approach this challenge in a different way.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50423-6_17

The aim of our work is to study these issues by experimentally comparing several state-of-the art imputation methods in real-world scenarios where one needs to impute (i.e., reconstruct) entire features. This work follows up on our previous work presented in paper [12], where we focus on the comparison of traditional (k-NN, linear regression, MICE) and modern (multi-layer perceptron, extreme gradient boosted trees) imputation methods.

In the current paper, we research more universal imputers represented by autoencoders and generative neural network models. These models have a common advantage in that one does not need to know which features are missing in advance. On the contrary, regular imputation methods need to be trained for each combination of missing features separately. A typical example where a universal imputer is needed is the prediction of a classification model from sensor data, where a sensor breakdown leads to missing data in one or more features. Usually, the prediction model itself is not able to handle this situation without a significant decrease in its performance. Furthermore, one typically does not know in advance which sensor is going to be broken. The best approach would be to retrain the model using data without missing features. However, in a production setting model retraining is impossible as the existing model needs to respond to corrupted data immediately.

We consider a situation where the prediction model is trained on a complete preprocessed dataset with numeric features, and we study its accuracy changes on new unseen data with imputed missing features. The amount of missing data (i.e. features) varies between 10% and 50%. Experiments are performed on ten real and two artificial datasets. The impact of imputation is measured as the classification accuracy change of the best performing from six commonly used classification models: logistic regression, multi-layer perceptron, k-NN, naive Bayes, extreme gradient boosted trees [7], and random forest. Besides accuracy we also use root mean squared error (RMSE) (which was also used in [35,6,17]) as a measure of the quality of the imputation.

We compare the denoising autoencoder (DAE) [33], Generative Adversarial Imputation Network (GAIN) [35], and Variational Autoencoder with Arbitrary Conditioning (VAEAC) [17] with k-NN and MICE [4], which are considered to be successful traditional imputation methods. Moreover, we introduce Wasserstein Generative Adversarial Imputation Network (WGAIN), a Wasserstein based modification of GAIN, see [2]. WGAIN is a generative imputation model and generally outperforms other presented models on the tested datasets. The Earth-Mover distance and the corresponding discriminator's critic of the Wasserstein approach do not suffer from vanishing gradients in the way that a vanilla GAN would. This enables the model to capture the desired distribution better.

The paper is organized as follows. In Section 2, we briefly review related work in this field. In Section 3 the WGAIN model is introduced. Section 4 is devoted to the description of experiments performed, including the evaluation of their results. Finally, the paper is concluded in Section 5.

2 Related Work

There are many traditional imputation methods, such as e.g., [11,24,32]. Some of the most common and successful are k-nearest neighbors imputation (k-NN) [18] and multivariate imputation by chained equations (MICE) [29,32].

Approaches based on deep learning have been under active development for the last few years. They use many variants of neural networks starting from multi-layer perceptron, e.g., in [30,3]. A more advanced approach is based on the autoencoder as a specific kind of neural network aiming to reconstruct inputs on its outputs. Here, one of the most commonly used models is the denoising autoencoder (DAE) [33], e.g., [34,10,5,15,8]. Typically, they are used in a discriminative way (see [15] for difference), meaning they impute a single value, which is deterministic once the network is trained.

On the other hand, the most recent research focuses on generative models which enables one to sample from the distribution conditioned on the observed features and thus get information about the uncertainty in imputed values. There are two groups of deep learning generative models. First, there are models based on the variational autoencoder (VAE) [19] and its conditional alternations, see [31,26,36,25]. In this group, some of the most successful imputation models are VAEAC [17] and HI-VAE [27].

The second group contains models based on the Generative Adversarial Network (GAN) [16]. Notably, one can encounter them in image reconstruction tasks (i.e., image inpainting), see [20,22,28]. One of the most prominent methods based on GAN is the GAIN [35], which uses the generator discriminator mechanism to achieve learning of the desired distribution. The generator observes some components of a real data vector, imputes the missing components conditioned on what is observed, and outputs a completed vector. The discriminator then takes a completed vector and attempts to determine which components were observed and which were imputed. The GAIN forms the base for our modification of the imputation method based on Wasserstein GAN [2], which is introduced in the next section. Only recently, GAIN was outperformed by the previously mentioned VAEAC and HI-VAE. However, for numeric variables, HI-VAE achieves a comparable error to the rest of the methods [27]. Therefore we have chosen only VAEAC for the experimental comparison.

3 Wasserstein Generative Adversarial Imputation Network

In this section, the WGAIN model is introduced as GAIN adapting the discriminative approach from Wasserstein GAN.

Let us denote $\mathcal{X} = \mathbb{R}^d$ the d-dimensional numeric data domain and let $\boldsymbol{X} = (X_1, \ldots, X_d)$ be a random vector with values in $\mathcal{X}$ whose distribution is denoted by $\mathrm{P}(\boldsymbol{X})$. Let the mask be a random binary vector $\boldsymbol{M}$, i.e., random vector with values in $\{0, 1\}^d$. The mask corresponds to unobserved values of $\boldsymbol{X}$ so that the value 0 of its jth component means that the jth feature of X_j is missing and the value 1 means that the jth feature of X_j is not missing. The distribution of $\boldsymbol{M}$ corresponds to the distribution of missingness in the data. Let us further denote by $\tilde{\boldsymbol{X}}$ the vector $\boldsymbol{X}$ having zeros in place of missing values given by

$$\tilde{\boldsymbol{X}} = \boldsymbol{X} \odot \boldsymbol{M},$$

where $\odot$ denotes element-wise multiplication. Our aim is to impute missing values in $\tilde{\boldsymbol{X}}$ based on information from non-missing features of $\tilde{\boldsymbol{X}}$ and $\boldsymbol{M}$. It is done in a generative way and it means that we want to learn the conditional distribution $\mathrm{P}(\boldsymbol{X}|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}, \boldsymbol{M} = \boldsymbol{m})$ of $\boldsymbol{X}$ given $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}$ and $\boldsymbol{M} = \boldsymbol{m}$. To do this let $\boldsymbol{Z}$ be a random vector with identically distributed independent components having normal distribution $\mathrm{N}(0, \sigma^2)$ with variance σ^2 and define

$$\tilde{\boldsymbol{X}}_{\boldsymbol{Z}} = \boldsymbol{Z} \odot (1 - \boldsymbol{M}) + \boldsymbol{X} \odot \boldsymbol{M},$$

i.e. $\tilde{\boldsymbol{X}}_{\boldsymbol{Z}}$ is $\tilde{\boldsymbol{X}}$ with missing components replaced by normal random variables.

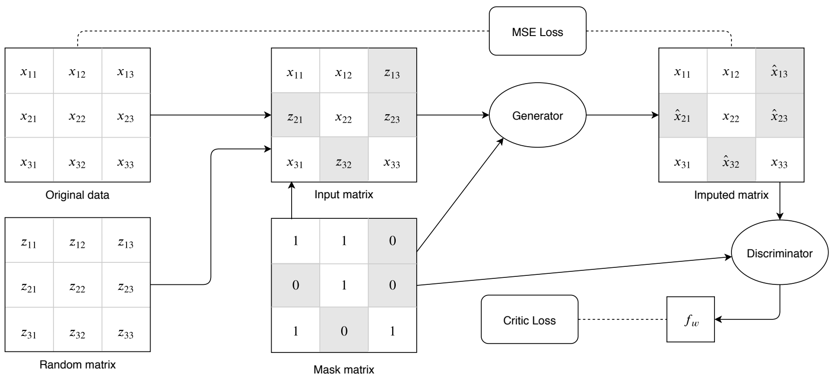
MSE Loss
x_{11} x_{12} x_{13}
x_{21} x_{22} x_{23}
x_{31} x_{32} x_{33}
Original data
x_{11} x_{12} z_{13}
z_{21} x_{22} z_{23}
x_{31} z_{32} x_{33}
Input matrix
Generator
x_{11} x_{12} $\hat{x}_{13}$
$\hat{x}_{21}$ x_{22} $\hat{x}_{23}$
x_{31} $\hat{x}_{32}$ x_{33}
Imputed matrix
z_{11} z_{12} z_{13}
z_{21} z_{22} z_{23}
z_{31} z_{32} z_{33}
Random matrix
1 1 0
0 1 0
1 0 1
Mask matrix
Discriminator
Critic Loss
f_w

Fig. 1. WGAIN structure and mini-batch data flow.

The WGAIN model consists of two parts, the generator g and the critic f, both represented by deep neural networks. The generator g is constructed as a mapping $g : \mathcal{X} \times \{0,1\}^d \to \mathcal{X}$ so that

$$\hat{\boldsymbol{X}}_{\boldsymbol{Z}} = g(\tilde{\boldsymbol{x}}_{\boldsymbol{Z}}, \boldsymbol{m}) \odot (1 - \boldsymbol{m}) + \tilde{\boldsymbol{x}} \odot \boldsymbol{m}$$

is a random vector whose conditional distribution $\mathrm{P}(\hat{\boldsymbol{X}}_{\boldsymbol{Z}}|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}, \boldsymbol{M} = \boldsymbol{m})$, determined by the distribution $\mathrm{P}(\boldsymbol{Z})$ of $\boldsymbol{Z}$, should be close to the conditional distribution $\mathrm{P}(\boldsymbol{X}|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}, \boldsymbol{M} = \boldsymbol{m})$. Note that $g(\tilde{\boldsymbol{x}}_{\boldsymbol{Z}}, \boldsymbol{m})$ is a random vector corresponding to $\tilde{\boldsymbol{x}}$ with all missing components imputed.

In order to train it, we employ the standard squared loss function

$$L_{\mathrm{MSE}}(\hat{\boldsymbol{x}}_{\boldsymbol{z}}, \boldsymbol{x}) = \|\hat{\boldsymbol{x}}_{\boldsymbol{z}} - \boldsymbol{x}\|^2,$$

forcing the output $\hat{\boldsymbol{X}}_{\boldsymbol{Z}}$ to be close to the original data $\boldsymbol{X}$. However, it turns out that this condition alone is not sufficient for learning the proper conditional distribution. To improve the performance of the generator, one may introduce a discriminator trying to find out which components of $\hat{\boldsymbol{X}}_{\boldsymbol{Z}}$ were imputed and use the discriminator for adversarial training. This approach was introduced in [35] and is the base of WGAIN.

In this paper we present a similar way how to improve the conditional distribution of the generator's output. It is based on the Earth-Mover (EM) distance between two probability distributions $\mathrm{P}(X), \mathrm{P}(Y)$ defined by

$$W\big(\mathrm{P}(X), \mathrm{P}(Y)\big) = \inf_{\gamma \in \boldsymbol{\Pi}(\mathrm{P}(X),\mathrm{P}(Y))} \mathrm{E}_{(X,Y)\sim\gamma} \|X - Y\|,$$

where $\boldsymbol{\Pi}(\mathrm{P}(X), \mathrm{P}(Y))$ denotes the set of all joint distributions (X, Y) whose marginals are respectively $\mathrm{P}(X)$ and $\mathrm{P}(Y)$. The term $\mathrm{E}_{(X,Y)\sim\gamma}\|X - Y\|$ might be understood as a measure of how much probability mass has to be transported in order to transform the distributions $\mathrm{P}(X)$ into the distribution $\mathrm{P}(Y)$ when the joint distribution is γ. The EM distance can thus be seen as the cost of the optimal transport plan, see [2] and references therein for more details. The EM distance is usually expressed using the Kantorovich-Rubinstein duality as

$$W\big(\mathrm{P}(X), \mathrm{P}(Y)\big) = \sup_{\|f\|_L \le 1} \mathrm{E}_{X\sim\mathrm{P}(X)} f(X) - \mathrm{E}_{Y\sim\mathrm{P}(Y)} f(Y), \qquad (1)$$

where $\|f\|_L$ means that f is Lipschitz continuous with Lipschitz constant 1 which might be changed to any constant K since it just multiplies $W\big(\mathrm{P}(X), \mathrm{P}(Y)\big)$ by the same constant.

In Wasserstein GAN one approximates (1) by training the neural network $f_{\boldsymbol{w}}$ parametrized with weights $\boldsymbol{w}$ in some compact space $\mathcal{W}$, thus enforcing the Lipschitz continuity. The function $f_{\boldsymbol{w}}$ is called the *critic* and is trained to maximize the expectations difference in (1). For a single dimensional generator g trying to transform random variable Z so that it has the distribution $\mathrm{P}(X)$ one maximizes

$$\max_{\boldsymbol{w}\in\mathcal{W}} \mathrm{E}_{X\sim\mathrm{P}(X)} f_{\boldsymbol{w}}(X) - \mathrm{E}_{Z\sim\mathrm{P}(Z)} f_{\boldsymbol{w}}(g(Z)).$$

In our case we want to minimize the EM distance between $\mathrm{P}(\hat{\boldsymbol{X}}_{\boldsymbol{Z}}|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}, \boldsymbol{M} = \boldsymbol{m})$ and $\mathrm{P}(\boldsymbol{X}|\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}, \boldsymbol{M} = \boldsymbol{m})$. Hence, we take the mask $\boldsymbol{M}$ as the second argument of the critic as additional information to the first argument given by $\boldsymbol{X}$ with correct features behind the mask $\boldsymbol{M}$. The critic is therefore a mapping $f_{\boldsymbol{w}} : \mathcal{X} \times \{0,1\}^d \to \mathbb{R}$ trained to maximize

$$\max_{\boldsymbol{w}\in\mathcal{W}} \mathrm{E}_{\boldsymbol{X}\sim\mathrm{P}(\boldsymbol{X})} f_{\boldsymbol{w}}(\boldsymbol{X}, \boldsymbol{M}) - \mathrm{E}_{\boldsymbol{Z}\sim\mathrm{P}(\boldsymbol{Z})} f_{\boldsymbol{w}}(\hat{\boldsymbol{X}}_{\boldsymbol{Z}}, \boldsymbol{M}),$$

which is usually estimated by sample means from mini-batches. The overall structure of WGAIN is depicted in Figure 1.

Algorithm 1: WGAIN training pseudo-code

```
Input: $\alpha$ - the learning rate; $w_{\max}$ - maximal norm used in clipping; $m$ - the
       mini-batch size
Draw $m$ samples from the dataset $\{\boldsymbol{x}_j\}_{j=1}^{m}$;
Draw $m$ samples from the mask distribution $\{\boldsymbol{m}_j\}_{j=1}^{m}$;
Draw $m$ samples from the normal distribution of $\boldsymbol{Z}$, $\{\boldsymbol{z}_j\}_{j=1}^{m}$;
while not converged do
    $\tilde{\boldsymbol{x}}_{\boldsymbol{z}_j} \leftarrow \boldsymbol{z}_j \odot (1-\boldsymbol{m}_j) + \boldsymbol{x}_j \odot \boldsymbol{m}_j$;
    $\hat{\boldsymbol{x}}_{\boldsymbol{z}_j} \leftarrow g(\tilde{\boldsymbol{x}}_{\boldsymbol{z}_j}, \boldsymbol{m}_j) \odot (1-\boldsymbol{m}_j) + \boldsymbol{x}_j \odot \boldsymbol{m}_j$;
    Update weights $\boldsymbol{w}$ of $f_{\boldsymbol{w}}$ using RMSProp with learning rate $\alpha$ and gradient
    $\nabla J(f_{\boldsymbol{w}}) = \lambda_{f_{\boldsymbol{w}}} \nabla \left[\frac{1}{m}\sum_{i=1}^{m} f_{\boldsymbol{w}}\big(\hat{\boldsymbol{x}}_{\boldsymbol{z}_j}, \boldsymbol{m}_j\big) - \frac{1}{m}\sum_{i=1}^{m} f_{\boldsymbol{w}}\big(\boldsymbol{x}_j, \boldsymbol{m}_j\big)\right]$;
    Clip the norm of $\boldsymbol{w}$ by $\boldsymbol{w}_{\max}$;
    Update weights of $g$ using RMSProp with learning rate $\alpha$ and gradient
    $\nabla J(g) = \nabla \left[-\lambda_g \frac{1}{m}\sum_{i=1}^{m} f_{\boldsymbol{w}}\big(\hat{\boldsymbol{x}}_{\boldsymbol{z}_j}, \boldsymbol{m}_j\big) + \lambda_{\mathrm{MSE}} \frac{1}{m}\sum_{i=1}^{m} \|\hat{\boldsymbol{x}}_{\boldsymbol{z}_j} - \boldsymbol{x}_j\|^2\right]$;
end
```

3.1 Training

The critic $f_{\boldsymbol{w}}$ is used in adversarial training of both the generator g and the critic itself. There the generator and the critic play an iterative two-player minimax game when the critic wants to recognize the imputed values from the real ones and the goal of the generator is to trick the critic so it cannot recognize them. Moreover, the generator's output is tighten to the correct output by the squared loss function L_{MSE}.

Putting it all together, we have two objective functions to minimize. The first corresponds to training of the discriminator given by

$$J(f_{\boldsymbol{w}}) = \lambda_{f_{\boldsymbol{w}}} \Big(\mathrm{E}_{\boldsymbol{Z}\sim \mathrm{P}(\boldsymbol{Z})} f_{\boldsymbol{w}}(\hat{\boldsymbol{X}}_{\boldsymbol{Z}}, \boldsymbol{M}) - \mathrm{E}_{\boldsymbol{X}\sim \mathrm{P}(\boldsymbol{X})} f_{\boldsymbol{w}}(\boldsymbol{X}, \boldsymbol{M}) \Big),$$

where the weight λ enables one to increase or decrease the influence of the corresponding gradient. Second is the objective for the generator,

$$J(g) = -\lambda_g \, \mathrm{E}_{\boldsymbol{Z}\sim \mathrm{P}(\boldsymbol{Z})} f_{\boldsymbol{w}}(\hat{\boldsymbol{X}}_{\boldsymbol{Z}}, \boldsymbol{M}) + \lambda_{\mathrm{MSE}} \, \mathrm{E}_{\boldsymbol{X}\sim \mathrm{P}(\boldsymbol{X}), \boldsymbol{Z}\sim \mathrm{P}(\boldsymbol{Z})} L_{\mathrm{MSE}}(\hat{\boldsymbol{X}}_{\boldsymbol{Z}}, \boldsymbol{X}),$$

where the first term λ_g and λ_{MSE} are weights enabling one to strengthen or weaken the influence of squared loss function. The optimization is done via alternating gradient descent, where the first step is updating the critic $f_{\boldsymbol{w}}$ and the second step is updating the generator g. Hence, when perfectly trained, the discriminator gives negative values to cases with imputed features and positive values for cases with true features. On the other hand, the generator entering the critic will be pushed to obtain large positive values of the critic as it gives to real values.

The pseudo-code of the WGAIN training is given in Algorithm 1.

4 Experiments

An experimental validation of WGAIN using ten real and two artificial publicly available datasets is presented below. These datasets contain numeric data only and are devoted to the classification task. Their overview, together with the corresponding best performing classification models, is given in Table 2.

During the experiments, all datasets were divided as follows: 70% of data was used to train all classification and imputation models and 30% was used as a test set to evaluate imputation performance. The imputation models were trained to impute in scenarios where randomly selected combinations of multiple features are missing. The amount of missingness varies from 10% to 50% of missing features. Finally, evaluation of the accuracy of the classification model combined with all imputation methods is performed on the test dataset.

4.1 Imputation Models and Their Parameters

Let us start with the presented WGAIN model. The generator and the critic architectures were the same for all datasets and are described in Table 1. During the training, the following settings were used:

– The original data $\boldsymbol{X}$ are sampled in mini-batches of size $m = 128$.
– The missingness is introduced using the mask $\boldsymbol{M}$ with the following distribution: for each training point, the portion of missingness is sampled from a uniform distribution between 0 and maximum missing rate, which was chosen to be 0.3. Then the binary elements of $\boldsymbol{M}$ were independently sampled with this portion of missingness, i.e., its item is 0 with a probability which was previously sampled.
– The components of random vector $\boldsymbol{Z}$ are i.i.d. with normal distribution having 0 mean and standard deviation 0.01.
– The weights of the objectives functions $J(f_{\boldsymbol{w}})$ and $J(g)$ are $\lambda_{f_{\boldsymbol{w}}} = 10$, $\lambda_g = 2$, and $\lambda_{\text{MSE}} = 1$.
– Maximal norm used in clipping of the critic weights is $w_{\max} = 1$.
– We use RMSProp with learning rate $\alpha = 0.0001$ as optimizers.
– The number of training epochs is 8000.

The GAIN implementation follows the original paper [35] and is analogous to the described WGAIN with the following differences:

– The generator architecture differs only in the sizes of layers, which are all equal to the input dimension.
– The discriminator architecture is analogous to the generator architecture except for the sigmoid activation function on the last layer.
– The binary elements of $\boldsymbol{M}$ are independently sampled with the common portion of missingness, which is 0.2.
– The hint rate used for the hint matrix is 0.9.
– As an optimizer, we use Adam with learning rate of 0.0001.
– The number of training epochs is 7000.

Table 1. Architecture details of the WGAIN. Abbreviation: FC=fully connected layer.

Layer	Generator
	concatenate data and mask
1	FC-(1.5 input dimension), ReLU
2	FC-(1.25 input dimension), ReLU
3	FC-(input dimension), Linear
Layer	**Critic**
	concatenate data and mask
1	FC-(1.5 input dimension), ReLU
2	FC-(1.25 input dimension), ReLU
3	FC-(1), Linear

In the case of DAE, we follow the structure presented in [15]. For the hyper-parameters search, the hyperband [21] algorithm was used. The typical best setup is the following: ELU as an activation function, three layers in both the encoder and decoder parts, the size of the code is twice the input dimension, and no regularization is used.

DAE, GAIN, and WGAIN models were implemented in the `TensorFlow` library [1].

The implementation of VAEAC was based on the repository [2] corresponding to the original paper [17]. All hyper-parameters stayed in the default settings.

For the MICE method (*mice*), we used the `IterativeImputer` class from the `scikit-learn` library[3]. In the default settings, the implementation uses Bayesian ridge regression as the internal imputation model and multiple imputations are pooled by the mean.

The k-NN imputation (*knn*) was implemented using the `fancyimpute` library [4]. A missing value is imputed by sampling the mean of the values of its neighbors weighted proportionally to their inverse distances. In the case where multiple features are missing, we impute all missing values at once (per row). For the hyper-parameter k values $11, 13, 15, 17, 19, 21, 23, 25$ were tested. The best k was chosen based on the RMSE value.

4.2 Evaluation

The impact of imputation is evaluated using the classification accuracy changes of the best performing classification model chosen from the six commonly used ones: logistic regression (LR), multi-layer perceptron (MLP), k-nearest neighbors (k-NN), naive Bayes (NB), extreme gradient boosted trees (XGBT) (for details see [7]), and random forest (RF). The best hyperparameters for each model were found using randomized search algorithm. The accuracy of the best

[1] `TensorFlow` platform: `https://www.tensorflow.org`
[2] `VAEAC` implementation: `https://github.com/tigvarts/vaeac`
[3] `Scikit-learn` library: `https://scikit-learn.org`
[4] `Fancyimpute` repository: `https://github.com/iskandr/fancyimpute`

performing model for each dataset is shown in Table 2. Furthermore, the root mean squared error (RMSE) between the original and the imputed data is also used for evaluation, e.g., [35,6,17].

Table 2. Details of datasets with the corresponding best performing classification model and its accuracy on the test set. The number of features (# f.) does not include the target label. The # r. stands for the number of records.

Name	Type	# f.	# r.	model name	accuracy
Cancer [23]	real	9	683	RF	0.975
EEG [23]	real	14	14980	k-NN	0.952
MAGIC [23]	real	10	19020	XGBT	0.868
Ozone-1 [23]	real	72	1846	k-NN	0.977
Ozone-8 [23]	real	72	1848	LR	0.941
QSAR [23]	real	41	1055	MLP	0.868
Shuttle [23]	real	9	57998	RF	0.999
Spambase [23]	real	57	4597	MLP	0.940
Waveform [23]	real	21	5000	LR	0.869
Yeast [23]	real	8	1484	XGBT	0.578
Ringnorm [1]	art.	20	7400	NB	0.979
Twonorm [1]	art.	20	7400	MLP	0.979

After all classification models were trained, and the most accurate one for each dataset was chosen, they were combined with imputation methods. Then, the accuracies of classification models on the imputed test dataset were measured.

Since it is not sound to compare accuracies for different datasets, we use a rank comparison. To do so, the algorithms are ranked for each dataset separately, the best performing algorithm getting the rank of 1, the second-best rank 2, etc. An example of accuracies and corresponding ranks for 10% of missingness is presented in Tables 4 and 5. Even in cases when WGAIN is not the best, its performance is always comparable to the best performers. The only exception is the EEG dataset, where k-NN imputation performs the best and the WGAIN is in second place with a difference of almost two percent.

The algorithms can be compared, taking the mean over the datasets. The results can be seen in Table 9. When the degree of missingness varies from 10% to 30% the WGAIN performs the best. When the degree of missingness is upwards of 30% the GAIN outperforms the WGAIN.

The results of the ranking evaluation can be statistically evaluated using the Friedman test [13,14] and the corresponding posthoc tests. For more details, see [9]. P-values of Friedman χ^2_F and F_F tests are shown in Table 8. One can see that from 20% to 40% of missing data the null-hypothesis, that all methods perform the same, can be rejected at a 10% significance level. However, when the Bonferroni-Dunn post-hoc test is applied the performance of WGAIN is significantly better than DAE only and just for 20% and 30% of missing data.

The same ranking process is repeated for RMSE with results in Table 3. An example of RMSE and corresponding ranks for 10% of missingness is presented

Table 3. Mean ranks of the RMSE for different degrees of missingness.

	Degree of missingness				
Method	10%	20%	30%	40%	50%
k-NN	2.67	2.67	2.67	3.08	2.75
MICE	3.17	3.50	3.33	3.00	2.91
DAE	5.08	5.08	5.17	5.33	4.91
VAEAC	3.25	3.33	3.42	3.17	3.50
GAIN	**2.17**	**2.08**	**2.17**	**2.08**	**2.50**
WGAIN	4.67	4.33	4.25	4.33	4.42

Table 4. Mean of the accuracies for 10% of missing features.

	k-NN	MICE	DAE	VAEAC	GAIN	WGAIN
Cancer	0.9700	0.9744	0.9744	0.9749	0.9739	**0.9755**
EEG	**0.9226**	0.9046	0.8994	0.6374	0.9028	0.9052
MAGIC	**0.8562**	0.8465	0.8459	0.8527	0.8522	0.8511
Ozone-1	0.9754	0.9763	**0.9768**	0.9762	0.9759	0.9763
Ozone-8	0.9404	**0.9407**	0.9405	0.9405	0.9406	0.9406
QSAR	0.8608	0.8619	0.8615	0.8619	0.8609	**0.8626**
Shuttle	0.9995	**0.9996**	0.9945	0.9994	0.9992	0.9995
Spambase	**0.9363**	0.9278	0.9307	0.9303	0.9339	0.9296
Waveform	0.8603	0.8604	0.8585	0.8596	**0.8605**	0.8593
Yeast	0.5516	0.5507	0.5533	0.5496	0.5541	**0.5558**
Ringnorm	0.9668	0.9671	0.9672	0.9673	0.9674	**0.9680**
Twonorm	0.9711	0.9716	0.9716	0.9716	0.9719	**0.9723**

Table 5. Ranks of accuracies of the imputation methods for 10% of missing features.

	k-NN	MICE	DAE	VAEAC	GAIN	WGAIN
Cancer	6	3.5	3.5	2	5	1
EEG	1	3	5	6	4	2
MAGIC	1	5	6	2	3	4
Ozone-1	6	2	1	4	5	3
Ozone-8	6	1	5	4	2.5	2.5
QSAR	6	2.5	4	2.5	5	1
Shuttle	2	1	6	3.5	5	3.5
Spambase	1	6	3	4	2	5
Waveform	3	2	6	4	1	5
Yeast	4	5	3	6	2	1
Ringnorm	6	5	4	3	2	1
Twonorm	6	4	4	4	2	1

Table 6. Mean of the RMSE for 10% of missing features.

	k-NN	MICE	DAE	VAEAC	GAIN	WGAIN
Cancer	**0.1905**	0.1960	0.2219	0.1943	0.1959	0.2087
EEG	**16.4752**	27.8197	29.1700	293.9315	21.8986	34.2722
MAGIC	**0.1821**	0.2067	0.2072	0.1866	0.1844	0.1928
Ozone-1	0.1364	**0.0826**	0.1204	0.1047	0.1038	0.1051
Ozone-8	0.1549	**0.0972**	0.1473	0.1233	0.1230	0.1206
QSAR	**0.2356**	0.3115	0.2505	0.2445	0.2376	0.2492
Shuttle	**0.0954**	0.1022	0.1316	0.1085	0.1053	0.1097
Spambase	**0.2404**	0.2723	0.2692	0.2659	0.2587	0.2731
Waveform	0.2312	0.2304	0.2690	0.2301	**0.2278**	0.2429
Yeast	**0.3542**	0.3610	0.3666	0.3585	0.3560	0.3631
Ringnorm	0.3222	**0.3184**	0.3187	0.3191	0.3190	0.3282
Twonorm	0.2967	0.2948	0.3081	0.2935	**0.2918**	0.2975

Table 7. Ranks of RMSE of the imputation methods for 10% of missings.

	k-NN	MICE	DAE	VAEAC	GAIN	WGAIN
Cancer	1	4	6	2	3	5
EEG	1	3	4	6	2	5
MAGIC	1	5	6	3	2	4
Ozone-1	6	1	5	3	2	4
Ozone-8	6	1	5	4	3	2
QSAR	1	6	5	3	2	4
Shuttle	1	2	6	4	3	5
Spambase	1	5	4	3	2	6
Waveform	4	3	6	2	1	5
Yeast	1	4	6	3	2	5
Ringnorm	5	1	2	4	3	6
Twonorm	4	3	6	2	1	5

Table 8. P-values of Friedman χ_F^2 and F_F test.

	Degree of missingness				
	10%	20%	30%	40%	50%
χ_F^2 test	0.252	0.049	0.106	0.020	0.477
F_F test	0.253	0.041	0.099	0.014	0.490

Table 9. Mean ranks of the accuracy changes for different degrees of missingness.

Method	Degree of missingness				
	10%	20%	30%	40%	50%
k-NN	4.00	3.54	3.63	3.17	3.25
MICE	3.33	4.04	3.75	4.21	3.71
DAE	4.21	4.79	4.67	4.71	4.33
VAEAC	3.75	3.13	3.50	3.59	3.54
GAIN	3.21	2.83	2.83	**2.21**	**2.79**
WGAIN	**2.50**	**2.67**	**2.63**	3.12	3.38

in Tables 6 and 7. Interestingly, the WGAIN performance is one of the worst, whereas the GAIN performs the best. This is in contrary to the fact that the WGAIN imputes the best from the accuracy point of view. Hence, we can see that low RMSE, which is usually taken as a measure of imputation quality may not lead to the desired performance on the target task. On the other hand, the RMSE differences are relatively small as can be seen in Table 6.

5 Conclusion

We propose a Wasserstein Generative Adversarial Imputation Network as a new deep learning imputation model. It is inspired by the GAIN. However, the discriminator is replaced by the Wasserstein critic. It is known that the Wasserstein approach does not suffer from vanishing gradients in the way that a vanilla GAN does. This enables the model to capture the desired distribution better. One may assume such benefits in WGAIN as well. We experimentally showed that in the imputation performance measured by classification accuracy, the WGAIN outperforms the other methods when the degree of missingness is lower than or equal to 30%. In other cases, it is competitive. In future work, we would like to focus on the use of WGAIN in image inpainting tasks.

Acknowledgements

This research has been supported by SGS grant No. SGS20/213/OHK3/3T/18 and by GACR grant No. GA18-18080S.

References

1. J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17, 2011.
2. Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.
3. Á. Arroyo, Á. Herrero, V. Tricio, E. Corchado, and M. Woźniak. Neural models for imputation of missing ozone data in air-quality datasets. *Complexity*, 2018:14, 2018.

4. M. J. Azur, E. A. Stuart, C. Frangakis, and P. J. Leaf. Multiple imputation by chained equations: what is it and how does it work? *International journal of methods in psychiatric research*, 20(1):40–49, 2011.
5. B. K. Beaulieu-Jones and J. H. Moore. Missing data imputation in the electronic health record using deeply learned autoencoders. In *Pacific Symposium on Biocomputing 2017*, pages 207–218. World Scientific, 2017.
6. R. D. Camino, Ch. A. Hammerschmidt, and R. State. Improving missing data imputation with deep generative models. *CoRR*, abs/1902.10666, 2019.
7. T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.
8. A. F. Costa, M. S. Santos, J. P. Soares, and P. H. Abreu. Missing data imputation via denoising autoencoders: The untold story. In Wouter Duivesteijn, Arno Siebes, and Antti Ukkonen, editors, *Advances in Intelligent Data Analysis XVII*, pages 87–98, Cham, 2018. Springer International Publishing.
9. J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, December 2006.
10. Y. Duan, Y. Lv, J.-L. Liu, and F.-Y. Wang. An efficient realization of deep learning for traffic data imputation. *Transportation Research Part C: Emerging Technologies*, 72:168 – 181, 2016.
11. A. Farhangfar, L. A. Kurgan, and J. G. Dy. Impact of imputation of missing values on classification error for discrete data. *Pattern Recognition*, 41:3692–3705, 2008.
12. M. Friedjungová, M. Jiřina, and D. Vašata. Missing features reconstruction and its impact on classification accuracy. In *Computational Science – ICCS 2019*, pages 207–220, Cham, 2019. Springer International Publishing.
13. M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.
14. M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *Ann. Math. Statist.*, 11(1):86–92, 03 1940.
15. L. Gondara and K. Wang. Mida: Multiple imputation using denoising autoencoders. In D. Phung, V. S. Tseng, G. I. Webb, B. Ho, M. Ganji, and L. Rashidi, editors, *Advances in Knowledge Discovery and Data Mining*, pages 260–272, Cham, 2018. Springer International Publishing.
16. I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.
17. O. Ivanov, M. Figurnov, and D. Vetrov. Variational autoencoder with arbitrary conditioning. In *International Conference on Learning Representations*, 2019.
18. P. Jonsson and C. Wohlin. An evaluation of k-nearest neighbour imputation using likert data. In *10th International Symposium on Software Metrics, 2004. Proceedings.*, pages 108–118. IEEE, 2004.
19. D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.
20. D. Lee, J. Kim, W.-J. Moon, and J. Ch. Ye. Collagan : Collaborative GAN for missing image data imputation. *CoRR*, abs/1901.09764, 2019.

21. L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.*, 18(1), 2017.
22. S. Ch.-X. Li, B. Jiang, and B. M. Marlin. Misgan: Learning from incomplete data with generative adversarial networks. *CoRR*, abs/1902.09599, 2019.
23. M. Lichman. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2013.
24. R. J. A. Little and D. B Rubin. *Statistical analysis with missing data*, volume 333. John Wiley & Sons, 2014.
25. M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret. Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot. *Sensors*, 17(9), 2017.
26. John T McCoy, Steve Kroon, and Lidia Auret. Variational autoencoders for missing data imputation with application to a simulated milling circuit. *IFAC-PapersOnLine*, 51(21):141–146, 2018.
27. A. Nazábal, P. M. Olmos, Z. Ghahramani, and I. Valera. Handling incomplete heterogeneous data using vaes. *CoRR*, abs/1807.03653, 2018.
28. Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.
29. J. L. Schafer. *Analysis of Incomplete Multivariate Data.* Chapman and Hall, London, 1997.
30. E.-L. Silva-Ramírez, R. Pino-Mejías, and M. López-Coello. Single imputation with multilayer perceptron and multiple imputation combining multilayer perceptron and k-nearest neighbours for monotone patterns. *Applied Soft Computing*, 29:65 – 74, 2015.
31. Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3483–3491. Curran Associates, Inc., 2015.
32. S. Van Buuren. *Flexible imputation of missing data.* Chapman and Hall/CRC, 2018.
33. P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. *Extracting and composing robust features with denoising autoencoders.* In Proceedings of the 25th international conference on Machine learning ACM, 2008.
34. L. Z. Wong, H. Chen, S. Lin, and D. Ch. Chen. Imputing missing values in sensor networks using sparse data representations. In *Proceedings of the 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '14, pages 227–230, New York, NY, USA, 2014. ACM.
35. J. Yoon, J. Jordon, and M. van der Schaar. GAIN: Missing data imputation using generative adversarial nets. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5689–5698, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.
36. A. Zadeh, Y. Ch. Lim, P. P. Liang, and L.-P. Morency. Variational auto-decoder. *CoRR*, abs/1903.00840, 2019.