STING: Self-attention based Time-series Imputation Networks using GAN

Eunkyu Oh, Taehun Kim, Yunhu Ji, Sushil Khyalia
Samsung Research
Seoul, Republic of Korea
{eunkyu1.oh, taehun33.kim, yunhu.ji, sus.hil}@samsung.com

2021 IEEE International Conference on Data Mining (ICDM) | 978-1-6654-2398-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICDM51629.2021.00155

***Abstract*—Time series data are ubiquitous in real-world applications. However, one of the most common problems is that the time series could have missing values by the inherent nature of the data collection process. So imputing missing values from multivariate (correlated) time series is imperative to improve a prediction performance while making an accurate data-driven decision. Conventional works for imputation simply delete missing values or fill them based on mean/zero. Although recent works based on deep neural networks have shown remarkable results, they still have a limitation to capture the complex generation process of multivariate time series. In this paper, we propose a novel imputation method for multivariate time series, called STING (Self-attention based Time-series Imputation Networks using GAN). We take advantage of generative adversarial networks and bidirectional recurrent neural networks to learn the latent representations of time series. In addition, we introduce a novel attention mechanism to capture the weighted correlations of a whole sequence and avoid the potential bias brought by unrelated ones. The experimental results on three real-world datasets demonstrate that STING outperforms the existing state-of-the-art methods in terms of imputation accuracy as well as downstream tasks with the imputed values therein.**

***Index Terms*—time-series imputation, self-attention, generative adversarial networks, bidirectional RNN**

I. Introduction

Multivariate time series data are everywhere and continuously generated every day. Many real-time application domains have analyzed these signals for predictive analytics. For instance, there are forecasting a stock price [1] and predicting medical a diagnosis of patients [2], [3]. However, it is inevitable that the time series data contain missing values for some reasons such as certain data features being collected later or records being lost due to equipment damage or communication error. These kinds of missing data significantly degrade a model quality and even make wrong judgments by introducing a substantial amount of bias [4]. So, imputing missing values in time series has become a challenging issue for making accurate data-driven decisions.

Conventional methods of imputing missing values can be classified into two categories: discriminative methods and generative methods. The former includes Multivariate Imputation by Chained Equations (MICE) [5] and MissForest [6] while the latter includes deep neural networks based algorithms (e.g., Denoising Auto Encoders (DAE) and Generative Adversarial Networks (GAN)) [7], [8]. However, these methods have been developed for non-time series so that they rarely account for the temporal dependencies between observations in time series. Recent state-of-the-art works for time series imputation are based on Recurrent Neural Networks (RNN) or GAN [2], [9]–[11]. They capture temporal dependencies with various aspects of observed (or missing) data characteristics such as time decay, feature correlations, and temporal belief gates.

Inspired by the existing deep generative models showing remarkable advancement, we propose a novel imputation method for multivariate time series data, called STING (Self-attention based Time-series Imputation Network using GAN). We take generative adversarial networks as the base architecture which could estimate a true data distribution while imputing the original incomplete time series. To be specific, the generator learns the underlying distribution to accurately impute the missing values, and the discriminator learns to distinguish between the observed and imputed elements. In addition, we propose a novel attention mechanism for imputation to pay selective attention to highly related information in each time step. This allows efficient training when the time series sequence is long and the time interval between two observations is large. The experiment results on three real-world datasets show that STING outperforms the state-of-the-art methods in terms of imputation performance. Our model is also superior to the baselines in the post-imputation task as an indirect measure of imputation performance.

II. Related Work

Che et al. [2] proposed Gated Recurrent Unit with Decay (GRU-D) that predicts the target labels while imputing missing values in a healthcare dataset by introducing a decay rate to control decay factors. As a similar work, Cao et al. [10] proposed an RNN-based method, called Bidirectional Recurrent Imputation for Time Series (BRITS) to combine historical-based estimation and feature-based estimation for feature correlations by applying the learning strategy that makes missing values get delayed gradients. However, as these models have the fundamental premise that missing patterns of data are often correlated with target labels (i.e., informative missingness), they take a unified approach by integrating the imputation and prediction task into one process. This makes a model depend on the integrity of the labels, so that it is hard to be used in unsupervised settings without labels [2] or when the labels are not clear. Unlike [2] and [10], our model is independent of target labels during any process.

978-1-6654-2398-4/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDM51629.2021.00155

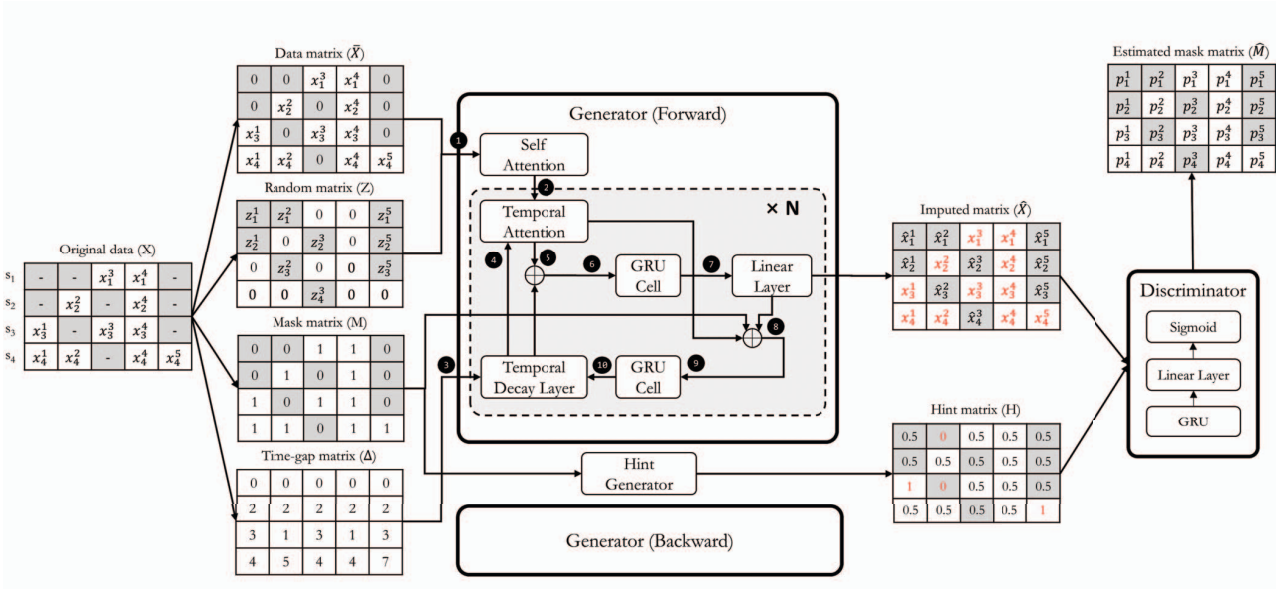

Fig. 1: The architecture of STING. The gray-colored elements in each matrix correspond to the missing values of the original data. The data flow of the backward generator is omitted as it has the same internal structure as the forward one. The timestamps in this example are $S = [0, 2, 3, 7]$.

Luo et al. [9] proposed a GAN-based imputation model, called GAN-2-stages. Gated Recurrent Unit for Imputation (GRUI) is proposed to decay the influence of past observations according to how long the time has passed in an irregular time interval. They further train the input "noise" of the generator to find the best noise from the latent input space. As the follow-up work, Luo et al. [11] proposed End-to-End Generative Adversarial Network (E^{2}GAN), which exploits compression and reconstruction strategies to avoid the "noise" optimization stage by using a denoising auto-encoder [7]. However, they hold limitations of the faster convergence of discriminator and self-feed training where incorrect outputs during training could continue to be reflected. Unlike the previous works, we take a bidirectional delayed gradient [10] for efficiently training. Also, our discriminator tries to solve a more specific problem for stable adversarial learning.

III. Method

The two generators (Gs) of STING take four input matrices from an original time series X in forward and backward directions as shown in Fig. 1. Both directions complement the lack of information problem in a unidirectional GRU structure suffering from missing values. A mask matrix M serves to inform G of information about which elements of X are observed values or missing values. In addition, a time-gap matrix Δ has temporal information about how much G should refer to the previous hidden states in the case where there are missing values arbitrarily. Specifically, $\delta_t^d \in \Delta$ denotes a time interval of the corresponding element from the last observation to the current timestamp s_t. After an imputed matrix $\hat{X}$ is generated, that is refined by filling in the observed values of X as we do not need to generate x_t^d we already know. Then, the refined matrix is forwarded to a discriminator.

Meanwhile, the discriminator (D) that takes the complete matrix $\hat{X}$ as an input learns to distinguish which elements are originally observed values or not. In this process, a hint matrix H is also provided as an additional input, which informs D of certain parts of M to enforce its attention on the particular components. H reveals some parts of M with $h_t^d = 0$ or $h_t^d = 1$, while $h_t^d = 0.5$ implies nothing about m_t^d where D's learning could be concentrated. If we do not provide enough information about M to D, several distributions that G could reproduce would all be optimal with respect to D. In other words, we cannot guarantee that G learns the desired distribution uniquely defined by original data without the hint mechanism. The detailed proofs and theoretical analysis for this mechanism are presented in [12]. In practice, we could control the amount of hint information in the hint generator to control the learning pace between G and D. Finally, the output of D is $\hat{M}$ where each value represents the probability p_t^d that each element of the input matrix $\hat{X}$ is an observed value. We adopt the strategy that D attempts to distinguish each element, not an entire input matrix itself, which allows D to focus more on the specific classification problem.

By jointly training G and D via a min-max game, G is able to learn the underlying distribution of the original data X while imputing the missing values appropriately so that D cannot distinguish them. The ideal result for G is that each p_t^d corresponding to the generated fake value $\hat{x}_t^d$ is set to 1. In contrast, D aims to set the fake value to 0, otherwise 1 by distinguishing between the real value x_t^d and fake value $\hat{x}_t^d$, which equals to output the same matrix as M.

A. Generator

Each G consists of an advanced GRU structure with two types of attention layers (i.e., self-attention and temporal attention), a temporal decay layer, and a double GRU-cell.

1) Attention: An attention mechanism aims to learn structural dependencies between the different coordinates of input data by finding the most relevant parts of the input given a query value and generating a query-specific representation [13]–[16]. We adopt a scaled dot-product attention [17] as $softmax\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$, where Q represents queries, K is keys, V is values, and the scale factor $\sqrt{d_k}$ is to avoid overly large values of an inner product. In particular, a self-attention module calculates the attention scores between different positions of its own sequence (i.e., Q=K=V). To allow a model to jointly attend to the information from different representation sub-spaces at different positions, we further adopt a multi-head attention mechanism [17] with four parallel attention heads.

In our work, the attention mechanism, which provides more targeted clues to an imputation problem, is adopted for two reasons. First, it allows our model to pay attention to time steps with a relatively similar pattern, not just a temporal order. It is suitable especially for time series data with a periodicity (e.g., traffic prediction, etc.). Second, it can obtain sufficient information from an entire sequence, not only a few time steps back and forth. The self-attention module converts the input sequences to context vectors for considering quantitative correlations of the whole sequence. Meanwhile, the temporal attention module reflects the correlations between the hidden states and context vectors.

2) Temporal Decay Layer: This can control the influence of past observations for irregular time intervals in a GRU, which is motivated by [2], [9]. Decay rates should be learned from the data because they differ from variable to variable based on the underlying properties associated with the variables as the missing patterns are unknown and could be complex. So, the decay rate vector γ_t at t is defined as

$$\gamma_t = \exp\left\{-\max\left(0,\ W_\gamma \delta_t + b_\gamma\right)\right\} \tag{1}$$

where W_γ and b_γ are trainable parameters. The exponential negative rectifier is used to keep each decay rate monotonically decreasing in the range between 0 and 1, which can be replaced with others if only this condition is met. Intuitively, this has the effect of decaying the extracted features from hidden states rather than utilizing raw input variables directly. They are learned to adjust how much the previous information is decayed before the hidden states enter a GRU-cell as

$$h'_{t-1} = \gamma_t \odot h_{t-1} \tag{2}$$

where $\odot$ denotes an element-wise multiplication, and h_{t-1} is previous hidden states.

3) Double GRU: The two GRU-cells in our model are not completely independent but have different purposes. One as main-cell is responsible for organizing and transferring the decayed previous information h'_{t-1}. The other as generation-cell generates a time series vector $\hat{x}_t$. This has its own hidden states where the temporal decay is not applied.

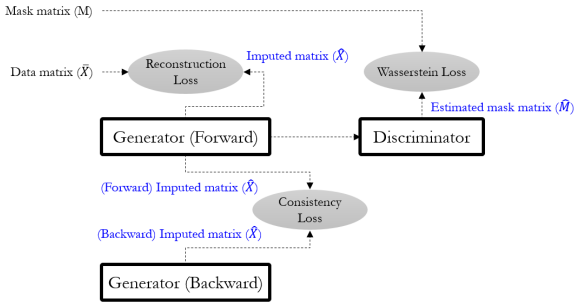

Fig. 2: Three types of loss functions.

4) Reconstruction Loss: This is the measure of how well observed values are reconstructed iteratively by comparing the original data X and imputed matrix $\hat{X}$ before being refined. This loss helps guide the generated data closer to the real distribution of ground-truth data by inducing G to keep the observed values used as conditions. To be specific, if the generated value $\hat{x}_t^d$ is originally an observed value (i.e., $m_t^d = 1$), we can directly obtain reconstruction loss $\mathcal{L}_R$, then replace it with the original value x_t^d when passing to the next step. In contrast, when we do not know the ground-truth of $\hat{x}_t^d$ (i.e., $m_t^d = 0$), we delay the comparison until the future observation in the following steps. This strategy called a delayed gradient [10] considers missing values as variables, not constants. To sum up, $\mathcal{L}_R$ is measured with L2-distance as

$$\mathcal{L}_R = \mathbb{E}\left[m_t^d\left(x_t^d - \hat{x}_t^d\right)^2\right] \tag{3}$$

5) Consistency Loss: This enforces the generations in opposite directions to be consistent by helping two Gs interact with each other. Adopting consistency loss $\mathcal{L}_C$ enhances a learning effect and improves stability by leading to the single result from two complete matrices. In addition, we can obtain a less delayed gradient due to the closer observation among two directions. We compute $\mathcal{L}_C$ with L1-distance before the generated matrices are refined for the observed values as

$$\mathcal{L}_C = \mathbb{E}\left[\left|\hat{x}_{t,forward}^d - \hat{x}_{t,backward}^d\right|\right] \tag{4}$$

where $\hat{X}_{forward}$ and $\hat{X}_{backward}$ are generated matrices from the forward G and backward G.

6) Wasserstein Loss: Wasserstein loss proposed in Wasserstein GAN (WGAN) [18] improves learning stability for GANs by getting out of a mode collapse problem. However, Our problem settings are quite different from the original WGAN. The final estimation of D is not a scalar value but elements of a matrix. We derive our adversarial loss for G $\mathcal{L}_W$ only for the case that input elements are missing values as

$$\mathcal{L}_W = -\mathbb{E}\left[\hat{m}_t^d \mid m_t^d = 0\right] \tag{5}$$

By integrating the three losses defined from (3) to (5) as shown in Fig. 2, we define the total loss of generators $\mathcal{L}_G$. In practice, $\mathcal{L}_W$ and $\mathcal{L}_R$ are two for each direction, but only one is described for convenience as

$$\mathcal{L}_G = \lambda_r \mathcal{L}_R + \lambda_c \mathcal{L}_C + \mathcal{L}_W \tag{6}$$

B. Discriminator

D consists of a GRU module, a fully connected layer, and a sigmoid activation which produces the probability that each element is real or fake. The structure of D is simple compared to G for stable adversarial learning as we experimentally found that D is relatively easy to converge. Our adversarial loss for D $\mathcal{L}_D$ is modified from WGAN to adopt to a matrix estimation. This loss leads D to distinguish whether each element of the input matrix $\hat{X}$ is an observed value or imputed value as

$$\mathcal{L}_D = \mathbb{E}\left[\hat{m}_t^d \mid m_t^d = 0\right] - \mathbb{E}\left[\hat{m}_t^d \mid m_t^d = 1\right] \tag{7}$$

where the first term indicates when D takes generated values and the second term is the case of observed values as inputs.

C. Searching for an Optimal Noise

Most of the research using GAN [19] aims to generate various realistic samples by varying a random vector z called "noise". z is randomly sampled from the latent space such as a Gaussian distribution, which implies that as z changes, a generated sample $G(z)$ could change a lot. Even though $G(z)$ follows the true distribution of original data, the degree of similarity could not be large enough. To increase the degree, we additionally search for an optimal noise z' [9], [20]–[22]. Since we already know some given conditions (i.e., observed values), we train z in an inference phase to match the observed values. This strategy allows G to generate the samples more suitable for the original distribution, which means that the missing values are also imputed more appropriately. We use the same loss as training G in (6). That is, the gradient $-\frac{\partial \mathcal{L}_G}{\partial z_t}$ repeatedly updates z_t to find a more suitable noise. After some iterations, we obtain the mean of two opposite matrices with the optimal noise z'_t as the final result.

IV. Experiments and Analysis

A. Datasets

We evaluate our proposed method on three real-world datasets from different domains for practical experiments.

PhysioNet Challenge 2012 Dataset (PhysioNet) [23] aims to develop methods for a patient-specific prediction of in-hospital mortality. It consists of 12,000 multivariate clinical time series from intensive care unit (ICU) stays. We use training set A (4,000 ICU stays). The preprocessed dataset has a total of 192,000 samples. Each sample contains 37 variables such as HR, Na, Lactate, etc. over 48 hours. This dataset has a high missing rate (80.53%).

KDD CUP Challenge 2018 Dataset (Air Quality) accessible from UCI Machine Learning Repository [24], is used in KDD CUP Challenge 2018 [25] to accurately forecast air quality indices (AQIs) of the future 48 hours. The time period is from March 1, 2013 to February 28, 2017 and the total number of samples is 420,768. 12 variables such as PM2.5, PM10, SO2, etc. were measured every hour from 12 monitoring stations in Beijing. The missing rate is 1.43%.

Gas Sensor Array Temperature Modulation Dataset (Gas Sensor) accessible from UCI Machine Learning Repository, contains 14 temperature-modulated metal oxide (MOX) gas sensors exposed to dynamic mixtures of carbon monoxide (CO) and humid synthetic air in a gas chamber for 3 weeks [26], [27]. We use one day of the whole dataset and the number of samples is 295,704. Each sample consists of 20 variables including CO concentration inside the gas chamber. Unlike the other datasets, all samples are completely observed.

B. Baselines

There are three types of models: statistics-based (Stats-based), machine learning-based (ML-based), and neural network-based (NN-based) models. We implemented ML-based models based on the python packages $sklearn$ and $fancyimpute$. The experimental settings of the NN-based models were set according to each corresponding paper.

- **Mean** simply fills missing values with the global means of corresponding variables.
- **Previous Value Filling (Prev)** fills missing values with the previously observed values.
- **KNN** [28] uses K-Nearest Neighbors with the mean of similar 10 samples.
- **Matrix Factorization (MF)** [29] directly factorizes an incomplete matrix into two low-rank matrices solved by gradient descent, then imputes the missing values by matrix completion.
- **Multiple Imputation by Chained Equations (MICE)** [5] models each feature with missing values as the function of other features iteratively.
- **Generative Adversarial Imputation Nets (GAIN)** [12] imputes missing values conditioned on what is actually observed by adopting GANs.
- **Gated Recurrent Unit with Decay (GRU-D)** [2] based on GRUs, imputes missing values by using a decay mechanism for irregular time intervals.
- **End-to-End Generative Adversarial Network ($\mathbf{E^2}$GAN)** [11] is based on GANs and the generator has an auto-encoder structure.
- **Bidirectional Recurrent Imputation for Time Series (BRITS)** [10] adapts a bidirectional RNN and introduces a delayed gradient for imputing the missing values.

C. Parameter Setting

A min-max normalization was applied on datasets. The experiments were repeated 10 times and the mean of accuracy was reported. Two Gs were pre-trained for 10 epochs with $\mathcal{L}_R$ and $\mathcal{L}_C$ in (3) and (4) because we experimentally found that if Gs are trained in advance, they could converge faster and get better performance. Then, Gs and D were updated one by one alternately at every iteration. We trained our model with an Adam optimizer. Learning rates of Gs and D were 0.001 and 0.0001. The hint ratio given to D was fixed at 0.1. The batch size was set to 128. As hyper-parameters for (6), λ_r and λ_c were set to 10 and 1. We implemented our model in PyTorch and performed all training on a single 2080Ti GPU with 11GB of RAM.

TABLE I: Overall model performances in terms of RMSE measured after a min-max normalization. The results of the best performing baseline and the best performer are underlined and boldfaced.

		PhysioNet	Air Quality	Gas Sensor
Stats-based	Mean	0.1037	0.1203	0.2000
	Prev	<u>0.0825</u>	<u>0.0336</u>	<u>0.0619</u>
ML-based	KNN	0.1137	0.1157	<u>0.0379</u>
	MF	0.1033	0.1008	0.1355
	MICE	<u>0.0986</u>	<u>0.0659</u>	0.0726
NN-based	GAIN	0.1300	0.1148	0.0722
	GRU-D	0.0830	0.0377	0.0630
	E^2GAN	0.0638	0.0379	<u>0.0374</u>
	BRITS	<u>0.0553</u>	<u>0.0301</u>	<u>0.0374</u>
	STING	**0.0531**	**0.0238**	**0.0153**

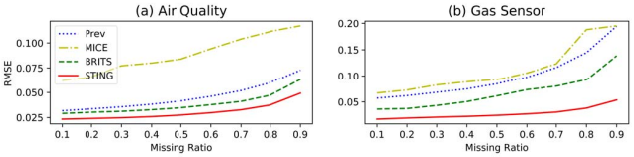

Fig. 3: Direct imputation performance on two datasets while varying the missing ratio.

D. Direct Evaluation of Imputation Performance

We randomly eliminate a certain ratio of observed values to be imputed, which are also used as the ground-truth. In consequence, the remaining observed values are used for training a model. After obtaining imputed data via an inference phase, performance is measured in terms of Root Mean Squared Error (RMSE). We have two experiments for the direct evaluation. In the first experiment, we evaluate performance while fixing a ground-truth ratio of observed values to 20%. In the second one, we vary the ground-truth ratio from 10% to 90%.

Table I summarizes the results of the first experiment. Although Prev is a simple method, this has fairly high performances compared with other models which take a long time and are complicated. In particular, for the dataset with little change over time (e.g., Air Quality), filling with previous values could have a great effect with little cost. KNN shows high performance on Gas Sensor, but is not stable on other datasets. On the other hand, MICE stably shows good performances on all datasets. GAIN does not perform well compared with other NN-based models as there is no suitable learning strategy for time series. GRU-D shows a similar tendency of performance to Prev because this learns the ratio between a mean and previous value for the target missing value. STING achieves the best performances on all datasets. In comparison with the best performing baseline (i.e., BRITS), the error improvement rates for three datasets are 4.0%, 21.0%, and 59.1%, which indicates our model stands out more on the less dependent dataset for previous values (e.g., Gas Sensor).

Fig. 3 shows the performance results on two datasets when the missing rate of observed values is varied. It is notable that

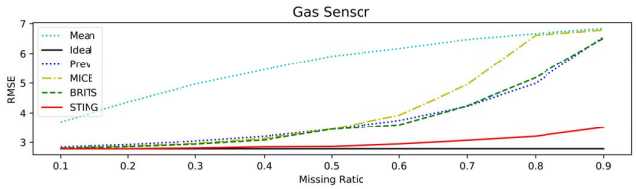

Fig. 4: Indirect imputation performance on Gas Sensor

we exclude PhysioNet which has a high missing rate (80.53%) because there is little change in the ratio. STING achieves the best performance in all conditions, showing a relatively slow increase in error.

E. Indirect Evaluation of Imputation Performance

This experiment aims to indirectly evaluate imputation performance by the results of solving a downstream task. If the distributions of original data and imputed data are similar, their downstream task results would be similar. We only exploit Gas Sensor because only a complete dataset provides a lower bound of prediction error through the ideal imputation model by which all missing values are imputed with the ground-truth. We do not train the imputation and prediction tasks at the same time because our purpose is to measure imputation performance, not to improve prediction performance. The procedure is as follows. The complete dataset is initially divided into 80% train data and 20% test data. We first train the regression model that predicts CO concentration as a target, which consists of a simple GRU with two layers and a dropout with 0.3, and a fully connected layer. Then, we randomly eliminate a certain ratio of the train and test data after removing the labels. Each imputation model learns the distribution of the train data, then imputes the missing values in the test data. Thereafter, the already trained regression model predicts the target of the test data. Finally, we measure RMSE between the predicted target values and actual ones. If the result of imputed test data is similar to the one of original test data (i.e., imputed by the ideal model), they could have a similar distribution.

Fig. 4 shows the RMSE results of the regression task while varying a missing ratio on the original data. Interestingly, there is a clear change before and after 0.5 while increasing dramatically afterward. As the ideal model is able to impute the test data regardless of any missing ratio, it has the lowest and constant results. Mean shows the worst performance, showing how inefficient imputing with mean values is on time series. On the other hand, STING achieves the best performance at any ratio while maintaining the closest results with the lower bound. In addition, STING shows a relatively small increase in error at 0.9 despite the poor condition.

F. Ablation Study

The potential sources of efficacy for STING are two attention modules, an optimal noise z', and a backward generator. To understand how each key feature affects performance improvement of STING, we conduct an ablation study for direct

TABLE II: Ablation study on an attention mechanism, an optimal noise, and a backward generator. Each rate of error increase is in parentheses.

	PhysioNet	Air Quality	Gas Sensor
STING	0.0531	0.0238	0.0153
w/o attention	0.0559 (5%)	**0.0294 (23%)**	**0.0355 (132%)**
w/o optimal z	0.0553 (4%)	0.0250 (5%)	0.0193 (26%)
w/o backward	**0.0583 (10%)**	0.0269 (13%)	0.0189 (23%)

evaluation. We experiment with three models by excluding just one function from STING on three datasets. The ground-truth ratio is set to 20% of observed values, and RMSE is measured.

Table II lists the results where the RMSE increases in all cases. The result of removing attention modules shows the highest error increase on Gas Sensor and Air Quality, which implies that the proposed attention modules perform a relatively important function in STING. That is, the process of learning correlations of the whole sequence could produce significantly important information in time series datasets. On the other hand, searching for z' has a relatively minor effect, which indicates it plays a complementary role.

V. Conclusion

In this paper, we proposed STING, a novel imputation method for multivariate time series data based on generative adversarial networks and bidirectional recurrent neural networks to learn the latent representations of time series. We also proposed novel self-attention and temporal attention mechanisms to capture the weighted correlations of a whole sequence and prevent the potential bias from unrelated time steps by paying attention to certain periodic patterns or related time steps. The various experiments on real-world time series datasets show that STING outperforms the traditional state-of-the-art methods in terms of both imputation accuracy and downstream performance with the imputed values.

References

[1] T.-J. Hsieh, H.-F. Hsiao, and W.-C. Yeh, "Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm," *Applied soft computing*, vol. 11, no. 2, pp. 2510–2525, 2011.

[2] 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.

[3] Z. Liu and M. Hauskrecht, "Learning linear dynamical systems from multivariate time series: A matrix factorization based framework," in *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 2016, pp. 810–818.

[4] R. Lall, "How multiple imputation makes a difference," *Political Analysis*, vol. 24, pp. 414–433, 2016.

[5] S. v. Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in r," *Journal of statistical software*, pp. 1–68, 2010.

[6] D. J. Stekhoven and P. Bühlmann, "Missforest—non-parametric missing value imputation for mixed-type data," *Bioinformatics*, vol. 28, no. 1, pp. 112–118, 2012.

[7] 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*, 2008, pp. 1096–1103.

[8] L. Gondara and K. Wang, "Multiple imputation using deep denoising autoencoders," *ArXiv*, vol. abs/1705.02737, 2017.

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

[10] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, "Brits: Bidirectional recurrent imputation for time series," *Advances in Neural Information Processing Systems*, vol. 31, pp. 6775–6785, 2018.

[11] Y. Luo, Y. Zhang, X. Cai, and X. Yuan, "E2gan: End-to-end generative adversarial network for multivariate time series imputation," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 3094–3100.

[12] J. Yoon, J. Jordon, and M. Van Der Schaar, "Gain: Missing data imputation using generative adversarial nets," *arXiv preprint arXiv:1806.02920*, 2018.

[13] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[14] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048–2057.

[15] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, "Image captioning with semantic attention," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4651–4659.

[16] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International conference on machine learning*. PMLR, 2019, pp. 7354–7363.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[18] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[19] 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 *NIPS*, 2014.

[20] L. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," *Advances in neural information processing systems*, vol. 28, pp. 262–270, 2015.

[21] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2414–2423.

[22] R. A. Yeh, C. Chen, T. Yian Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, "Semantic image inpainting with deep generative models," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5485–5493.

[23] 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.

[24] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[25] S. Zhang, B. Guo, A. Dong, J. He, Z. Xu, and S. X. Chen, "Cautionary tales on air-quality improvement in beijing," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 473, no. 2205, p. 20170457, 2017.

[26] J. Burgués, J. M. Jiménez-Soto, and S. Marco, "Estimation of the limit of detection in semiconductor gas sensors through linearized calibration models," *Analytica chimica acta*, vol. 1013, pp. 13–25, 2018.

[27] J. Burgués and S. Marco, "Multivariate estimation of the limit of detection by orthogonal partial least squares in temperature-modulated mox sensors," *Analytica chimica acta*, vol. 1019, pp. 49–64, 2018.

[28] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing value estimation methods for dna microarrays," *Bioinformatics*, vol. 17, no. 6, pp. 520–525, 2001.

[29] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.