2020 25th International Conference on Pattern Recognition (ICPR)
Milan, Italy, Jan 10-15, 2021

End-to-end Multi-task Learning of Missing Value Imputation and Forecasting in Time-Series Data

Jinhee Kim* and Taesung Kim*
KAIST
Email: {seharanul17, zkm1989}@kaist.ac.kr

Jang-Ho Choi
Electronics and Telecommunications Research Institute
Email: janghochoi@etri.re.kr

Jaegul Choo
KAIST
Email: jchoo@kaist.ac.kr

Abstract—**Multivariate time-series prediction is a common task, but it often becomes challenging due to missing data caused by unreliable sensors and other issues. In fact, inaccurate imputation of missing values can degrade the downstream prediction performance, so it may be better not to rely on the estimated values of missing data. Furthermore, observed data may contain noise, so denoising them can be helpful for the main task at hand. In response, we propose a novel approach that can automatically utilize the optimal combination of the observed and the estimated values to generate not only complete, but also noise-reduced data by our own gating mechanism. We evaluate our model on incomplete real-world time-series datasets and achieved state-of-the-art performance. Moreover, we present in-depth studies using a carefully designed, synthetic multivariate time-series dataset to verify the effectiveness of the proposed model. The ablation studies and the experimental analysis of the proposed gating mechanism show that it works as an effective denoising and imputation method for time-series classification tasks.**

I. Introduction

Interest in multivariate time-series analysis has increased in recent years, in numerous research areas such as weather forecasting [1], traffic analysis [1], patient diagnosis [2], and economics [3]. However, such data are often partially observed, limiting their applicability in real-world applications. Missing values are not only caused by the hardware problems and human error during data collection but also due to the difficulty in reliably acquiring data for, say, clinical sensor data. Even with such partially missing, incomplete data, one still has to perform downstream tasks such as predicting the patient mortality rate in an intensive care unit (ICU) [2] or forecasting financial time series [4]. Missing value estimation is a typical approach to mitigate this issue.

Recently, deep learning-based methods, which use recurrent neural networks (RNNs) have established state-of-the-art performances in predicting the multivariate time-series data with missing values. They aim at improving the prediction quality of downstream tasks via appropriate substitution of missing values. The labels of the downstream tasks are highly correlated with the underlying true distribution. In that even the observed variables may be noisy and the data with high missing rates have a small amount of information, the imputation result may also fail to obey the true distribution. However, there has not been work that raises a question of whether imputation results based on auto-encoder or adversarial learning are appropriate for predicting downstream tasks.

*Both authors contributed equally.

Therefore, we introduce a novel deep learning algorithm that gates real data, as well as missing ones, to remove possible noise from our input to downstream task modules. We further adopt the generative adversarial network (GAN) [5] architecture to estimate the true distribution of the observed data. By jointly training the downstream task module and gating mechanism with adversarial loss, our model produces realistic and helpful imputation to predict the downstream task.

In addition, we adopted the input dropout method in our model to learn the true distribution of the data via destruction-and-reconstruction process. In contrast to previous state-of-the-art work [6] that added noise to destroy the original data, we dropped out the known values and learned to reconstruct them. We empirically verified that input dropout of the original method can help imputation by modeling denoising effect. Even though previous decaying mechanisms had great success, they overlooked the importance of time gap changes that occur over time. Obviously, variables observed consecutively for long periods and those observed only once should have different reliability. Therefore, we propose a time decay mechanism that considers the sequential changes in the time interval. In addition, we discovered that the decaying of the previous hidden state vector makes the current output hidden state vector small. We address this problem with the supplementation of the hidden state vector with the current input vector.

To verify our method, we designed a synthetic dataset with a known true distribution which can be used to observe the denoising effect of the model and to analyze the missing value imputation and downstream task prediction performances. Our proposed model outperforms existing state-of-the-art models that perform well on real-world dataset.

In summary, the main contributions of the proposed method are (1) a novel end-to-end model that imputes missing values and performs downstream tasks simultaneously with gating module and input dropping, (2) the achievement of state-of-the-art performance on a real-world dataset, and (3) the synthetic dataset that can be used to validate not only imputation and downstream task performances, but also denoising effect of the model.

2020 25th International Conference on Pattern Recognition (ICPR) | 978-1-7281-8808-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICPR48806.2021.9412112

II. Related Work

Deep learning based imputation methods. Recently, many deep learning based imputation models have arisen. By adopting the RNN architecture, they took advantage of the network which can process temporal information and achieved a great success in multivariate time-series imputation. In particular, several researchers [6]–[10] proposed RNN-based models with a decaying mechanism that can effectively handle varying time gaps between observed variables. The missing inputs to the main network are imputed with temporally decayed statistics obtained from the observed variables.

Additionally, Cao *et al.* (2018) [9] improved the downstream task prediction performance by training the imputation and the downstream task simultaneously. This greatly helped the model to generate appropriate imputation results which can also make high quality predictions for the downstream task.

Imputation using generative adversarial networks. Several studies [6], [8], [10], [11] adopted GAN [5] architecture that jointly train a discriminator and a generator on the multivariate time-series imputation task. The discriminator distinguishes the generator output and the real data, while the generator deceives the discriminator by making a realistic output. With this additional architecture, researchers successfully imputed the missing values that follow the distribution of the training data. However, there is a limitation in that the reliability of the generator output was not considered, even though inaccurate imputation of missing values can degrade the downstream prediction performance.

III. Problem Formulation

In this paper, our goal is to solve multivariate time-series prediction problems, either classification or regression problems, by imputing missing values within the data in an end-to-end manner. Section IV provides detailed explanations of the proposed model. Fig. 1 illustrates an example of the annotations we explain below.

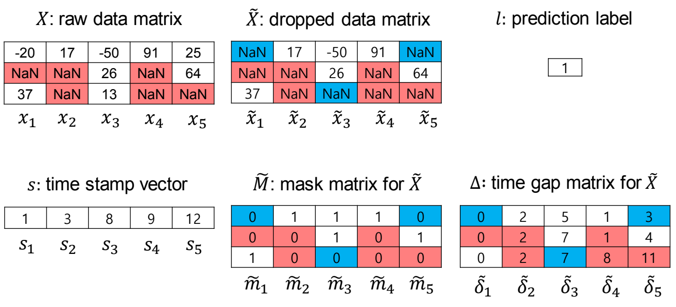
Let $\{X_n\}_{n=1}^{N}$ and $\{l_n\}_{n=1}^{N}$ denote a multivariate time-series dataset and the corresponding label, respectively. $X = \{x_1, x_2, ..., x_T\} \in \{X_n\}_{n=1}^{N}$ with label $l \in \{l_n\}_{n=1}^{N}$ is a multivariate sequence of length T, where $x_t \in \mathbb{R}^d$ is the t-th observation of X at timestamp s_t. Practically, it is common to have the timestamp between two observations, s_t, set equal for all t. We denote the i-th variable of the t-th observation as x_t^i and it is either observed or missing data. $M = \{m_1, m_2, ..., m_T\}$ is a mask matrix whose element $m_t^i \in \{0, 1\}$ indicates whether the given variable x_t^i is missing or not. $m_t^i = 1$ if x_t^i is observed, otherwise $m_t^i = 0$. Moreover, we introduce a dropped matrix $\tilde{X}$, which is generated by additionally dropping a portion of the observed variables from X to make model learn how to impute missing values. It will be covered in detail in Section IV-A. The corresponding mask matrix for $\tilde{X}$ is denoted as $\tilde{M}$.

Furthermore, we define a time gap matrix $\Delta = \{\delta_1, \delta_2, ..., \delta_T\}$ that serves time interval from the last observation for each variable of $\tilde{X}$, i.e., δ_t^i shows how long a variable $\tilde{x}^i$ has been missing consecutively until the t-th time step. Its mathematical expression is as follows:

$$\delta_t^i = \begin{cases} 0 & \textbf{if } t = 0 \\ s_t - s_{t-1} + \delta_{t-1}^i & \textbf{if } t > 1, m_{t-1}^i = 0 \\ s_t - s_{t-1} & \textbf{if } t > 1, m_{t-1}^i = 1 \end{cases}$$

If s_t is equal for all t, δ^i is totally dependent on m^i, but even in such case, it is widely used to provide model with an additional time interval information [6]–[9].

As mentioned above, this paper addresses the problem of solving downstream tasks and imputing unknown variables within the given data jointly, in other words, predicting l and X using $\tilde{X}$, $\tilde{M}$ and Δ. Although the downstream task can be either classification or regression, we mainly focused on classification problem. To this end, we propose an end-to-end GANs-based model that performs missing value imputation and downstream classification jointly.


X: raw data matrix
$\tilde{X}$: dropped data matrix
l: prediction label
s: time stamp vector
$\tilde{M}$: mask matrix for $\tilde{X}$
Δ: time gap matrix for $\tilde{X}$

X: raw data matrix

x_1	x_2	x_3	x_4	x_5
-20	17	-50	91	25
NaN	NaN	26	NaN	64
37	NaN	13	NaN	NaN

$\tilde{X}$: dropped data matrix

$\tilde{x}_1$	$\tilde{x}_2$	$\tilde{x}_3$	$\tilde{x}_4$	$\tilde{x}_5$
NaN	17	-50	91	NaN
NaN	NaN	26	NaN	64
37	NaN	NaN	NaN	NaN

l: prediction label

1

s: time stamp vector

s_1	s_2	s_3	s_4	s_5
1	3	8	9	12

$\tilde{M}$: mask matrix for $\tilde{X}$

$\tilde{m}_1$	$\tilde{m}_2$	$\tilde{m}_3$	$\tilde{m}_4$	$\tilde{m}_5$
0	1	1	1	0
0	0	1	0	1
1	0	0	0	0

Δ: time gap matrix for $\tilde{X}$

$\tilde{\delta}_1$	$\tilde{\delta}_2$	$\tilde{\delta}_3$	$\tilde{\delta}_4$	$\tilde{\delta}_5$
0	2	5	1	3
0	2	7	1	4
0	2	7	8	11

Fig. 1. An example of model inputs with colors indicating the status of each variable: ***white*** for observed variables, ***red*** for missing variables and ***blue*** for additionally dropped variables. Here, X is a time series data of length five with three features. s is a time stamp vector for X. l is the given prediction label for X, which can be either classification or regression label. $\tilde{X}$ shows input data after additional 30% of the observed variables is dropped. $\tilde{M}$ and Δ is a mask matrix and a time gap matrix for $\tilde{X}$, respectively.

IV. Proposed Method

In this section, we explain the details of the proposed methods. Our model is designed for end-to-end missing value imputation and downstream task prediction in multivariate time-series data. The model consists of two modules: the imputation and the prediction module. Each part will be explained in Section IV-A and Section IV-C, respectively. Fig. 2 shows an overview of our model.

A. Imputation using GAN

Here, we present the architecture of imputation module that aims to impute missing values within the data. We take advantage of GAN, so that our generator (G) learns the distribution of the real data under the supervision of the discriminator (D). Particularly, we used Wasserstein GAN, which is easier to train stably than original GAN, alleviating the problem of non-convergence and mode dropping phenomenon [12]. Additionally, both G and D are gated recurrent unit (GRU)-based RNNs, where G is a bidirectional RNN with *decaying cell*. The details will be explained in Section IV-B.

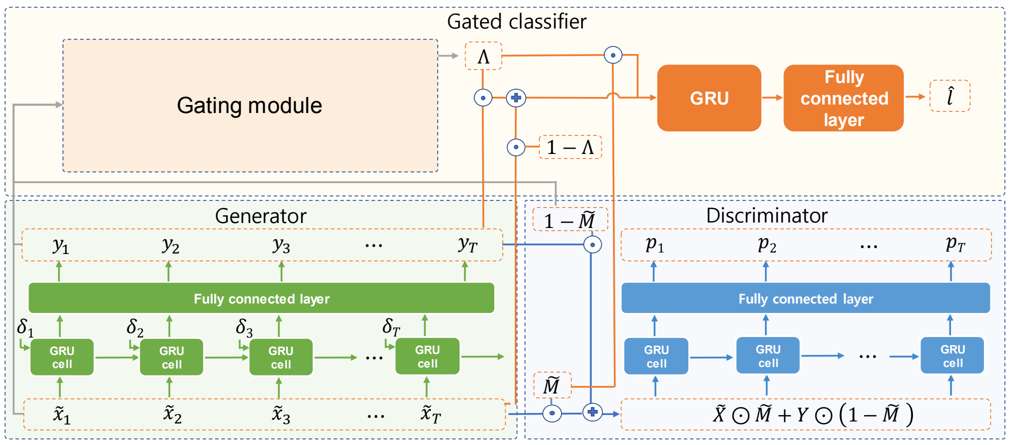


Fig. 2. An overview of the proposed model. y_t and p_t denotes the prediction output of the generator and the discriminator at time t, respectively. $\hat{l}$ indicates the prediction output of the the gated classifier for downstream task. The proposed decaying mechanism explained in Section IV-B is adopted to the generator with its cells receiving δ_t. On the other hand, the discriminator and the gated classifier is GRU without any decaying.

Given an input vector x_t at time t, G outputs the same t-th sequence vector x_t in the input space. Since it takes as an incomplete time series data, it should handle two kinds of input variables, the observed variables and the missing data. For the former, where $m_t^i = 1$, G works as an auto-encoder, which learns the distribution of the given data while reconstructing the input values. On the other hand, G does not reconstruct the same input value but estimates a new value to fill in the missing input value.

The concept of denoising auto-encoder is to reconstruct a complete input data from a partially destroyed one, essentially learning to generate a clean "noise-reduced input" [13]. Based on this idea, we remove values of a fixed ratio α of observed variables of raw data matrix X and use the corrupted data as an input data for G instead of X. We replace the "dropped" values with the average value at every iteration, with the imputation label being the values before they are removed. In this way, G is trained to alleviate noise in input data while imputing missing values.

The corrupted data $\tilde{X}$ is obtained by randomly removing known variables given X.

$$\tilde{X} \sim Drop_{\alpha}(\tilde{X}|X) \tag{1}$$

For each raw data matrix X with missing rate r, randomly chosen $100\alpha\%$ of $100(1-r)\%$ observed variables are dropped and treated as missing values, i.e., the corresponding mask values of the selected variables become zero. We compute an updated mask matrix $\tilde{M}$ whose zero values denote where missing values are in $\tilde{X}$, along with artificially dropped values. The dropping rate α is a hyperparameter and the selection of variables to drop can be either deterministic or stochastic at every iteration.

At first glance, with many values already missing, especially when the missing rate r is very high, it seems unreasonable to remove more variables. However, if both input data and imputation labels come from raw data, X, the generator cannot learn how to impute the missing values, because it has not seen the ground truth for reconstructing missing values during the training process. Therefore, we propose the dropping function, which helps the model to predict appropriate values for missing values by generating missing values that we know the ground truth: namely, $\tilde{X}$.

Given $\tilde{X}$, not X, it is possible for the model to learn how to reconstruct the values of the unknown variables, because now there exists ground truth imputation labels for the missing values. If a well-trained model with dropping mechanism produces a complete output from a known missing input, it guarantees that the model would also accurately impute missing values that originally existed in X. Even more, as stated above, G also learns to subtract out noise in input data. To conclude, dropping additional observed variables guarantees that missing values without ground truth are imputed appropriately and makes G learn how to remove noise in both observed variables and missing variables.

To sum up, G behaves as an auto-encoder for the observed variable and authentic *generator* for the other. Accordingly, in order to indicate where missing values are, we concatenate time-series data $\tilde{X}$ and the mask matrix $\tilde{M}$ and then feed it to the generator. Inspired by masked language modeling of BERT [14], we also make $\tilde{M}$ work as a mask token when G handles the missing values. With this in mind, we invert $\tilde{M}$ and then feed it to G, so that concatenated masking has the value of one at the observed and zero at the missing.

Additionally, missing data need to be imputed temporarily with appropriate values, such as mean of the variable or random noise, so G can handle them as an input. In this paper, we replace missing values with the mean value of each variable. Finally, the input of G is as follows as:

$$Z = [\tilde{X}; (1 - \tilde{M})] \tag{2}$$

where ; denotes a concatenation operator. To repeat, G receives additional information to distinguish observed variables from missing data through $\tilde{M}$. $L2$ loss is minimized to decrease the distance between prediction output of *G* and the raw time-series data. Finally, the loss of *G* for reconstructing input data is as follows:

$$L_{G,r} = \|(G(Z) - X) \odot M\|_F^2 \tag{3}$$

In addition to loss $L_{G,r}$, the generator is also trained by minimizing the adversarial loss, fooling *D* using the imputation results. The input of *D* is the observations with missing data replaced by the output of *G*. *D* learns to distinguish the observed variables from the estimated values, classifying the former as real and the latter as fake. Specifically, input variables are classified *elementwise*. That is, the output of *D* is a $d \times T$ matrix whose elements are $\in [0, 1]$. We designed *D* to distinguish input sequence elementwise, as there rarely exists a sequence in which all variables are observed in highly-missing time-series data. With well-trained *D*, the *G* learns to produce complete data that follows the distribution of real data fooling *D*. Altogether, the adversarial loss for *G* and *D* is as follows:

$$L_D = \mathbb{E}[D(G(Z)) \odot (1 - \tilde{M})] + \mathbb{E}[-D(X) \odot \tilde{M}] \tag{4}$$

$$L_{G,a} = \mathbb{E}[-D(G(Z)) \odot (1 - \tilde{M})] \tag{5}$$

$$L_G = L_{G,r} + \beta L_{G,a} \tag{6}$$

where β is a hyperparameter to balance two losses.

B. Decaying mechanism for missing values

In this section, we discuss the *decaying mechanism* for the generator in detail. The prevailing method for adopting RNN-based model to handle incomplete data is decaying input variable or hidden state vector of RNN cells using the time gap matrix δ_t^i [6]–[9]. Especially, the common equation for γ_t that denotes the decaying rate of hidden state vector considering the time gap is

$$\gamma_t^c = exp(-max(0, W_{\gamma^c}\delta_t + b_{\gamma^c}))$$

where W_{γ^c} and b_{γ^c} are model parameters to learn. It effectively captures unknown missing patterns associated with variables [7]. However, this method only uses t-th time gap values when measuring the decaying rate for $(t-1)$-th hidden state vector and fails to handle the temporal change of time gap values over time. If a variable is consecutively missing for a long time, the reliability of hidden state from the previous time step becomes low. Therefore, The missing pattern of a variable with respect to time should be considered and the hidden state vector of RNN should be decayed if a variable has been missing for a long while. Therefore, we propose a decaying method that feeds k time gap vectors, $\delta_{t-k+1:t}$, into a fully connected neural network with two linear layers followed by nonlinear functions:

$$\eta_t = max(0, W_\eta \delta_{t-k+1:t} + b_\eta) \tag{7}$$

$$\gamma_t = exp(-max(Maxpool(0, W_\gamma \eta_t^T + b_\gamma)^T)) \tag{8}$$

where W_η and W_γ are model parameters to learn, k is a hyperparameter that denotes the length of time gap sequence and *Maxpool* indicates max pooling operation. Eq. (7) and Eq. (8) include a linear transformation that regards time gap variations between different variables and different time steps, respectively.

With the assumption that there are multiple patterns of temporal time gap variations, we map η_t to a two-dimensional space instead of a vector space and perform max pooling operation, i.e., selecting one pattern out of various candidates. Accordingly, γ_t is in the same vector space with the hidden state vector of RNN.

Additionally, we introduce input decaying that downscales the input data $\tilde{x}_t$ with the rate of $(1 - \gamma_t)$ to keep the scale of gates in GRU constant. As hidden state vectors are directly related to predicting the imputation result, they should contain the information of underlying true distribution of input data and maintain a constant scale for its value. However, if the decaying rate is repeatedly multiplied, the hidden state vectors keep decreasing and the scale of the gates fluctuates. This inconsistency of the scale can lead to inappropriate prediction results. For this reason, we decay $\tilde{x}_t$ with the rate of $(1-\gamma_t)$ so that the sum of coefficients of $\tilde{x}_t$ and h_t always equals to one. As a result, the update functions of GRU with the decaying mechanism is as follows:

$$h_{t-1} = \gamma_t \odot h_{t-1}, \quad z_t = [(1 - \gamma_t); 0] \odot z_t$$
$$u_t = \sigma(W_u[h_{t-1}; z_t] + b_u), \quad r_t = \sigma(W_r[h_{t-1}; z_t] + b_r)$$
$$\tilde{h_t} = tanh(W_h[r_t \odot h_{t-1}; z_t] + b_h)$$
$$h_t = (1 - u_t) \odot h_{t-1} + u_t \odot \tilde{h}_t$$

where W_u, W_r and W_h are model parameters to learn. In Eq. (IV-B), we pad $(1 - \gamma_t)$ with zero vectors to have the shape of z_t. Note that it results in multiplying zero matrix and the concatenated $\tilde{m}_t$.

C. Gated Classifier and an end-to-end multitask learning

In this section, we introduce a novel deep learning model using the gating mechanism, *gated classifier C*, which considers the reliability of imputation results and substitutes the results partially with other values based on the reliability.

It is proposed based on the question of suitability of imputation results based on auto-encoder or generative adversarial training for predicting downstream task. If the missing rate of input data is very high, there is little information that we can extract to estimate its distribution. If so, the output of the imputation module might not be reliable. Since our prediction label is highly related to the true distribution, unreliable noisy outputs should not be used for downstream task. Therefore, we propose *gated classifier* that mixes observations and predictions of the imputation module based on the reliability of each value and uses the resulting data for classification.

Gated classifier includes *gating module* which estimates the suitability of the output of the generator compared to the raw input data for predicting downstream task. Fig. 3 shows how a gating matrix is introduced. First, we define a *gating matrix* $\Lambda \in \mathbb{R}^{d\times T}$ that denotes the degree of reliability for Y, the output of the generator. The gating value,

$\lambda_t^i \in \{0,1\}$ is an element of Λ that quantifies the reliability of corresponding variable y_t^i. We concatenate $\tilde{X}$ and Y to estimate relative confidence of Y compared to $\tilde{X}$. Since there exists missing values substituted with the mean value in Y, we also concatenate inverted $\tilde{M}$ to indicate the location of missing values similarly to the formulation of generator's input. Note that the concatenation result $[\tilde{X}^T; Y^T; (1-\tilde{M}^T)]^T$ is a $3d \times T$ matrix.

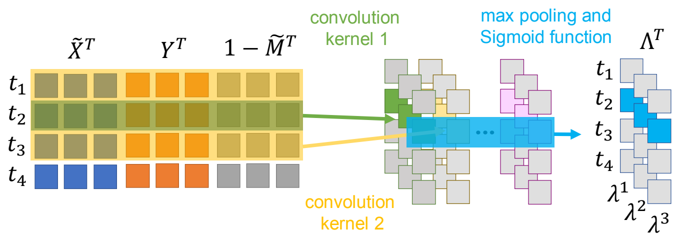


Fig. 3. The gating module takes the concatenation result of $\tilde{X}$, Y, and $(1-\tilde{M})$ as an input and compute gating values for each variable of Y.

Next, we compute the convolution of the concatenation with various kernel sizes. Regardless of the kernel size, each convolution is a mapping $\mathbb{R}^{3d\times T} \to \mathbb{R}^{d\times T}$, zero padding the input if necessary. That is, the number of its output channels always equals the number of variables of $\tilde{X}$. Each kernel sees the entire variable of raw data, generator output and inverted mask matrix at time t at once, so that gating module can make a decision about t-th variable based on the full information at time t. As a consequence, the kernel size is $(3d, *)$, where d decides how much temporal information is available. For example, if a kernel size is $(3d, 3)$, the input of the convolution is

$$\begin{gathered}[[\tilde{x}_{t-1}^T; y_{t-1}^T; (1-\tilde{m}_{t-1}^T)]^T;\\ [\tilde{x}_t^T; y_t^T; (1-\tilde{m}_t^T)]^T;\\ [\tilde{x}_{t+1}^T; y_{t+1}^T; (1-\tilde{m}_{t+1})]^T]\end{gathered}$$

where its outputs are logits for t-th observations. The number of kernels equals the number of candidate logits for each variable x_t^i, which means that each kernel generate one logits for x_t^i. After convolutions, we conduct a max-pooling operation on convolution filter outputs so as to select one logit for each variable. As a result, our gating module output has the same dimension with raw data. Finally, we apply sigmoid function to the output and it is used as a gating matrix.

After we obtain the gating matrix, Λ, we mix the generator output and raw data by the ratio of gating value.

$$S = Y \odot \Lambda + \tilde{X} \odot (1-\Lambda) \tag{9}$$

where S indicates the mixed output. It is what the gating module decides to be the best input data for performing downstream task. However, since we replaced missing values in $\tilde{X}$ with the mean value, which is zero due to the normalization, the mixing operation causes a difference in the scale of observations and the missing values where $\tilde{m}_t^i = 1$. Therefore, we compensate the shortfall with GRU cell weights. It should be filled by a ratio of $(1-\lambda_t^i)$ for each missing values and no compensation is necessary for observed variables. Thus, we multiply $(1-\lambda_t^i)$ and $(1-\tilde{M})$ to indicates the deficiency ratio to fill-in for each variable. And then, it is concatenated with S and fed to GRU. Consequently, the first-layer weights of GRU are multiplied with the ratios and S with the former compensating the latter: $W_c[s_t; \lambda_t \cdot (1-\tilde{m}_t)]$. To repeat, $W_c(\lambda_t \cdot (1-\tilde{m}_t))$ fills in the deficiency in $W_c o_t$. The GRU layer of gated classifier is then followed by a fully-connected layer that outputs the classification results.

The gated classifier and the imputation module, *G* and *D*, are jointly trained in an end-to-end manner. Accordingly, with the assumption that the prediction label is strongly related to true distribution of the data, *G* generates better results that follow true distribution while minimizing classification loss.

V. Synthetic Dataset

In this paper, we propose a synthetic dataset designed for evaluating and analyzing the multivariate time-series imputation and downstream task prediction. Here, we explain the dataset formulation in detail.

The dataset is designed for in-depth analysis of multivariate time-series imputation and downstream task performance. Many real datasets have a high missing rate and recovery is usually impossible. Moreover, we cannot guarantee that even the known values are error-free. For this reason, we designed a noisy and partially missing dataset that is error-free and of which the complete state is known as well. By using completely known data, we can easily validate the imputation accuracy of missing data and verify whether underlying true distribution is learned. In addition, we can evaluate the denoising effect of the model by comparing the model prediction with error-free state of the dataset.

The synthetic dataset that we formulated consists of three variables and binary classification label that is dependent on the variables. There are three periodic variables: The first two variables are directly related to the classification label, while the third has no relation with the classification label at all. Each variable has additional noise which is sampled from the normal distribution $\mathcal{N}(0, 0.3^2)$. We set the label to one if the first and second features without noise at $t+1$ time step are both positive. If not, the label is set to zero. We formulated imbalance dataset on purpose and use Area Under the ROC Curve (AUC) as a metric for evaluating classification performance. The ratio of label one to zero is 16.16%. Because features related to label have periodic characteristic, our dataset label is periodic as well. The numerical expressions of the three variables are as follows:

$$\begin{gathered} f_t^1 = sine(t/9) \\ f_t^2 = sine(t/7) + |sine(t/7)| - 1 \\ f_t^3 = cosine(t/11) \end{gathered}$$

We choose periodic distributions in the range of $[-1, 1]$. It is enough to have three features to estimate abilities of denoising and imputation of missing data, but if more variable is added, additional analysis is possible. It is just an example of possible synthetic datasets for addressing the missing value problem in time-series data. One can further customize the synthetic dataset according to its purpose. For example, one can use a different distribution of input variables, adjust error generating methods, or replace classification labels with regression labels to generate a different synthetic dataset.

The Pearson correlations between features we choose and labels are 0.37, 0.54 and 0.00073 for each variable, respectively. When noise is removed, the Pearson correlations become 0.4, 0.59 and 0.00073. This shows that even the first two feature that determine the label are not strongly related to the label and that noise disturbs the linear correlation between each independent variable and dependent variable. Under these circumstances, the model should learn the true distribution of features without noise to achieve high classification performance. Therefore, we can conclude that a model with higher AUC learned data distribution closer to the true distribution than not. Since the error-free and complete dataset is known, in addition to comparing downstream task performances, we can also estimate the model's ability of reducing noise within the data by measuring the distance between generated imputation result and error-free and complete state of the dataset. Without such synthetic dataset, one can only show denoising effects through indirect methods. In summary, proposed synthetic dataset has the following advantages:

- Feasible to investigate model on numerical multivariate time-series imputation and classification.
- Possible to identify whether the missing value, e.g., 80.5% in PhysioNet dataset, is really imputed well.
- Verification of the denoising effects on observed variables, which is not possible with real data.

VI. Experiments

A. Datasets

Physionet Challenge 2012 dataset. The PhysioNet dataset [2] consists of 4000 Intensive Care Unit(ICU) stay records which have 80.5% of unknown variables. The main purpose of the dataset is the development of methods for predicting the mortality rate in ICU. The dataset has 35 features except for patient information. Every record in the dataset has a 48 hours multivariate time-series sequence. Since the dataset has various time gap between each clinical measurement and sequence length, to compile the dataset as fixed size, we convert the period as an hour by rounding the time to the nearest hour.

Synthetic Dataset. We proposed synthetic dataset that consists of three features to evaluate the denoising effect of our model and analyze the effect of the additional drop with respect to the additional missing rate. Detail formulation of the dataset is described in Section V. Two of the features are relevant to the label l and the other is not.

B. Experimental Settings

We normalize each dataset with its average value and variance, so that the training process can be stable and the distance error metric can be compared between various variables in an appropriate scale. For the PhysioNet 2012 ICU dataset, we randomly choose 80% of the dataset as training data and 20% of the dataset as validation data. For the synthetic dataset, We choose one period of the label as validation data and ten periods of the label as training data. The training set size is 3947 and the validation set size is 395. Since all period has different noise from normal distribution, entire periods are not the same each other. We conduct each experiment five times and report the average value and standard deviation of performances. Each standard deviation is recorded in parentheses.

TABLE I
Imputation performances on the PhysioNet dataset.

Method	MSE	MAE
zero imputation	1.051 (0.110)	0.706 (0.007)
mean imputation	0.830 (0.007)	0.459 (0.004)
LOCF	0.727 (0.073)	0.458 (0.002)
GRU-D	1.258 (0.061)	0.824 (0.015)
GRUI	1.329 (0.909)	0.840 (0.041)
E^2GAN	0.756 (0.100)	0.555 (0.009)
BRITS	1.022 (0.110)	0.632 (0.006)
ours	**0.477 (0.065)**	**0.382 (0.239)**

C. Imputation performance

Since we cannot obtain complete real dataset, we additionally dropped a part of the validation data entirely at random to validate our imputation accuracy. We discarded 10% of the validation set of the PhysioNet dataset and the synthetic dataset, respectively. Note that the discarded values are only used for validating the imputation result and they are unseen data to the model. For the imputation task, we report mean squared error(MSE) and mean absolute error(MAE) of imputation results of additionally dropped variables. We use normalized variables with mean value and standard deviation of the training dataset to handle the different scale of each variable. Table I shows the comparative results of the imputation accuracy between our proposed method and other imputation methods, which include the statistical imputation methods and the deep learning based models. Specifically, we evaluate mean imputation, zero imputation, last observation carried forward(LOCF) method, GRU-D [7], BRITS [9], GRUI [8], and E^2GAN [6] as baseline methods. The mean imputation method uses the average value of each time series rather than the average value of entire time series data. Our proposed method outperforms the other imputation methods. Except for the proposed model, LOCF method achieve the best imputation accuracy. With the assumption that the patient status does not change dramatically in a short-period, the success of imputation of this method is reasonable. With the same premise, mean imputation succeeds higher accuracy than most of other methods as well. However, these imputation methods cannot capture the temporal information and correlation between each variable. Since GRU-D does not target the imputation task but downstream task, it fails to predict appropriate imputation result.

TABLE II
MORTALITY CLASSIFICATION PERFORMANCES ON PHYSIONET DATASET. AUC INDICATES THE AREA UNDER THE ROC CURVE, A METRIC FOR BINARY CLASSIFICATION.

Method		AUC
Non-RNN	Zero	0.8319 (0.0070)
	Mean	0.8189 (0.0085)
	LOCF	0.8380 (0.0041)
2-stage	GRUI	0.8072 (0.0093)
	E^2GAN	0.8329 (0.0092)
End-to-end	GRU-D	0.8297 (0.0069)
	BRITS	0.8575 (0.0040)
	Ours	**0.8649 (0.0020)**

TABLE III
ABLATION STUDY RESULT OF OUR PROPOSED METHOD ON THE PHYSIONET DATASET.

Models	AUC
Ours	**0.8649 (0.0020)**
Ours w/o adversarial learning	0.8621 (0.0053)
Ours w/o end-to-end learning	0.8488 (0.0050)
Ours w/o dropping mechanism	0.8580 (0.0045)
Ours w/o input decaying	0.8548 (0.0079)
Ours w/o hidden vector decaying	0.8505 (0.0066)
Ours w/o gating module 1	0.8555 (0.0073)
Ours w/o gating module 2	0.8477 (0.0029)

D. downstream task performance

Table II shows the mortality prediction performances of various methods. Since our proposed model imputes the missing values and predicts the downstream task simultaneously, we use the dataset whose values are partially discarded. The imputation models such as GRUI and E^2GAN originally do not have a classifier. Thus, we conduct an additional training step on the downstream task after imputation for those models. The additional classifier consists of a GRU layer and a fully-connected layer. The last output hidden state vector of the GRU layer is fed into the fully-connected layer to predict the downstream task label. Our model performed better than other baselines. This shows that the proposed model creates an imputation result that can be helpful in predicting the downstream task.

Although the zero imputation method shows the lower imputation performance than the mean imputation method, it outperforms the mean imputation method on the mortality prediction task. This means that it is advantageous to leave the unknown variables empty rather than to use an inappropriate imputation result when performing the downstream task. Furthermore, the same pattern appears between BRITS and E^2GAN, where the former is an end-to-end learning method and the latter is a 2-stage method. This also suggests that learning in an end-to-end manner is helpful to generate appropriate imputation result in predicting the downstream task label.

E. Ablation study

We evaluate the contributions by ablation study on the PhysioNet dataset. Table III shows the results of the ablated proposed models. We compare the following ablated models:

- Ours: the proposed model
- Ours w/o adversarial learning: the model without adversarial loss
- Ours w/o end-to-end learning: the model with two-stage learning of imputation and classification modules (not end-to-end)
- Ours w/o dropping mechanism: the model without the proposed additional input dropping
- Ours w/o input decaying: the model that does not decay the inputs
- Ours w/o hidden vector decaying: the model that does not decay the hidden vectors
- Ours w/o gating module 1: the model without the gating module, whose classifier input is raw data with missing values imputed with the generator output
- Ours w/o gating module 2: the model without the gating module, whose classifier input is same with the gated classifier, $[\tilde{X}; Y; (1 - \tilde{M})]$

Table III shows the mortality prediction performances of ablated models on the PhysioNet dataset. Our model outperforms ablated models, indicating that entire modules of the proposed model have important roles to predict downstream task. The proposed model shows the best performance among all models.

When the GAN architecture is ablated, the model fails to generate the realistic imputation, while causing degraded performance, which shows that adversarial loss makes the model produce realistic imputation. Our model outperforms Ours w/o AD, which shows that our model generates a clean noise-reduced imputation by learning a reconstruction of a complete data from a partially destroyed one. When the proposed decaying mechanism is replaced or removed, the model cannot consider the time gap between observations appropriately. Without gating module, the model performance is declined significantly, which suggests that the proposed model successfully estimates the balance between the generated output and raw data, generating the optimal data to predict the label. Even when the input fed into the classifier is identical to that of the gating module, the model without the module fails. Especially when end-to-end learning is ablated from the model, the downstream task prediction performance is significantly decreased. This demonstrates that it is essential to create an appropriate imputation result for the downstream task through end-to-end learning in order to improve the prediction performance.

F. Experiments on additional dropping

Table IV shows that removing the additional dropping of the data from our model degrades the prediction performance. Furthermore, regardless of the additional missing rate, the additional dropping increases the prediction performance significantly. This result represents that additional dropping helps the model to reproduce reasonable imputation results for unknown variables.

TABLE IV
EXPERIMENTAL RESULTS WITH VARIOUS ADDITIONAL MISSING RATE α ON THE SYNTHETIC DATASET. WE REPORT THE AUC SCORE OF THE CLASSIFICATION TASK AND THE SQUARED DISTANCE ERROR ON THE DATA WITH NOISE AND WITHOUT NOISE, NAMED MSE WITH NOISE WITH NOISE AND MSE W/O NOISE, RESPECTIVELY.

		Deterministic dropping					Dynamic dropping				
Additional missing rate	0%	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
AUC	0.939 (0.005)	0.949 (0.007)	**0.950 (0.006)**	0.945 (0.010)	0.944 (0.009)	0.942 (0.008)	0.948 (0.008)	0.945 (0.006)	0.942 (0.005)	0.941 (0.005)	0.940 (0.006)
MSE w/ noise	1.106 (0.031)	1.012 (0.042)	1.021 (0.045)	1.000 (0.071)	0.988 (0.070)	0.992 (0.077)	1.046 (0.037)	1.004 (0.036)	0.994 (0.050)	0.984 (0.068)	**0.983 (0.076)**
MSE w/o noise	0.911 (0.026)	0.818 (0.041)	0.827 (0.043)	0.805 (0.071)	0.794 (0.069)	0.799 (0.076)	0.851 (0.037)	0.810 (0.034)	0.800 (0.049)	0.790 (0.067)	**0.790 (0.074)**

In the view of missing data imputation, the higher the additional missing rate is, the higher the ability to predict unknown variables is. Additionally, MSE w/o noise is smaller than MSE with noise, indicating that our proposed model successfully captures the true distribution removing the noise in the data.

VII. CONCLUSION

In this paper, we propose a novel end-to-end model to impute the missing values and predict downstream task simultaneously. The model can capture the true distribution of data through joint training of the GAN-based imputation module and the gated classifier. The gating module generates the optimal combination of the observed and the estimated values to generate a complete, noise-reduced data for downstream task modules. In addition, we propose a decaying mechanism that can handle the temporal change of time gap of observation and tackle down-sizing problem of output hidden state vector caused by the decaying mechanism to impute missing values in an appropriate scale. Furthermore, we designed a synthetic dataset, which can be used to verify denoising effect of the model. We empirically evaluate our model on both the real-world dataset and the synthetic dataset, showing the effectiveness of the proposed model.

ACKNOWLEDGMENT

This work was supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2018-0-00219, Space-time complex artificial intelligence blue-green algae prediction technology based on direct-readable water quality complex sensor and hyperspectral image).

REFERENCES

[1] X. Tang, B. Gong, Y. Yu, H. Yao, Y. Li, H. Xie, and X. Wang, "Joint modeling of dense and incomplete trajectories for citywide traffic volume inference," in *Proc. the International Conference on World Wide Web (WWW)*, 2019, pp. 1806–1817.

[2] I. Silva, G. Moody, D. J. Scott, L. A. Celi, and R. G. Mark, "Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012," in *2012 Computing in Cardiology*. IEEE, 2012, pp. 245–248.

[3] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PloS one*, vol. 12, no. 7, 2017.

[4] V. Kodogiannis and A. Lolis, "Forecasting financial time series using neural network and fuzzy system-based techniques," *Neural computing & applications*, vol. 11, no. 2, pp. 90–102, 2002.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2014, pp. 2672–2680.

[6] Y. Luo, Y. Zhang, X. Cai, and X. Yuan, "E 2 GAN: end-to-end generative adversarial network for multivariate time series imputation," in *Proc. the International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 2019, pp. 3094–3100.

[7] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific reports*, vol. 8, no. 1, pp. 1–12, 2018.

[8] Y. Luo, X. Cai, Y. Zhang, J. Xu *et al.*, "Multivariate time series imputation with generative adversarial networks," in *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 1596–1607.

[9] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, "Brits: Bidirectional recurrent imputation for time series," in *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 6775–6785.

[10] X. Tang, H. Yao, Y. Sun, C. C. Aggarwal, P. Mitra, and S. Wang, "Joint modeling of local and global temporal dynamics for multivariate time series forecasting with missing values." in *Proc. the AAAI Conference on Artificial Intelligence (AAAI)*, 2020, pp. 5956–5963.

[11] J. Yoon, J. Jordon, and M. Schaar, "Gain: Missing data imputation using generative adversarial nets," in *Proc. the International Conference on Machine Learning (ICML)*, 2018, pp. 5689–5698.

[12] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. the International Conference on Machine Learning (ICML)*, 2017, pp. 214–223.

[13] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. the International Conference on Machine Learning (ICML)*, 2008, pp. 1096–1103.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT (1)*, 2019.