E^2GAN: End-to-End Generative Adversarial Network for Multivariate Time Series Imputation

Yonghong Luo[1] , **Ying Zhang**[1*] , **Xiangrui Cai**[2] and **Xiaojie Yuan**[1,2]
[1]College of Computer Science, Nankai University, Tianjin, China
[2]College of Cyber Science, Nankai Univeristy, Tianjin, China
{luoyonghong, zhangying, caixiangrui, yuanxiaojie}@dbis.nankai.edu.cn

Abstract

The missing values, appear in most of multivariate time series, prevent advanced analysis of multivariate time series data. Existing imputation approaches try to deal with missing values by deletion, statistical imputation, machine learning based imputation and generative imputation. However, these methods are either incapable of dealing with temporal information or multi-stage. This paper proposes an end-to-end generative model E^2GAN to impute missing values in multivariate time series. With the help of the discriminative loss and the squared error loss, E^2GAN can impute the incomplete time series by the nearest generated complete time series at one stage. Experiments on multiple real-world datasets show that our model outperforms the baselines on the imputation accuracy and achieves state-of-the-art classification/regression results on the downstream applications. Additionally, our method also gains better time efficiency than multi-stage method on the training of neural networks.

1 Introduction

Recent advances in sensors and hardwares promote the rapid emergence of electronic data especially the multivariate time series data. Examples of multivariate time series appear in fields like medicine, meteorology, economics and transportation science [Chatfield, 2016]. Applications of multivariate time series include predicting the health status of patients [Hyland *et al.*, 2018], weather forecasting [Bright *et al.*, 2015], stock market analysis [Granger and Newbold, 2014], adjusting the traffic flow of big cities [Lv *et al.*, 2015], etc. The analysis of time series can also lead to a deep and thorough understanding of the time series itself.

Nevertheless, missing values in multivariate time series data poses a challenge for engineers and researchers, due to the broken of collection sensors, the ill designed collection process and mistakes made by humans. Taking PhysioNet challenge 2012 dataset [Silva *et al.*, 2012], a public medical dataset, as an example, the average missing rate of this multivariate time series data is above 80%. The missing values hinder the advanced analysis of time series [Berglund *et al.*, 2015].

Traditional missing values processing methods can be summarized into two categories. The first one tries to delete the observations which are partially missing. Deletion methods prevent the further analysis in some way [Graham, 2009]. The imputation method that imputes the missing values with reasonable values is the other type of method to deal with missing values. State-of-the-art imputation algorithms include K-Nearest Neighbor (KNN) based imputation [Hudak *et al.*, 2008], Matrix Factorization based imputation [Acar *et al.*, 2010] and Recurrent Neural Networks (RNN) based imputation [Berglund *et al.*, 2015; Che *et al.*, 2018; Cao *et al.*, 2018].

Recently, Generative Adversarial Networks (GAN) [Goodfellow *et al.*, 2014] based methods, which first learn to generate new samples that follow the distribution of the training dataset and then impute the missing values, have achieved state-of-the-art performance. [Luo *et al.*, 2018] proposes a two-stage time series imputation method. This method first trains a GAN model. At the second stage, for each sample, it tries to optimize the "noise" input vector and find the best matched input vector of the generator so that the generated sample is most similar to the original one. However, this two-stage method needs a lot of time to find the best matched input vector and this vector is not always the best especially when the initial value of the "noise" is not properly set.

Inspired by [Sabokrou *et al.*, 2018], this paper propose an end-to-end GAN-based imputation model E^2GAN, which not only simplifies the process of time series imputation but also generates more reasonable values for the filling of missing values. E^2GAN takes a compressing and reconstructing strategy to avoid the "noise" optimization stage in [Luo *et al.*, 2018]. In the generator, we add a random vector to the original sample and try to map it into a low-dimensional vector. Then we reconstruct it from the low-dimensional vector. The generator seeks to find a network structure that can not only best compress and reconstruct the multivariate time series but also fool the discriminator. The proposed method automatically learns internal representations of the time series and try its best to reconstruct this temporal data. The new architecture also improves the imputation performance by getting a better feature representation of samples, which contributes to a better reconstructed samples and improves the imputation

[*]Corresponding author.

results.

Finally, we use the reconstructed sample to impute the missing values. The experiments conducted on two real datasets show that our method achieves a new state-of-the-art imputation accuracy and achieves better time efficiency than the two-stage method.

In summary, the main contributions of this paper are as follows: (1) We propose a novel end-to-end generative adversarial network that simplifies the process of time series imputation, i.e., reduces the training time. (2) The noised compressing and reconstructing strategy makes sure that the imputed values more reasonable than [Luo *et al.*, 2018]'s. (3) Experiments on multiple real-world datasets show that the proposed method achieves new state-of-the-art imputation accuracy.

2 Related Works

There is a lot of literatures on the processing and imputation of missing values. Due to the limited space, we only describe a few closely related ones. *Deletion method*, who simply deletes the partly missed observations, prevents the further analysis of dataset due to the deleted information [Graham, 2009]. *Statistical imputation* tries to fill missing values by mean value [Kantardzic, 2011], last observed value [Amiri and Jensen, 2016] and mode value [Purwar and Singh, 2015]. Many previous works show that machine learning based imputation methods are useful for data imputation or time series imputation. Multivariate Imputation by Chained Equations (MICE) [White *et al.*, 2011] fills the missing values by using iterative regression model. MICE imputes each incomplete variable by a separate model. ARIMA [Box *et al.*, 2015], ARFIMA [Galbraith and Zinde-Walsh, 2001] and SARIMA [Hamzaçebi, 2008] are representative auto-regressive methods that can be used to predict and impute missing values. K-Nearest Neighbor (KNN) [Hudak *et al.*, 2008] algorithm uses the mean value of k nearest neighbors to fill missing values. Matrix Factorization [Acar *et al.*, 2010] algorithm factorizes the incomplete dataset into low-rank matrices and adopts the product of these two matrices to impute the missing values.

There are also some recurrent neural networks based imputation methods. [Berglund *et al.*, 2015] have proposed two probabilistic interpretations of bidirectional recurrent neural networks that can be used to reconstruct missing samples efficiently. [Che *et al.*, 2018] have proposed GRUD, which imputes missing values of clinical dataset with a smooth fasion. GRUD takes the advantage of last observed value and mean value to represent missing patterns of incomplete time series. BRITS [Cao *et al.*, 2018] is a novel neural network method that directly learns the missing values in a bidirectional recurrent dynamical system, without any specific assumption.

Recently, Generative Adversarial Networks (GAN) [Goodfellow *et al.*, 2014], which seeks to generate new samples that obey the distribution of training dataset, has been used to impute missing values. The generated samples, which are designed to fill the missing values, significantly boost the imputation accuracy. GAIN [Yoon *et al.*, 2018], a GAN-based imputation method, uses a hint vector that is conditioned on what we actually observed to impute missing values. GAIN has made tremendous advances in data imputation. However, the main drawback of GAIN is the weak ability to impute time series. [Luo *et al.*, 2018] have proposed a two-stage GAN based time series imputation method. This method first train a GAN model that can produce new sample from a input vector. In the second stage, this method tries to find a "best" matched input "noise" vector of the generator so that the generated sample is most similar to the original one. This two-stage training strategy needs a lot of time to train and the "best" matched vector is not always the best at some point.

3 Problem Formulation

A d-dimensional multivariate time series $\boldsymbol{x}$, observed at $\boldsymbol{t}$=$(t_0, \ldots, t_{n-1})$, is denoted by $\boldsymbol{x}$=$(x_0, \ldots, x_{n-1}) \in \mathbb{R}^{d\times n}$, where $\boldsymbol{t}$ is the observing timestamp, and x_t is the tth observation. The locations of the missing values is of great importance. We record the locations of missing values and the time lags of two adjacent values.

Suppose that $\boldsymbol{m}\in \mathbb{R}^{d\times n}$ is a mask matrix that takes values in $\{0, 1\}$. $\boldsymbol{m}$ means whether the values of $\boldsymbol{x}$ exist or not, for example, if x_i^j exists, m_i^j=1, otherwise, 0. The purpose of multivariate time series imputation is to impute the missing values in $\boldsymbol{x}$ as accurate as possible.

The following part is an intuitive explanation of a 3-dimensional multivariate time series $\boldsymbol{x}$ and its corresponding $\boldsymbol{m}$.

$$\boldsymbol{x} = \begin{bmatrix} 1 & 6 & / & 9 & \\ / & / & 7 & 2 & \cdots \\ 9 & / & / & 79 & \end{bmatrix}, \boldsymbol{m} = \begin{bmatrix} 1 & 1 & 0 & 1 & \\ 0 & 0 & 1 & 1 & \cdots \\ 1 & 0 & 0 & 1 & \end{bmatrix},$$

$$\boldsymbol{t} = (0, 2, 5, 10, \ldots).$$

We define a matrix $\boldsymbol{\delta} \in \mathbb{R}^{d\times n}$ that records the time lag between current value and last observed value. The following part shows the calculation and a calculated example of the $\boldsymbol{\delta}$.

$$\delta_{t_i}^j = \begin{cases} 0, & i == 0 \\ t_i - t_{i-1}, & m_{t_{i-1}}^j == 1 \,\&\, i > 0 \\ \delta_{t_{i-1}}^j + t_i - t_{i-1}, & m_{t_{i-1}}^j == 0 \,\&\, i > 0 \end{cases}. \quad (1)$$

$$\boldsymbol{\delta} = \begin{bmatrix} 0 & 2 & 3 & 8 & \\ 0 & 2 & 5 & 5 & \cdots \\ 0 & 2 & 5 & 10 & \end{bmatrix}.$$

4 Approach

In this part, we show the details of the proposed method. A Generative Adversarial Network (GAN) [Goodfellow *et al.*, 2014] is composed by a Generator (G) and a Discriminator (D), whose goal, respectively, is to map low-dimensional vectors (random noise) to high-dimensional samples and distinguish between the fake data and the real data. They are jointly trained by adopting a min-max game. The ultimate aim of the GAN is to produce new samples that obey the distribution of the training dataset, by the adversarial training of the generator and discriminator. However, the original GAN can rarely get out of the mode collapse problem. Recently, [Arjovsky *et al.*, 2017] have proposed the Wasserstein GAN (WGAN), which can improve the learning stability and get away from the problem of mode collapse. The formulations of WGAN is a little different from the original one:

$$L_G = \mathbb{E}_{\boldsymbol{z}\sim P_g}\left[-D(G(\boldsymbol{z}))\right] , \tag{2}$$

$$L_D = \mathbb{E}_{\boldsymbol{z}\sim P_g}\left[D(G(\boldsymbol{z}))\right] - \mathbb{E}_{\boldsymbol{x}\sim P_r}\left[D(\boldsymbol{x})\right] . \tag{3}$$

In our generator, we take advantage of auto-encoder and recurrent cell. We compress the input incomplete time series $\boldsymbol{x}$ into a low-dimensional vector $\boldsymbol{z}$ with the help of recurrent neural network. Then we use this vector to reconstruct a complete time series $\boldsymbol{x'}$ to fool the discriminator. Meanwhile, we also force the $\boldsymbol{x'}$ as close as possible to $\boldsymbol{x}$ by using a squared error loss function. The discriminator of our method attempts to distinguish actual incomplete time series $\boldsymbol{x}$ and the fake but complete sample $\boldsymbol{x'}$ through the adoption of recursive neural network.

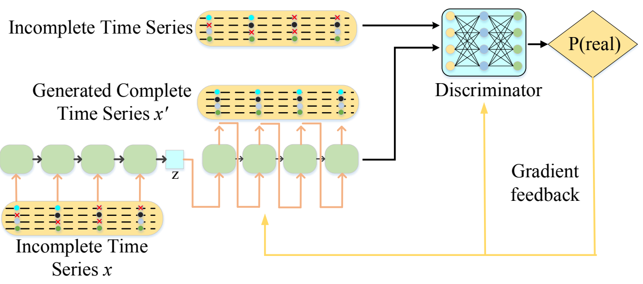
Incomplete Time Series
Generated Complete Time Series x'
Discriminator
P(real)
Gradient feedback
z
Incomplete Time Series x

Figure 1: The structure of the proposed method. The generator is a denoising auto-encoder which is mainly composed by the GRUI cell. The discriminator is another encoder that produces truth probability.

4.1 GRU Cell for Imputation

In the previous subsection, we have mentioned the mapping from high-dimensional incomplete time series to low-dimensional vector with the help of recurrent neural network. In this subsection, we will show the structure of this recurrent neural network that can not only compress the incomplete time series $\boldsymbol{x}$ but also reconstruct complete time series $\boldsymbol{x'}$ from a low-dimensional vector $\boldsymbol{z}$.

We have adopted the GRUI (**GRU** for **I**mputation), proposed in [Luo *et al.*, 2018], to process the incomplete time series. The GRUI is inspired by the GRUD proposed by [Che *et al.*, 2018]. Nevertheless, the GRUI is more simple than the GRUD. The time lag of two adjacent existent values is not always fixed due to the missing values. These irregular time lags should be made good use of. It's intuitive to decay the historical influence of the past observations if the current value is missed for a long time. The main idea of GRUI is to introduce a time decay vector $\boldsymbol{\beta}$ to decrease the memory of GRU cell. The following are the update functions of GRUI.

$$\boldsymbol{\beta}_{t_i} = 1/e^{\max(\boldsymbol{0}, \boldsymbol{W}_\beta \boldsymbol{\delta}_{t_i} + \boldsymbol{b}_\beta)} , \boldsymbol{h}'_{t_{i-1}} = \boldsymbol{\beta}_{t_i} \odot \boldsymbol{h}_{t_{i-1}} , \tag{4}$$

$$\boldsymbol{\mu}_{t_i} = \sigma(\boldsymbol{W}_\mu \left[\boldsymbol{h}'_{t_{i-1}}, \boldsymbol{x}_{t_i}\right] + \boldsymbol{b}_\mu) , \tag{5}$$

$$\boldsymbol{r}_{t_i} = \sigma(\boldsymbol{W}_r \left[\boldsymbol{h}'_{t_{i-1}}, \boldsymbol{x}_{t_i}\right] + \boldsymbol{b}_r) , \tag{6}$$

$$\tilde{\boldsymbol{h}}_{t_i} = tanh(\boldsymbol{W}_{\tilde{h}} \left[\boldsymbol{r}_{t_i} \odot \boldsymbol{h}'_{t_{i-1}}, \boldsymbol{x}_{t_i}\right] + \boldsymbol{b}_{\tilde{h}}) , \tag{7}$$

$$\boldsymbol{h}_{t_i} = (\boldsymbol{1} - \boldsymbol{\mu}_{t_i}) \odot \boldsymbol{h}_{t'_{i-1}} + \boldsymbol{\mu}_{t_i} \odot \tilde{\boldsymbol{h}}_{t_i} , \tag{8}$$

where $\boldsymbol{\delta}$ is the time lag matrix introduced in the "Problem Formulation" part, and $\boldsymbol{W}_\beta$, $\boldsymbol{W}_{\tilde{h}}$, $\boldsymbol{W}_r$, $\boldsymbol{W}_\mu$, $\boldsymbol{b}_\beta$, $\boldsymbol{b}_\mu$, $\boldsymbol{b}_r$, $\boldsymbol{b}_{\tilde{h}}$ are training parameters. The formulation of $\boldsymbol{\beta}$ guarantees that with the increase of time lags $\boldsymbol{\delta}$, the value of $\boldsymbol{\beta}$ decreases. The smaller the $\boldsymbol{\delta}$, the bigger the $\boldsymbol{\beta}$. This formulation also make sure that $\boldsymbol{\beta} \in (\boldsymbol{0}, \boldsymbol{1}]$. The GRUI is not the research focus of this paper, nevertheless, we can process the incomplete time series successfully by using the GRUI.

4.2 Generator Network Architecture

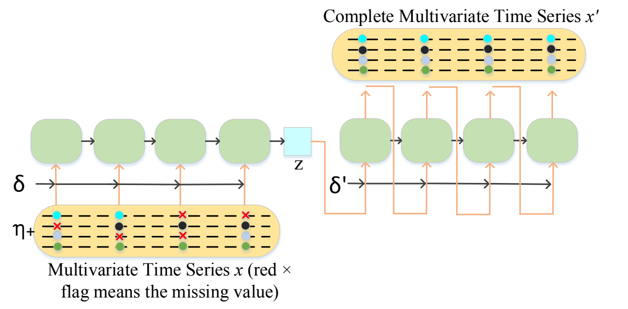
Complete Multivariate Time Series x'
δ
δ'
z
η+
Multivariate Time Series x (red × flag means the missing value)

Figure 2: The generator is an auto-encoder whose input is a noised incomplete time series and the output is a complete time series. By the squared error loss and the discriminative loss, we can get a generated sample $\boldsymbol{x'}$ that is not only most similar to original one but also belonging to the distribution of training dataset.

The generator of proposed method is shown at Figure 2. Since the auto-encoder is designed to reconstruct target samples, we choose it to generate complete time series $\boldsymbol{x'}$. Different from traditional auto-encoder, we add a random noise $\boldsymbol{\eta}$ which is sampled from a standard distribution $\mathcal{N}(0, 0.01)$ to the original sample. We use a similar idea just like the denoising auto-encoder, that is, compress and reconstruct the original samples from the destroyed samples. The main idea of denoising auto-encoder [Vincent *et al.*, 2008] is to drop out original samples and reconstruct the complete samples. However, the way to destroy original samples of our method is to add some noise rather than drop out some values of the input samples. The original samples is naturally missed, it is not a good idea to drop out the already missed samples especially the missing rates of some datasets are very high. For the naturally incomplete dataset, we just add some noise to the input samples and train an denoising auto-encoder.

$$\boldsymbol{G}(\boldsymbol{x} + \boldsymbol{\eta}) = \boldsymbol{x'} . \tag{9}$$

Since the purpose of the generator is to produce new sample $\boldsymbol{x'}$ that is most similar to $\boldsymbol{x}$, we add a squared error loss to the loss function of the generator. The loss function of the generator is a lot different from the one of traditional WGAN:

$$L_G = \lambda||\boldsymbol{x} \odot \boldsymbol{m} - \boldsymbol{G}(\boldsymbol{x} + \boldsymbol{\eta}) \odot \boldsymbol{m}||_2 - \boldsymbol{D}(\boldsymbol{x'}) , \tag{10}$$

where λ is a hyper-parameter that controls the weight of the discriminative loss and the squared error loss. In this way, $\boldsymbol{x'}$ can be used to impute the missing values of $\boldsymbol{x}$.

Figure 2 is a visual explanation of the generator network. To implement the generator network, we first feed the GRUI cell with the incomplete time series $\boldsymbol{x}$ and its time lags matrix

$\boldsymbol{\delta}$. We use zero value to replace the missing values of $\boldsymbol{x}$ at the input stage of GRUI. After a recurrent processing of the input time series, the last hidden state of the recurrent neural network will flow to a fully connected layer. The output of this fully connected layer is the compressed low-dimensional vector $\boldsymbol{z}$.

Next, we take $\boldsymbol{z}$ as the initial input of another fully connected layer. Then we use this output as the very first input of another GRUI layer. The current output of this GRUI layer will be fed into the next iteration of the same GRUI layer. At the final stage, we combine all the outputs of this GRUI layer as the generated sample $\boldsymbol{x'}$.

4.3 Discriminator Network Architecture

The network structure of discriminator is a decoder. In other words, the discriminator is also made up of a GRUI layer and a fully connected layer. The task of the discriminator is to distinguish between fake sample $\boldsymbol{x'}$ and true sample $\boldsymbol{x}$. The output of the discriminator is a probability that indicates the degree of authenticity. We try to find a set of parameters that can produce a high probability when we feed true sample $\boldsymbol{x}$, and produce a low probability when we feed fake sample $\boldsymbol{x'}$. Therefore, the loss function of the discriminator can be designed as following shows.

$$L_D = -\boldsymbol{D}(\boldsymbol{x}) + \boldsymbol{D}(\boldsymbol{x'})\,. \tag{11}$$

We feed incomplete time series $\boldsymbol{x}$ or complete time series $\boldsymbol{x'}$, with their corresponding $\boldsymbol{\delta}$, into the discriminator. With the help of GRUI, the time series can be successfully handled. The last hidden state of GRUI layer is fed into one fully connected layer that outputs the probability of being true. In order to enhance the generator, we update k times for generator and 1 time for discriminator at one iteration.

4.4 Imputation

In this paper, for every incomplete time series $\boldsymbol{x}$, we try to map it into a low-dimensional vector $\boldsymbol{z}$ and reconstruct a complete time series $\boldsymbol{x'}$ from $\boldsymbol{z}$, so that the fake time series $\boldsymbol{x'}$ is most close to the original one. We use the corresponding values of $\boldsymbol{x'}$ to fill in the missing values of $\boldsymbol{x}$. The imputation formula can be summarized as follows.

$$\boldsymbol{x}_{imputed} = \boldsymbol{x} \odot \boldsymbol{m} + (\boldsymbol{1} - \boldsymbol{m}) \odot \boldsymbol{x'}\,. \tag{12}$$

5 Experiments

In this section, the proposed method is evaluated on two real-world datasets. The experimental results are analyzed and compared in details.

5.1 Datasets and Baseline Methods

We use two real-world datasets including a medical dataset and a meteorologic dataset for the evaluation of the proposed method.

- **PhysioNet Challenge 2012 dataset (PhysioNet)**. The PhysioNet dataset, a public 80.67% missing medical dataset provided by [Silva *et al.*, 2012], is made up of records from 4000 ICU stays. Every ICU stay is a 48 hours multivariate incomplete time series that includes 42 features. The focus of the PhysioNet dataset is to develop methods that can predict in-hospital mortality well (classification task). 554 (13.85%) patients died in hospital, i.e., with positive label. The area under ROC curve (AUC) score is used for evaluating the performance of proposed method because of the non-balanced dataset. It is impossible to directly calculate the imputation accuracy since we can not get the complete dataset. We impute this dataset by different imputation methods and then use these imputed datasets to train same type of classifiers. Finally, the mortality prediction results calculated by these classifiers are used to indirectly determine the performance of different imputation methods.
- **KDD CUP 2018 Dataset (KDD).**[1] The KDD dataset (15% missing) is a public meteorologic dataset that comes from the KDD CUP Challenge 2018. The KDD dataset contains air quality and weather data which is hourly collected between 2017/1/30 to 2018/1/30. There are two tasks that conducted on this dataset. *1) Imputation task.* We randomly drop out p percent of all time series, where $p \in \{10, 20, \dots, 80\}$. Then we impute these time series and calculate the imputation accuracy by the mean squared error (MSE) between original values and imputed values. *2) Downstream task.* We randomly drop out 50 percent of all time series. Then we use different methods to impute them and get different imputed datasets. We use these datasets to train same type of regression models that predict the mean air quality of the next 6 hours. Finally, we get the imputation accuracy indirectly by the performances of this downstream task.

The baseline models are:

- **Statistical imputation methods**: We simply replace the missing values with **zero** value, **mean** value and **last observed** value.
- **Matrix Factorization (MF) imputation** [Acar *et al.*, 2010]: MF method is used to factorise the incomplete matrix into low-rank matrices and fill the missing values.
- **KNN** [Hudak *et al.*, 2008]: The missing values are replaced by using the k nearest neighbor samples.
- **MICE** [White *et al.*, 2011]: Multivariate Imputation by Chained Equations (MICE) fills the missing values by using iterative regression model.
- **GRUD** [Che *et al.*, 2018]: GRUD can be used to impute missing values. We use it as one of the baselines.
- **GAN-2-stage** [Luo *et al.*, 2018]: This method uses a GAN based two-stage method to impute missing values. We call this method as "GAN-2-stage".
- **GAIN** [Yoon *et al.*, 2018]: GAIN is another GAN based imputation method that uses a hint vector to impute the missing values.
- **BRITS** [Cao *et al.*, 2018]: This method is one of the state-of-the-art methods that uses bidirectional recurrent network to impute time series.

[1]KDD CUP. Available on: http://www.kdd.org/kdd2018/, 2018.

5.2 Implementation Details

We use ADAM algorithm to train networks. The hidden units number of all GRUI cells are 64. The dimension of low-dimensional vector $\boldsymbol{z}$ is also 64. The dropout rates of our experiments are always fixed with 0.5. We adopt the batch normalization technology on the GRUI layers. All input datasets are normalized with zero mean and unit variance. For all the experiments, we select 10% of dataset as validation set and another 10% as test set. Before the training of GAN, the generator is pre-trained for epochs with a mean squared error loss for fitting the next value in the training datasets.

5.3 Performance Comparison for Time series Imputation

Table 1 is the imputation results on KDD dataset by using proposed method and baseline methods which include statistical imputation, KNN imputation, MF imputation, MICE imputation, BRITS and GAN based imputation (GAN-2-stages and GAIN). The first column of Table 1 is missing rate which indicates how many percent values are dropped, and other columns are mean squared error. Our parameters for the KDD dataset are: epoch is 15, batch size is 16, learning rate is 0.005, pretrain epoch is 10, λ is 2. We can see that, in all cases, the GAN based methods gain the best imputation accuracies. The proposed method is one of the best methods and wins others methods in most cases. Furthermore, the proposed method gains huge advantage in time efficiency, we will show it at the latter part of this paper.

Miss -ing	Last	Mean	KNN	MF	MICE	GAIN	GAN-2 -stage	E^2GAN
10%	.614	.374	.465	.382	.468	.378	.355	**.334**(5.9%)
20%	.701	.578	.604	.598	.573	.557	.532	**.523**(1.7%)
30%	.812	.686	.640	.633	.662	.635	**.599**	.606(−1.2%)
40%	.808	.681	.685	.676	.678	.664	.652	**.650**(0.3%)
50%	.788	.747	.723	.710	.727	.693	**.653**	.657(−0.6%)
60%	.807	.801	.750	.722	.740	.732	.714	**.709**(0.7%)
70%	.885	.835	.783	.782	.825	.772	.751	**.747**(0.5%)
80%	.933	.827	.824	.791	.919	.798	.776	**.763**(1.7%)

Table 1: The MSE (the smaller, the better) results of the E^2GAN and other imputation methods on the KDD dataset. In most cases, the E^2GAN owns the best imputation accuracy.

5.4 Performance Comparison for Downstream Applications

Since we can not always get the complete dataset, it is impossible to directly compare the imputation accuracies. We use the performances of downstream applications to indirectly measure imputation accuracies. The downstream applications mainly contain two type of tasks which include the classification task that uses the PhysioNet dataset, and the regression task which uses the 50% discarded KDD dataset.

For the classification task which uses the PhysioNet dataset, we first impute this dataset with different comparative imputation methods. Then we use these datasets to train the same type of classifiers including logistic regression classifier, SVM (with RBF, Linear, Poly and Sigmoid kernels) classifiers, random forest classifier and RNN classifier. The RNN classifier is made up of a simple GRU layer and a fully connected layer. The parameters of PhysioNet are: epoch is 10, learning rate is 0.005, λ is 50, pretrain epoch is 5. Table 2 is the AUC scores of the classification task that uses the PhysioNet dataset. We can see that, the classifiers trained on the dataset that imputed by our proposed method always gain the best AUC scores. This phenomenon has shown the success of proposed method. We can also summarize that, the new state-of-the-art mortality prediction result (0.8724) on the PhysioNet dataset is achieved by using our method while the old one is 0.8603.

We also compare the mortality prediction results with other methods that do not impute missing values and directly classify the patients. Table 3 shows the detail results. It is obvious that simple RNN model trained on imputed dataset gains the best AUC score.

For the downstream task of 50% missed KDD dataset, we first impute it by using baseline models and proposed method. Then we take advantage of these datasets to train same tpye of regression models to predict the mean air quality of next 6 hours. We adopt the effects of these downstream applications, i.e., MSE, to indirectly measure the imputation accuracies. Figure 3 provides the detail results. We can see that, the regression models trained on the dataset that were imputed by the proposed method, achieve the best performance.

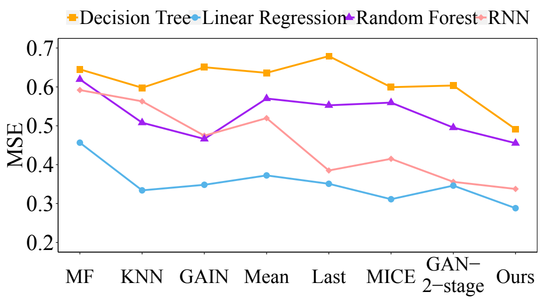

Figure 3: The MSE of air quality prediction results by different regression models trained on datasets that are imputed by different methods. The smaller the MSE, the better the results.

5.5 Discussions

Time Efficiency

We investigate the time efficiency of proposed method. The size of dataset and number of epochs are the most influential factors of time efficiency. It is difficult to compare the time efficiency of neural networks, especially the generative adversarial networks, because of different parameter settings. We try to compare our method with another GAN based time series imputation method. We fix the hardware, epoch, and some others parameters, and then compare the execution time of GAN-2-stages and our method. For the PhysioNet dataset, our method takes 204±5 seconds to reach the best AUC score, while the GAN-2-stages takes 2650±10 seconds. For the KDD dataset, our method costs 105±3 seconds and GAN-2-stages costs 1805±8 seconds. The main reason of this phenomenon is the drop of optimizing input “noise” vector of the generator. In other words, the proposed method only need to train a GAN model and directly impute missing

Method		Zero	Mean	Last	MF	KNN	MICE	GAIN	GAN-2-stage	E^2GAN
SVM	Poly	0.7378	0.6774	0.7709	0.3226	0.6773	0.6773	0.7605	0.7725	**0.7892**(2.2%)
	Linear	0.6436	0.6582	0.6672	0.6583	0.6582	0.6584	0.7185	0.7185	**0.7464**(3.9%)
	Sigmoid	0.5285	0.7887	0.7452	0.7886	0.7885	0.7890	0.7967	0.7921	**0.8070**(1.3%)
	RBF	0.5000	0.8043	**0.8213**	0.8045	0.8044	0.8043	0.8178	0.8157	0.8201(-1.4%)
/	RF	0.6937	0.6906	0.7443	0.7074	0.7003	0.6882	0.7302	0.7546	**0.7998**(5.7%)
/	LR	0.6586	0.6620	0.6701	0.6846	0.6120	0.6113	0.7122	0.7012	**0.7677**(9.4%)
/	RNN	0.7659	0.8423	0.8362	0.8495	0.8534	0.8521	0.8431	0.8603	**0.8724**(1.4%)

Table 2: The AUC score of mortality prediction by different classification models trained on datasets that are imputed by different methods.

Model	Result
GRUD [Che *et al.*, 2018]	0.8424
Regularized Logistic Regression [Johnson *et al.*, 2014]	0.848
Bayesian Ensemble Model [Johnson *et al.*, 2012]	0.8602
Two Stage GAN model [Luo *et al.*, 2018]	0.8603
Bidirectional Recurrent Model [Cao *et al.*, 2018]	0.8502
E^2GAN & Simple RNN Model	**0.8724**(1.4%)

Table 3: The AUC score of the mortality prediction task on the PhysioNet dataset. The simple RNN model that uses the dataset imputed by our method achieves the highest AUC score.

values. This end-to-end imputation method saves a lot of time compared to the two stages imputation method.

Influence of Dimensions

We also investigate the influence of dimensions as shown in Figure 4. The red lines are performance and the blue lines are time. The performances of downstream applications improve with the increasing of dimensions. However, there are some peaks at the performance lines. This indicates that some dimensions are more or less influential than others. And this phenomenon can be used to judge the importance of features if we draw a more detailed figure. The blue lines show how long the imputation will take with different dimensional datasets. We can conclude that the execution time grows with the raise of dimension. Besides, it is obvious that the growth of execution time is linear. It proves the proposed method's superiority of time efficiency in some way.

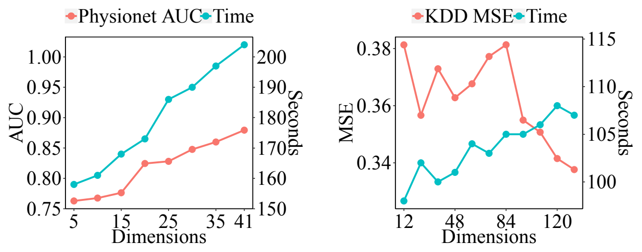

Figure 4: The influence of dimensions in mortality prediction task and air quality prediction task.

Method/Missing	10%	20%	30%	40%	50%	60%	70%	80%
E^2GAN	**.334**	**.523**	**.606**	**.650**	**.657**	**.709**	**.747**	**.763**
E^2GAN-no-D	.393	.552	.650	.696	.727	.771	.781	.823
E^2GAN-no-noise	.378	.547	.644	.714	.747	.772	.788	.824

Table 4: The ablation study of discriminator and noise vector η.

Ablation Study

We have evaluated the effects of the discriminator and the noise vector η. Table 4 shows the results on the KDD dataset (measured by MSE). The first row is the missing rate. The second row shows results of E^2GAN, and the third row shows results of E^2GAN with no discriminator. The last row shows results of E^2GAN with no noise vector η. As we can observe, the discriminator and the noise vector η are both helpful. The imputation accuracy decreases 8.35% without the discriminator and 8.5% without the noise vector η in average.

6 Conclusion

We propose a novel end-to-end model E^2GAN for the imputation of missing values in multivariate time series by taking advantage of the generative adversarial networks. Autoencoder and GRUI are used to generate new complete sample that is most close to the original incomplete one with the help of discriminative loss and squared error loss. Thus, we can impute the missing values with this generated new complete sample within a short training time. Various experiments with real-world datasets show that the proposed method boosts the imputation accuracy and achieves better classification/regression results on the downstream applications. With the help of our proposed method, we also get a new state-of-the-art mortality prediction result.

Acknowledgments

We thank the reviewers for their constructive comments. This work is partially supported by National Natural Science Foundation of China (No. 61772289 and No. U1836109). This work is also partially supported by Natural Science Foundation of Tianjin (No. 18ZXZNGX00110 and No. 17JCQNJC00200)

References

[Acar *et al.*, 2010] Evrim Acar, Daniel M Dunlavy, Tamara G Kolda, and Morten Mørup. Scalable tensor factorizations with missing data. In *Proceedings of the 2010 SIAM international conference on data mining*, pages 701–712. SIAM, 2010.

[Amiri and Jensen, 2016] Mehran Amiri and Richard Jensen. Missing data imputation using fuzzy-rough methods. *Neurocomputing*, 205:152–164, 2016.

[Arjovsky *et al.*, 2017] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, pages 214–223, 2017.

[Berglund *et al.*, 2015] Mathias Berglund, Tapani Raiko, Mikko Honkala, Leo Kärkkäinen, Akos Vetek, and Juha T Karhunen. Bidirectional recurrent neural networks as generative models. In *Advances in Neural Information Processing Systems*, pages 856–864, 2015.

[Box *et al.*, 2015] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[Bright *et al.*, 2015] JM Bright, CJ Smith, PG Taylor, and R Crook. Stochastic generation of synthetic minutely irradiance time series derived from mean hourly weather observation data. *Solar Energy*, 115:229–242, 2015.

[Cao *et al.*, 2018] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. Brits: Bidirectional recurrent imputation for time series. In *Advances in Neural Information Processing Systems*, pages 6776–6786, 2018.

[Chatfield, 2016] Chris Chatfield. *The analysis of time series: an introduction*. CRC press, 2016.

[Che *et al.*, 2018] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.

[Galbraith and Zinde-Walsh, 2001] John Galbraith and Victoria Zinde-Walsh. Autoregression-based estimators for arfima models. 2001.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[Graham, 2009] John W Graham. Missing data analysis: Making it work in the real world. *Annual review of psychology*, 60:549–576, 2009.

[Granger and Newbold, 2014] Clive William John Granger and Paul Newbold. *Forecasting economic time series*. Academic Press, 2014.

[Hamzaçebi, 2008] Coşkun Hamzaçebi. Improving artificial neural networks' performance in seasonal time series forecasting. *Information Sciences*, 178(23):4550–4559, 2008.

[Hudak *et al.*, 2008] Andrew T Hudak, Nicholas L Crookston, Jeffrey S Evans, David E Hall, and Michael J Falkowski. Nearest neighbor imputation of species-level, plot-scale forest structure attributes from lidar data. *Remote Sensing of Environment*, 112(5):2232–2245, 2008.

[Hyland *et al.*, 2018] Stephanie Hyland, Cristóbal Esteban, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. 2018.

[Johnson *et al.*, 2012] Alistair EW Johnson, Nic Dunkley, Louis Mayaud, Athanasios Tsanas, Andrew A Kramer, and Gari D Clifford. Patient specific predictions in the intensive care unit using a bayesian ensemble. In *Computing in Cardiology (CinC), 2012*, pages 249–252. IEEE, 2012.

[Johnson *et al.*, 2014] Alistair EW Johnson, Andrew A Kramer, and Gari D Clifford. Data preprocessing and mortality prediction: The physionet/cinc 2012 challenge revisited. In *Computing in Cardiology Conference (CinC), 2014*, pages 157–160. IEEE, 2014.

[Kantardzic, 2011] Mehmed Kantardzic. *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons, 2011.

[Luo *et al.*, 2018] Yonghong Luo, Xiangrui Cai, Ying Zhang, Jun Xu, et al. Multivariate time series imputation with generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1601–1612, 2018.

[Lv *et al.*, 2015] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, Fei-Yue Wang, et al. Traffic flow prediction with big data: A deep learning approach. *IEEE Trans. Intelligent Transportation Systems*, 16(2):865–873, 2015.

[Purwar and Singh, 2015] Archana Purwar and Sandeep Kumar Singh. Hybrid prediction model with missing value imputation for medical data. *Expert Systems with Applications*, 42(13):5621–5631, 2015.

[Sabokrou *et al.*, 2018] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. Adversarially learned one-class classifier for novelty detection. In *CVPR2019*, pages 3379–3388, 2018.

[Silva *et al.*, 2012] Ikaro Silva, George Moody, Daniel J Scott, Leo A Celi, and Roger G Mark. Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. *Computing in cardiology*, 39:245, 2012.

[Vincent *et al.*, 2008] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[White *et al.*, 2011] Ian R White, Patrick Royston, and Angela M Wood. Multiple imputation using chained equations: issues and guidance for practice. *Statistics in medicine*, 30(4):377–399, 2011.

[Yoon *et al.*, 2018] Jinsung Yoon, James Jordon, and Mihaela Schaar. Gain: Missing data imputation using generative adversarial nets. In *International Conference on Machine Learning*, pages 5675–5684, 2018.