2022 IEEE International Conference on
Systems, Man, and Cybernetics (SMC)

Multi-indicator Water Time Series Imputation with Autoregressive Generative Adversarial Networks

Jing Bi[1], Zichao Wang[1], Haitao Yuan[2], Kun Ni[1] and Junfei Qiao[1]

[1]Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
[2]School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China

2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC) | 978-1-6654-5258-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/SMC53654.2022.9945604

***Abstract*—The water quality data has missing values and lacks integrity because water environment monitoring equipments are easily damaged by environmental influences, thereby affecting the analysis accuracy of downstream tasks. Traditional data imputation methods include mean/last filling, K-nearest neighbor, matrix factorization, Lahrangian interpolation, *etc.*, do not consider time dependence or fail to use complex relations among multiple features. Inspired by successful applications of various variants of Generative Adversarial Networks (GANs) on time series data, this work proposes a time series data imputation method called GEDA, which integrates GAN, an Encoder-Decoder structure, and an Autoregressive network. GEDA adopts GAN to learn the probability distribution of multi-feature time series, and imputes the missing values with the generated data. Then, GEDA combines feature extraction of the encoder-decoder structure, and time dependence capturing of the autoregressive network. Real-life dataset-based experimental results demonstrate GEDA outperforms several state-of-the-art data imputation methods in terms of accuracy.**

***Index Terms*—Water quality, data imputation, generative adversarial networks, encoder-decoder, autoregressive networks**

I. Introduction

Water quality monitoring equipments are easily affected by the external environment, *e.g.*, river corrosion and bad weather, which makes the collected water environment data lack of integrity. This greatly affects the accuracy of water quality prediction. Therefore, it is highly essential to impute the missing values. In addition, there is strong time dependence in the time series of water quality data, and there are complex relations among multiple indicators. Thus, it is also challenging to realize the data imputation. To impute the missing values of the data, traditional methods fall into three categories including simple filling, interpolation, and machine learning methods. Simple filling adopts some special values, *e.g.*, zero, mean, mode, and the last valid observation. However, it does not consider the correlations among multiple features and time dependence. The interpolation method [1] adopts linear interpolation and multiple regression prediction models, and it includes Lagrangian interpolation, Newton interpolation, cubic Hermite interpolation and cubic spline interpolation. Among machine learning methods, K-Nearest Neighbor (KNN) [2], principal components analysis [3], Matrix Factorization (MF) [4] and other methods are widely used for missing value imputation. They consider relations among multiple features, but do not consider time dependence.

The water environment monitoring data is collected in real time and often contains much noise. Deep learning is a branch of machine learning, and it is widely applied to solve a variety of data mining problems. Compared with traditional machine learning methods, deep learning can effectively extract deep features of data, and directly investigate its hidden attributes. Goodfellow *et al.* [5] propose a Generative Adversarial Networks (GANs), which are implicit density generative models. It trains the generator and discriminator against each other and continuously adjusts the parameters to learn the data distribution. In recent years, GAN have been widely used on time series data. Mogren *et al.* [6] propose an Recurrent GAN model that adopts Recurrent Neural Network (RNN) to construct generator and discriminator to simulate the tonal sequence of classical music. Hyland *et al.* [7] propose a Deep Convolutional GAN that combines a convolutional neural network and GANs to generate the time series of medical data. However, these methods only rely on the feedback of the discriminator to guide the generator to generate time series data, which may not guide the generator to effectively mine the time dependence. In addition, the complex indicator relations also increase the learning difficulty of the model. Yoon *et al.* [8] combine unsupervised GAN with supervised learning, which guides the generator to generate time-dependent data. Compared with recurrent GAN, their model achieves significant improvement on the time series data.

Inspired by the successful applications of GAN on the time series data, some researchers adopt GAN for the data imputation. The main idea is to regard the data imputation task as data generation. These methods first generate the complete data close to the real one, then adopt the generated data to impute missing values. Yoon *et al.* [9] adopt GAN for the data imputation, but they do not focus on the time series data. Luo *et al.* [10] adopt GAN for the time series data imputation, and use a special Gate Recurrent Unit (GRU) to learn irregular time intervals between valid observations, which is more suitable for incomplete datasets to train. However, this model adopts the basic GAN structure, which is insufficient for learning temporal dependence and complex feature relations.

To solve above-mentioned problems, this work proposes a time series data imputation method called GEDA, which integrates GAN, an Encoder-Decoder structure, and an

This work was supported in part by the National Natural Science Foundation of China under Grants 62073005 and 62173013, and the Fundamental Research Funds for the Central Universities under Grant YWF-22-L-1203.

Autoregressive network. Specifically, GEDA adopts an imputed GRU (GRUI) as the RNN unit. GEDA mainly includes two stages, *i.e.*, GAN training and data imputation. The GAN training adopts the encoder-decoder structure to generate semantic representation to extract features. Then, it adopts the AutoRegressive (AR) network to obtain the probability distribution of the semantic representation, which facilitates encoder and generator to capture time dependence. The data imputation adjusts randomly sampled noise in the latent space, and makes the generated data as close to the real one as possible. Then, the generated data is used to impute missing values. Real-life dataset-based experimental results demonstrate GEDA outperforms several state-of-the-art data imputation methods in terms of accuracy.

II. Model framework

X denotes a multivariate d-dimensional incomplete time series, and $X = (x_1, x_2, ..., x_L)^T \in \mathbb{R}^{L\times d}$. x_t denotes the sample of X in time step t, and x_t^j denotes the jth feature of x_t. We first train a GAN network to generate data, and adopt the generated data to fill in the missing values. Sections II-A, II-B, and II-C introduce the framework of the GAN. Section II-D introduces the training process. Section II-E describes the use of GAN to fill in the missing values.

A. GAN framework

The model framework consists of five parts: Encoder, Decoder, Generator, Discriminator, and an Autoregressive network. The overall framework is shown in the Fig. 1. The encoder-decoder structure has many types of implementation ways and the only requirement is that it needs to be autoregressive, *i.e.*, the output in each time step is determined by the previous inputs. The seq2seq model based on the attention mechanism [11], temporal convolutional network [12], transformers [13] are typical examples of the encoder-decoder structure. This work adopts the seq2seq with attention structure as the encoder-decoder structure.

En represents the encoder, which extracts feature and learns the semantic representation. In the later adversarial training, En can help a generator G to generate semantically similar time series data. $x_t \in \mathbb{R}^d$ and $s_t \in \mathbb{R}^m$ denote the input and output of the encoder, and s_t is obtained as:

$$s_t = \mathrm{En}(x_{1:t-1}) \tag{1}$$

where m is the dimension of the semantic representation.

De represents the decoder, which reconstructs the real data according to the semantic representation. $\tilde{x}_t$ denotes the reconstructed data in time step t, which is obtained as:

$$\tilde{x}_t = \mathrm{De}(s_{1:t-1}) \tag{2}$$

L_{R} denotes a loss function of the encoder-decoder, which yields the reconstruction loss. L_{R} is obtained as:

$$L_{\mathrm{R}} = \mathbb{E}_{x_{1:L}\sim p}\left(\sum\nolimits_t ||x_t - \tilde{x}_t||_2\right) \tag{3}$$

where $p(x_{1:L})$ denotes the density of the real data.

AR represents the autoregressive network, whose output is the prediction of the next time step. Its loss function is the mean square error (MSE) between the predicted values and the ground truth ones. In our autoregressive network, the joint probability density function of the semantic representation is decomposed by $q(s) = \prod_{t=1}^{L} q(s_t|s_{1:t-1})$. The input of AR is divided into two types. One is the semantic representation of the real data outputted by the encoder, and the other is that generated by the generator. Given two different inputs, GAN is used to make the behaviors (outputs or hidden states) of the AR network which denote the distribution of data as closely as possible, thereby making the distribution of the real data and generated one as close as possible [14]. L_{AR} denotes a loss function of the AR network, which yields its MSE. L_{AR} is obtained as:.

$$L_{\mathrm{AR}} = \mathbb{E}_{x_{1:L}\sim p}\left[\sum\nolimits_{t=1} ||s_t - s'_t||_2\right] \tag{4}$$

where $s'_t = \mathrm{AR}(s_{1:t-1})$ denotes output of AR in time step t.

G represents the generator, which attempts to map a random vector z sampled from the latent space to a time series in the semantic one, and makes that the discriminator cannot distinguish the generated data from the real one. D represents the discriminator that outputs the probability of being real in each time step. It is worth noting that the input of the discriminator is classified into two types. One type includes the real data and the generated one, and another includes the behavior of the AR network representing the data distribution. Each type is further divided into real but incomplete data, and fake but complete one generated by the generator.

The former adopts the discriminator to make the generated data and the real one indistinguishable, thereby implicitly learning the distribution of the real data. The latter adopts the discriminator to explicitly make the distribution of the real data and the generated one closer. There are problems with traditional training of GAN, *i.e.*, the training process is unstable and requires constant balancing of training levels of the generator and the discriminator, and there is a collapse mode problem where generated samples lack diversity [15]. Wasserstein GAN (WGAN) is an improved GAN that adopts the Wasserstein distance to effectively solve above problems, and provide an effective metric (Wasserstein distance) to evaluate the entire training process. The loss function of WGAN is given as follows:

$$L_{\mathrm{G}} = \mathbb{E}_{z\sim\hat{p}}\left[-\mathrm{D}(\mathrm{G}(z))\right] \tag{5}$$

$$L_{\mathrm{D}} = \mathbb{E}_{z\sim\hat{p}}\left[\mathrm{D}(\mathrm{G}(z))\right] - \mathbb{E}_{x\sim p}\left[\mathrm{D}(\mathrm{x})\right] \tag{6}$$

B. Imputed GRU (GRUI)

The time interval between two valid observations is irregular. To capture the irregular time interval, GRUI [10] is used in this work. To obtain the time interval between two valid values, we introduce a time interval matrix $\delta \in \mathbb{R}^{L\times d}$, which is obtained as:

$$\delta_{t_i}^j = \begin{cases} t_i - t_{i-1}, & M_{t_{i-1}}^j = 1 \\ \delta_{t_{i-1}}^j + t_i - t_{i-1}, & M_{t_{i-1}}^j = 0 \ \&\ i > 0 \\ 0, & i = 0 \end{cases} \tag{7}$$

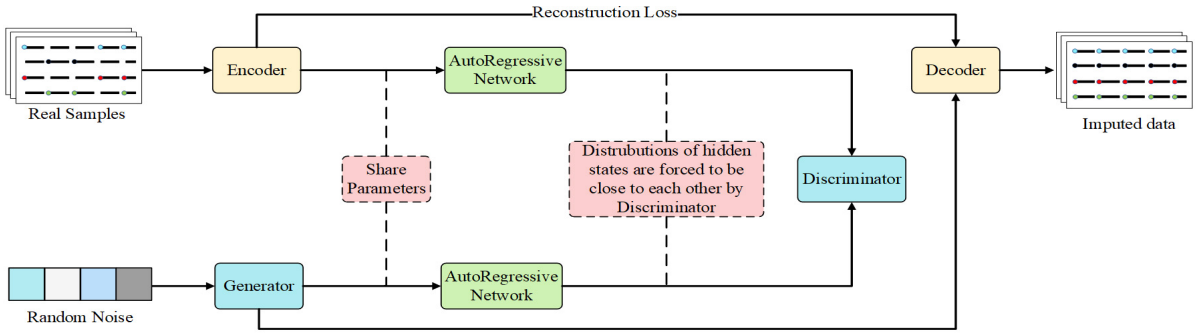


Fig. 1. The proposed GAN model including encoder-decoder, generator, discriminator and an autoregressive network

where t_i denotes the ith time step. A mask matrix $M \in \mathbb{R}^{L\times d}$ indicates whether the value in X exists or not, *i.e.*, $M^j_{t_i}=1$ if $x^j_{t_i}$ exists; otherwise, $M^j_{t_i}=0$.

Compared with GRU, a weight decay vector $\beta \in \mathbb{R}^{L\times m}$ is adopted to fit the decay of the influence of past observations. Longer time interval between two valid observations means the less influence of previous observations on current time step.

$$\beta_{t_i}=\frac{1}{e^{\max(0,W_\beta \delta_{t_i}+b_\beta)}} \tag{8}$$

where W_β and b_β denote the weight and bias that need to be learned.

We control β_{t_i} to be between $(0, 1]$ by a negative exponential function. The structure of each cell unit in GRUI is shown in the Fig. 3.

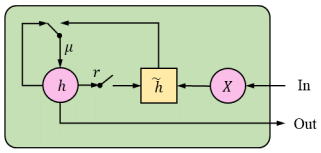


Fig. 2. GRU cell

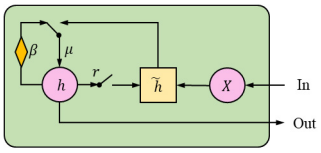


Fig. 3. GRUI cell

The update formulas of GRUI cell units from the input to the output are given as:

$$h'_{t_i-1}=\beta_{t_i}\odot h_{t_i-1} \tag{9}$$

$$\mu_{t_i}=\sigma(W_\mu[h'_{t_i-1}]+b_\mu) \tag{10}$$

$$r_{t_i}=\sigma(W_r[h'_{t_i-1}, x_{t_i}]+b_r), \tag{11}$$

$$\tilde{h}_{t_i}=\tanh(W_{\tilde{h}}[r_i\odot h'_{t_i-1}, x_{t_i}]+b_{\tilde{h}}) \tag{12}$$

$$h_{t_i}=(1-\mu_{t_i})\odot h'_{t_i-1}+\mu_{t_i}\odot\tilde{h}_{t_i} \tag{13}$$

where h_{t-1} denotes the hidden state in time step $t-1$, μ, r, and $\tilde{h}$ denote the update gate, the reset gate, and candidate state, respectively. W_μ, W_r and $W_{\tilde{h}}$ denote the weight matrices. b_μ, b_r and $b_{\tilde{h}}$ denote bias vectors. $\sigma(\cdot)$ and $\tanh(\cdot)$ denote the sigmoid activation and hyperbolic tangent functions. $\odot$ denotes the element-wise multiplication.

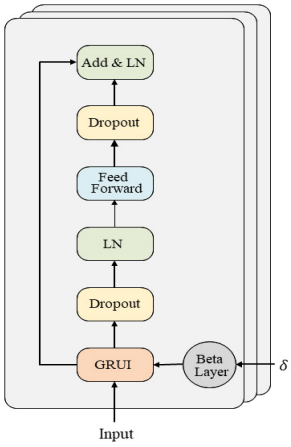


Fig. 4. The framework of sub-component

C. Component framework

The framework of each component is consistent, as shown in Fig. 4. Each layer consists of three sub-layers, the first is a beta layer that maps δ to β, the second is GRUI, and the third is a feed forward layer. The output of each layer is the input to the next layer after dropout [16] and the layer normalization (LN), and a residual connection (Add) [17] is adopted between the two layers. The output of each layer is LayerNorm$(x_t+\text{layer}(x_t))$, where layer(x_t) represents the output of an entire layer.

D. Training process

Let θ_{En}, θ_{De}, θ_{AR}, θ_{G}, and θ_{D} denote the parameters of the encoder, decoder, AR, generator and discriminator, respectively. The first step is to pre-train the encoder-decoder structure to extract features and reduce the dimension of the generated space. The second step is to pre-train AR to make it capture temporal dependence and fit probability distributions of semantic representations. The third step is to jointly train the encoder and decoder. L_{AR} is added to the optimization objective to facilitate the encoder to capture the time dependence. The optimization objective is given as:

$$\min_{\theta_{\text{En}},\theta_{\text{De}}} (\lambda L_{\text{AR}}+L_{\text{R}}) \tag{14}$$

where $\lambda \geq 0$ is a hyperparameter that balances the two losses.

The fourth step is to train the generator and the discriminator in an adversarial manner. When training the generator, L_{AR} is included to make the generator to generate time-dependent semantic representation data. The optimization objective is given as:

$$\underset{\theta_{\mathrm{G}}}{\mathrm{Min}}(\eta L_{\mathrm{AR}} + L_G) \tag{15}$$

where $\eta \geq 0$ is a hyperparameter that balances the two losses.

E. Imputation

The noise vector z is randomly sampled from the latent space, and the generated data may vary greatly as z changes. Therefore, although the generated data obeys the distribution of the original real data, their distance may also be large. Thus, it is important to find the optimal z from the latent space to make the generated data as close as possible to the real data. $L_{\mathrm{imputation}}$ denotes a loss function to measure the rationality of data imputation.

$$L_{\mathrm{imputation}}(z) = L_{\mathrm{r}}(z) + \gamma L_{\mathrm{d}}(z) \tag{16}$$

where γ denotes a hyperparameter that balances L_{r} and L_{d}, L_{r} denotes the distance between the real data and the generated data, and L_d denotes the output of the discriminator to make the generated data as real as possible. L_r is obtained as:

$$L_{\mathrm{r}} = ||X \odot M - \mathrm{G}(z) \odot M||_2 \tag{17}$$

It is worth noting that we only calculate valid values and ignoring missing ones. L_d is obtained as:

$$L_{\mathrm{d}} = -\left[\mathrm{D}(\mathrm{G}(z)) + \mathrm{D}(\mathrm{AR}(\mathrm{G}(z)))\right] \tag{18}$$

We also consider both the generated data and the behaviors of AR. For sequence X, we randomly sample z from a standard Gaussian distribution, and feed it into the trained generator to obtain the semantic representation. Then, we feed it into the decoder to obtain the time series data. Finally, we take z as a parameter and adjust it by the back-propagation through the loss function $L_{\mathrm{imputation}}(z)$. When the loss function is optimal, we adopt the generated data to impute missing values in X. X_{imputed} denotes the imputed data given as:

$$X_{\mathrm{imputed}} = X \odot M + (1 - M) \odot \mathrm{De}(\mathrm{G}(z)) \tag{19}$$

III. Experimental Results

GEDA is evaluated with real-life water environment data in the Beijing-Tianjin-Hebei region. To demonstrate its performance, we compare it with several benchmark methods including last value filling, mean filling, MF, and KNN.

A. Dataset

The water environment data is collected from the National Surface Water Quality Automatic Monitoring Real-time Data Release System. The time span is from Sept. 2018 to Nov. 2021, the sampling interval is 4 hours, and there are about 5500 samples. Each sample contains four indicators, *i.e.*, water temperature, pH value, total nitrogen, and dissolved oxygen. We obtain relatively complete monitoring data for a period of time, and then manually remove some data, resulting in a data missing rate of 30%.

B. Benchmark methods

We choose the following four widely used benchmark methods for comparison.

1) Last value filling (LVF). It fills each missing value with its last valid value.
2) Mean filling (Mean). It fills each missing value with the mean value of all samples in the dataset.
3) MF. It factorizes an incomplete matrix into two low-order matrices U and V, and then adjusts them through gradient descent. It adopts U and V to generate a complete matrix to fill in the missing values of an incomplete matrix.
4) KNN. It adopts the mean of the K nearest neighbors to fill in the missing values.

C. Evaluation metrics

The data imputation aims to provide a complete data set, and two important characteristics of the data need to be guaranteed. They include fidelity that means the real data and the generated one are inseparable, and availability that means when the data is used for the same purpose, the generated samples need to have the same effect as real ones. Therefore, we adopt three evaluation metrics.

1) Accuracy (AUC) [18]. To quantitatively evaluate the fidelity of the imputed data, a classifier of a two-layer GRU, is trained to distinguish the imputed data from the real one. In the best imputation result, the classification probability of imputed and real data is 0.5. Therefore, AUC, *i.e.*, $|0.5 - \xi|$ is adopted as a metric where ξ is the classification accuracy.
2) Mean Absolute Error (MAE) [19]. It is used to evaluate the fidelity of the imputed data.
3) MSE. To evaluate quantitatively the availability of the imputed data, we train a prediction network (2-layer GRU) to predict the next value. MSE between the real value and the predicted one is adopted as an evaluation metric.

D. Parameter Tuning

The selection of hyperparameters in GEDA is critical to its performance. They include the length of the time series (L), the number of neurons in GRU (m), and the weight (γ) of the loss function during data imputation. L is a crucial parameter for GEDA. If it is too long, the time series contains too much past information. The water quality environment changes greatly over time, and if there is too much noise in a long sequence, the prediction accuracy might be reduced. If it is too short, the necessary information may be missing and the prediction accuracy is affected. Table I shows the AUC, MAE and MSE of GEDA with different L. It is observed that when L=12, three metrics are all the best.

The appropriate number of neurons in GRU also has an important impact on the accuracy of GEDA. Table II shows the AUC, MSE and MAE of GEDA with different m. It is observed that when m=36, three metrics are all the best. In the

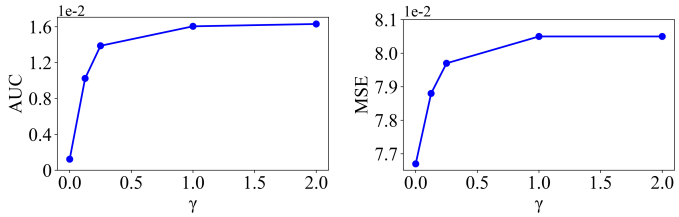
TABLE I
AUC, MAE AND MSE OF GEDA WITH DIFFERENT L

L	AUC	MSE	MAE
8	0.0179	0.0769	0.0264
12	**0.0086**	**0.0755**	**0.0287**
24	0.0229	0.0769	0.0363
36	0.0247	0.0782	0.0382

TABLE II
AUC, MSE AND MAE OF GEDA WITH DIFFERENT m

m	AUC	MSE	MAE
12	0.0237	0.0800	0.0339
24	0.0224	0.0773	0.0314
36	**0.0086**	**0.0755**	**0.0287**
48	0.0089	0.0758	0.0319

process of data imputation, it is necessary to adjust the random vector z to generate data that is sufficiently realistic and close to the original data. γ controls the balance between fidelity and distance from the real data. Larger γ means that the generated data is more real. Smaller γ means that the generated data is more close to real data. Fig. 5 shows the influence of weight γ on AUC and MSE. It is observed that with the increase of γ, AUC and MSE both increase. This shows that although the generated data is more realistic, the distance between the generated data and the original one becomes larger, which leads to a larger difference between the imputed data and the original one.



(a) AUC (b) MSE

Fig. 5. The influence of weight γ on AUC and MSE

Table III shows AUC, MSE and MAE of GEDA with GRU and GRUI. It is shown that when GRUI is used as a basic unit, GEDA effectively improves the fidelity of the generated data and the feature capturing ability of time dependence.

TABLE III
AUC, MSE AND MAE OF GEDA WITH GRU AND GRUI

	AUC	MSE	MAE
GRU	0.0123	0.0767	0.0304
GRUI	**0.0086**	**0.0755**	**0.0287**

Table IV shows AUC, MSE and MAE with different GAN frameworks including GAN, GAN with Encoder and Decoder (ED), GAN with AR, and GEDA. It is observed that compared with traditional GAN, the integration of the AR network effectively improves the result of data imputation. In addition, the integration of the encoder-decoder structure improves the fidelity of the generated data. Finally, our GEDA yields the best AUC, MSE and MAE.

TABLE IV
AUC, MSE AND MAE WITH DIFFERENT GAN FRAMEWORKS

	AUC	MSE	MAE
GAN	0.0459	0.0785	0.0370
GAN with ED	0.0305	0.0773	0.0330
GAN with AR	0.0435	0.0769	0.0339
GEDA	**0.0123**	**0.0767**	**0.0304**

The final parameter settings for our GEDA are summarized in Table V.

TABLE V
PARAMETER SETTINGS OF THE GEDA FOR TIME SERIES IMPUTATION

Parameter	**Value**	**Description**
L	12	Sequence length
m	36	GRUI hidden units
Optimizer	RMSprop	Selected optimizer
λ	0.01	Weight balance L_{AR} and L_{R}
γ	0	Weight balance L_{r} and L_{d}
η	0.01	Weight balance L_{AR} and $L(\mathrm{G},\mathrm{D})$

E. Comparison Results

Fig. 6 shows the comparison results of GEDA and other benchmark methods in terms of AUC, MSE and MAE.

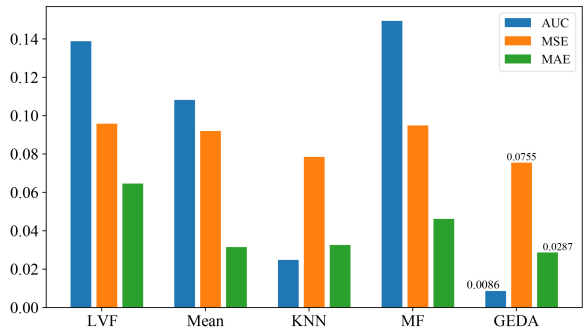


Fig. 6. AUC, MSE and MAE of different methods

It is observed that GEDA achieves the best results in three metrics of AUC, MSE and MAE, which indicates that the data imputed with GEDA has the highest degree of fidelity and the strongest availability. Due to the large fluctuation of data and irregular time intervals between two valid observations, the effect of LVF is poor. The time series has periodicity and seasonality, and therefore, the mean filling is not suitable for the imputation of time series data. Both KNN and MF consider the correlation between multiple features. However, due to the high temporal correlation of the time series data, KNN selects the mean value of the samples near current time step as the imputed data, which can be regarded as having a certain time dependence. Thus, its results are better than MF. GEDA considers both the time dependence and the correlation

between multiple features, and therefore, the data imputation results are the best among five methods.

After the original time series is imputed, the complete time series is yielded. Then, we adopt it to further compare the prediction accuracy of GEDA and other benchmark methods. Fig. 7 shows the comparison of one-step prediction result and the ground truth with GEDA and other benchmark methods. It is observed that the fitting results of four benchmark methods on the ground truth are worse than that of GEDA, indicating that they have worse ability to capture time dependence, and only achieve imputation results.

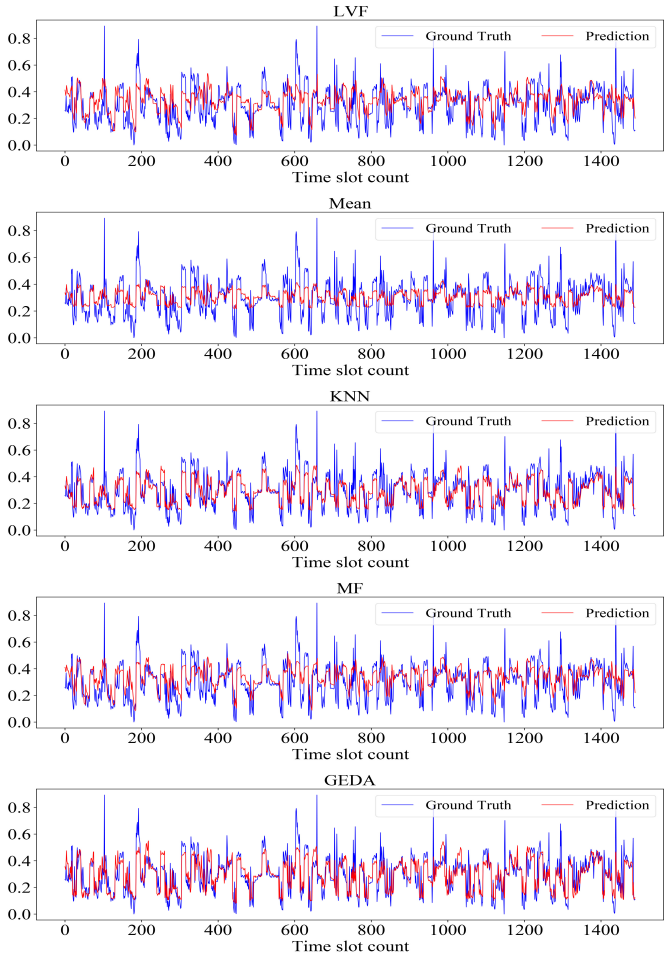


Fig. 7. One-step prediction with GEDA and other benchmark methods

IV. Conclusions and Future Work

This work proposes an imputation method of time series data, which is named GEDA, which combines Generative Adversarial Networks (GANs), an Encoder-Decoder structure, and an Autoregressive network. GEDA uses the encoder-decoder structure for feature extraction, and adopts the AutoRegressive (AR) network to capture time dependence. The AR network is introduced to represent the probability distribution of the data. Its behavior with the real data is as close as possible to that with the generated data by the adversarial training of GAN. In this way, multi-feature correlation, time dependence, and data distribution can be learned in an adversarial manner. Furthermore, this work imputes the missing values in the time series data of water quality by the data generated by the trained GEDA model. Real-life dataset-based experiments show that compared with four benchmark methods, GEDA achieves the best results in terms of accuracy.

In the future, we plan to improve GEDA in the following directions. First, inspired by transformer models in translation tasks, we plan to adopt the transformer as the encoder-decoder structure. Second, other variants of GAN for imputation will be also applied to further improve our current imputation model. Third, we plan to adopt more downstream tasks as evaluation metrics.

References

[1] Q. Dong, *et al.*, "An Integrated Deep Neural Network Approach for Large-Scale Water Quality Time Series Prediction," in *Proc. IEEE Int. Conf. on Systems, Man and Cybernetics*, 2019, pp. 3537–3542.

[2] S. Zhang, "Nearest Neighbor Selection for Iteratively kNN Imputation," *Journal of Systems and Software*, vol. 85, no. 11, pp. 2541–2552, 2012.

[3] L. Qu, L. Li, Y. Zhang, *et al.*, "PPCA-based Missing Data Imputation for Traffic Flow Volume: A Systematical Approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 512–522, 2009.

[4] R. Mazumder, T. Hastie, and R. Tibshirani, "Spectral Regularization Algorithms for Learning Large Incomplete Matrices," *Journal of Machine Learning Research*, vol. 11, pp. 2287-–2322, 2010.

[5] I. Goodfellow, *et al.*, "Generative Adversarial Nets," *In Proc. Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

[6] S. Hyland, C. Esteban, and G. Rätsch, "Real-valued (medical) Time Series Generation With Recurrent Conditional GANs," *arXiv preprint arXiv:1706.02633*, 2017.

[7] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks," *arXiv preprint arXiv:1511.06434*, 2015.

[8] J. Yoon, D. Jarrett, and M. Van der Schaar, "Time-series Generative Adversarial Networks," *In Proc. 2019 Advances in Neural Information Processing Systems*, pp. 5508–5518, 2019.

[9] J. Yoon, J. Jordon, M. Schaar, "Gain: Missing Data Imputation Using Generative Adversarial Nets," *In Proc. 2018 International Conference on Machine Learning*, pp. 5689–5698, 2018.

[10] Y. Luo, X. Cai, and J. Xu, "Multivariate Time Series Imputation with Generative Adversarial Networks," *In Proc. 2018 Advances in Neural Information Processing Systems*, pp. 1596–1607, 2018.

[11] I. Sutskeve, O. Sutskever, and Q.V. Le, "Sequence to Sequence Learning with Neural Networks," *In Proc. 2014 Advances in Neural Information Processing Systems*, pp. 3104–3112, 2014.

[12] F. Luo, *et al.*, "Temporal Convolutional Networks for Multiperson Activity Recognition Using a 2-D LIDAR," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7432-7442, Aug. 2020.

[13] A. Vaswani, *et al.*, "Attention is All You Need," *In Proc. 2017 Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[14] A. M. Lamb, *et al.*, "Professor Forcing: A New Algorithm for Training Recurrent Networks," *In Proc. 2016 Advances in Neural Information Processing Systems*, pp. 4601–4609, 2016.

[15] M. Arjovsky, *et al.*, "Wasserstein Generative Adversarial Networks," *In Proc. 2017 Int. Conf. on Machine Learning*. pp. 214–223, 2017.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, *et al.* "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929-1958, 2014.

[17] K. He, X. Zhang, S. Ren, *et al.* "Deep Residual Learning for Image Recognition," *In Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition*. pp. 770–778, 2016.

[18] J. Bi, *et al.*, "An Improved Attention-based LSTM for Multi-Step Dissolved Oxygen Prediction in Water Environment," in *Proc. 2020 IEEE Int. Conf. on Networking, Sensing and Control*, 2020, pp. 1–6.

[19] J. Bi, Y. Lin, Q. Dong, H. Yuan, and M. Zhou, "Large-scale Water Quality Prediction with Integrated Deep Neural Network," *Information Sciences*, vol. 571, pp. 191–205, 2021.