2024 5th International Conference on Computer Vision, Image and Deep Learning (CVIDL) | 979-8-3503-7382-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/CVIDL62147.2024.10604056

TsCDD-GAN: A Conditional Dual-Discriminator Generative Adversarial Network for Incomplete Time Series Data Imputation and Clustering

Bo Peng
School of Computer Science and Engineering
Tianjin University of Technology
Tianjin, China
pb@stud.tjut.edu.cn

Chen Dong*
School of Computer Science and Engineering
Tianjin University of Technology
Tianjin, China
* Corresponding author: dongc@tjut.edu.cn

***Abstract*—Time series clustering is a complex unsupervised data mining and analysis technique that can be applied to various fields such as signal processing, financial analysis, and more. However, time series data often contain missing values due to reasons such as sensor failures, data corruption, or intentional omissions in the real world. Traditional generative imputation methods operate independently from subsequent clustering tasks, and their performance relies on the combination with the clustering method, lacking a deeper connection. Efficiently integrating imputation and clustering tasks into a unified framework remains a formidable challenge. This paper presents a novel model that harmonizes the imputation and clustering processes, referred to as the Conditional Dual-Discriminator Generative Adversarial Network for Incomplete Time Series Data Imputation and Clustering (TsCDD-GAN). The TsCDD-GAN model leverages advanced bi-directional Recurrent Units (Bi-directional RNN) as the generator to learn and impute missing data, integrating two discriminators. The first discriminator assists the generator in producing more authentic values, while the second, a conditional discriminator, extracts clustering information from the latent representations of the data. This enables the generator to not only generate more realistic values but also ensure that the imputed data exhibit cluster-friendly characteristics. Through experiments conducted on the UCR time series dataset, our model has demonstrated better performance in terms of imputation accuracy on incomplete datasets.**

Keywords-Generative Adversarial Network; Bi-directional RNN; Unsupervised Learning; Deep Learning; Imputation; Incomplete Time Series;

I. INTRODUCTION

Time series analysis has become an increasingly popular research direction, giving rise to various studies based on deep learning and other methodologies [1]. Time series clustering constitutes a sophisticated unsupervised technique within the realms of data mining and analysis, finds extensive application across various domains including signal processing, financial analysis, and more. By clustering time series data, it becomes possible to discern patterns and trends within the data, which in turn facilitates tasks such as anomaly detection, pattern recognition, and segmentation. This methodology plays a pivotal role in deciphering complex data structures and uncovering intrinsic data regularities.

However, in the real world, time series data often contain missing values due to natural or human factors such as sensor malfunctions and data corruption. This is particularly problematic in clustering analyses, where missing data can undermine data integrity and clustering accuracy, degrading model performance and reliability, and hindering further analysis. In generative imputation, single-stage methods don't consider clustering, relying on pairing with clustering for performance, missing a deeper integration. Cao et al. (2018) proposed BRITS, a bidirectional recurrent imputation for time series. BRITS imputed missing values via bi-directional RNN [2]. This disconnected approach often fails to adequately consider the temporal dynamics and clustering requirements among the data, potentially leading to information loss during the clustering process and thus affecting clustering efficacy. Luo et al. (2018) introduced a method for multivariate time series imputation using generative adversarial networks (GAN), which has demonstrated effectiveness in handling missing values in time series data [3]. In generative imputation models designed to handle clustering tasks, the clustering module operates independently of the GAN network, leading to a redundant network structure that does not deeply integrate the clustering optimization structure with the GAN network.

To address this challenge, this paper introduces a novel model that unifies the imputation and clustering processes for incomplete time series data. This model, referred to as the conditional dual-discriminator generative adversarial network for incomplete time series data imputation and clustering (TsCDD-GAN), leverages the power of generative adversarial networks (GANs) to jointly optimize the imputation and clustering tasks. The TsCDD-GAN model integrates a generator and two discriminators. The generator employs a bidirectional RNN network to fill in missing values by comprehensively analyzing the short- and long-term characteristics of the time series. One discriminator is responsible for distinguishing whether the data generated by the generator is real or fake, while the other discriminator, through learning the latent representations of the data, uses k-means clustering within the discriminator to generate cluster representations. This model aims to reduce the loss from generation to clustering. In contrast to traditional two-stage tasks, this model achieves an integrated optimization from imputation to clustering.

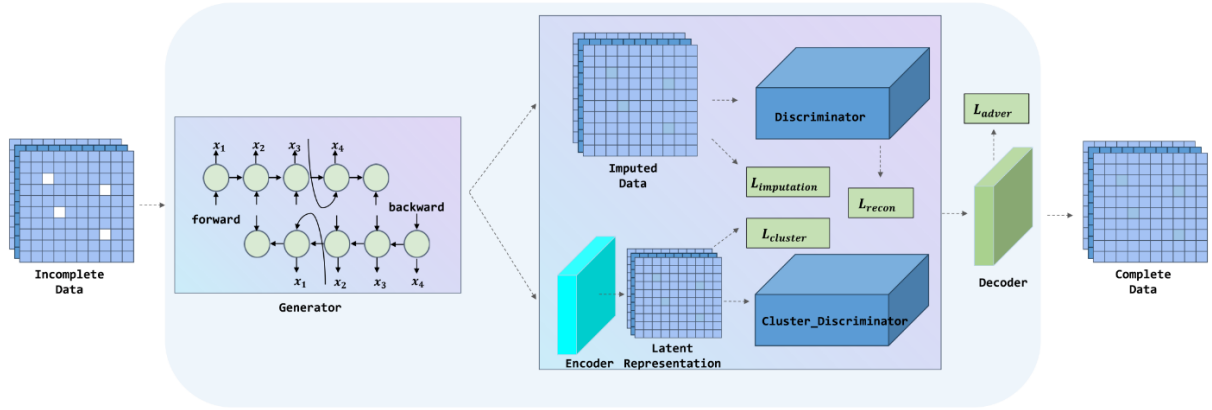

Figure 1. Conditional Dual-Discriminator Generative Adversarial Network

II. Related Work

A. Generative Imputation Model

For simple mathematical methods such as mean filling and median filling, as well as machine learning methods like KNN for imputation, these approaches do not consider the correlations between time series data [4]. Generative filling using GAN networks can fully utilize the relationships within time series data, resulting in the generation of more realistic data. Based on the GAN network, the choice of the generator's architecture also affects the quality of data generation. Building on RNN networks, generators based on bi-directional RNNs and bi-directional LSTMs have been developed, showcasing excellent versatility [5] [6]. However, in time series clustering tasks, the downstream clustering task is independent of the imputation task, necessitating the exploration of various model combinations. In real-world applications, the necessity to experiment with various model combinations leads to inefficiencies in the deployment process.

B. Latent Representation Learning

In machine learning, representing high-dimensional data with a lower-dimensional, compressed format is referred to as Latent Representation. In the context of unsupervised learning for time series data, an Encoder is used to compress the input data into a latent representation, which is then reconstructed back to the original data by a Decoder [7].

During this process, the autoencoder is capable of learning an effective representation of the data that captures key features and structures while ignoring insignificant details, enhancing the model's understanding of the data. This representation is very useful for subsequent tasks [8] [9].

C. Joint Optimization Model

In time series clustering tasks, to reduce the error from imputation to clustering, a joint optimization of time series data imputation and clustering is conducted through two parallel network branches. By performing clustering learning on the latent representation, the clustering features of the time series data are extracted and the results are fed back into the GAN network. This guides the GAN network to generate cluster-friendly data [10]

III. Method

Here, we refer to our model as the Time Series Conditional Dual-Discriminator Generative Adversarial Network (TsCDD-GAN), and its structure is depicted in Fig. 1. The front branch consists of a generator that estimates and fills in incomplete time series data using a bi-directional RNN network, outputting the hidden states and the filled complete time series data. The rear branch comprises two discriminators: the Discriminator and the Cluster_Discriminator. The Discriminator receives the data filled by the generator and outputs the probability of each data point being an observation. The Cluster_Discriminator integrates an encoder-decoder network, receiving the hidden states output by the generator and inputting them into the encoder layer to obtain latent representations. In the decoder layer, the latent representations are reconstructed using the K-means objective to produce cluster-friendly data.

The Time Series Conditional Dual-Discriminator Generative Adversarial Network (TsCDD-GAN) overcomes the limitations of traditional GANs by integrating both data imputation and clustering within a single framework. By using a bi-directional RNN network in its front branch, the generator effectively estimates and fills in incomplete time series data, producing hidden states and complete time series data. This addresses the issue of compromised data integrity and clustering accuracy due to missing values. The rear branch of TsCDD-GAN, consisting of the Discriminator and the Cluster_Discriminator, enhances the model's performance and reliability. The Discriminator evaluates the authenticity of the data filled by the generator, while the Cluster_Discriminator, incorporating an encoder-decoder network, ensures that the filled data is cluster-friendly by reconstructing the hidden states using the K-means objective.

By operating both discriminators in tandem and optimizing them simultaneously with the filling and clustering processes, TsCDD-GAN achieves a higher degree of accuracy and cluster compatibility in the generated data. This integrated approach not only elevates the quality of the imputed time series but also simplifies the clustering phase, resulting in more dependable clustering results. TsCDD-GAN's novel integration of imputation and clustering optimizes the performance and robustness of analytical models, paving the way for more precise subsequent data analyses. This contribution stands to revolutionize the handling of incomplete time series data,

offering a comprehensive solution that addresses the limitations of traditional GANs and enhances the overall reliability of time series analysis in various domains.

A. Problem Statement

Considering a multivariate time series dataset, Data, which comprises N time series, denoted as $Data = \{X_1,\ X_2, \dots, X_N\}$. Each time series X_i is a matrix defined over the temporal dimension D and the feature dimension r, represented as $X_i = \{x_1,\ x_2, \dots, x_d\}$, where each time point x_d is a r-dimensional vector. In the presence of missing values within our dataset, we have an associated matrix set $Q = \{Q_1, Q_2, \dots, Q_N\}$ that corresponds to each time series X_i. Each matrix Q_i is composed of D elements, denoted as $Q_i = (q_1, q_2, \dots, q_d)$, where each element q_d is a r-dimensional binary vector with components either 0 or 1. For missing feature components in x_t, we mark q_d with 0; For present components, we mark it with 1.

B. Architecture of TsCDD-GAN

The generator's architecture employs a bidirectional RNN, aimed at capturing the overall structure and characteristics of the time series [11]. The bidirectional RNN network can be represented as follows:

$$h_t = \tan \mathrm{h}(W_h h_{t-1} + W_x x_t + b) \tag{1}$$

h_t: Hidden state at time t, capturing information up to that point and used to generate output at the next step. h_{t-1}: Hidden state from the previous step, used to update the current state. W_h: Weight matrix from the previous hidden state to the current hidden state, determining the update. W_x: Weight matrix from the input to the hidden state, influencing the hidden state update. x_t: Input at time t, typically the observed value in time series analysis. b: Bias vector for the hidden state, affecting the update. tanh: Represents the hyperbolic tangent function. This function transforms the weighted sum into a value between [-1, 1], which helps to limit the value of the hidden state and avoids large gradient issues.

The entire formula represents how the hidden state at the current time step is updated based on the hidden state from the previous time step and the input at the current time step. This update process is recursive, with each time step using the information from the previous time step to generate the output at the current time step. By this means, RNNs can learn the long-term dependencies within time series data.

The formula L_{recon} calculates the sum of the squares of the differences between the original sequence X_i and the reconstructed sequence $f_{\mathrm{dec}}(H_i)$, then divides this sum by the length of the time series N, and finally takes the square root. In encoder-decoder, the potential representation of the time series data is extracted and at the end the features of the time series data are reconstructed, where the reconstruction loss is used to encourage into a sequence as close as possible to the original data. By minimizing this loss, the model learns how to reconstruct the missing data, thus improving the data quality.

$$Hi = f_{\text{generator / encoder}}(Xi, Mi) \tag{2}$$

$$L_{recon} = \frac{1}{N}\sum_{i=1}^{N} \left|\left|\left(X_i - f_{\mathrm{dec}}(H_i)\right) \circ O_i\right|\right|^2 \tag{3}$$

The formula L_{adver} calculates the expected value of the difference between the discriminator's output on the generated data and 1 for each missing data point i. The discriminator should output a value close to 0 to indicate that the data is generated. If the data is missing ($M_i = 0$), then the generated data (U_i) is generated by the generator, and the discriminator should output a value close to 0, indicating that the data is generated. The (1 - M_i) in Eq. ensures that the expected value of the difference between the discriminator's output on the generated data and 1 is computed only if the data is missing.

$$L_{adver} = \frac{1}{N}\sum_{i=1}^{N} (1 - O_i) \log\left(1 - D(U_i)\right) \tag{4}$$

By minimizing this adversarial loss, the generator can learn to generate more realistic data, while the discriminator can learn to better distinguish between real and generated data. This helps to improve the performance of the overall model, especially when dealing with missing data.

C. Loss Function

Finally, we can obtain the overall loss function of TsCDD-GAN as follows:

$$L_{TsCDD} = L_{imputation} + L_{recon} + L_{adver} + L_{cluster} \tag{5}$$

In which $L_{imputation}$ is designed to encapsulate the essence of incomplete time series, ensuring that the inferred missing values are more congruent with the attributes of the original data. L_{recon} constitutes the reconstruction loss within the encoder-decoder architecture, reconstructing significant features from the original samples. L_{adver} is an adversarial loss function that mitigates the propagation of errors from imputation to clustering tasks, aiding in joint optimization. $L_{cluster}$ represents the clustering loss within the cluster discriminator, which incentivizes the learning of clustering features within the latent representations to establish cluster structures, thereby enhancing the clustering friendliness [12].

IV. Experiments

A. Benchmarks

In our experiments, we selected 20 UCR time series datasets from the UEA Time Series Classification Repository [13]. Table I presents the details of the UEA Time Series dataset.

Table I Information about Some Ucr Datasets

Name	Train	Test	Class	Length
Arrow	36	175	3	251
BeetleFly	20	20	2	512
Beef	30	30	5	470
BME	30	150	3	128
Bird	20	20	2	512

Name	Train	Test	Class	Length
Car	60	60	4	577
CBF	30	900	3	128
Chinetown	20	343	2	24
ECG200	100	100	2	96
Facefour	24	88	4	350

B. Experiment Setup

We first trained the model on the training dataset and then evaluated its performance on the test dataset. As previously discussed, the generator component of the model is configured as a bidirectional multi-layer RNN. The discriminator part of the model consists of five layers of RNN, with each layer having a predefined size set at {32, 16, 8, 16, 32}. Gated Recurrent Units (GRUs) are employed within the RNNs. The encoder has a variable number of layers, l, which can be either 1 or 2, and each layer within the encoder has a variable size, h. The cluster discriminator processes the final concatenated hidden state from the encoder and passes it through a fully connected layer to derive the latent representations. The decoder, which is a single-layer RNN, takes the learned representation as its starting state and employs the forward identifier forward as a starting signal for iterative prediction.

The experiments were conducted on the Pytorch platform, utilizing an Intel Core i7-11800H CPU operating at 4.60 GHz, 32 GB of RAM, and a GeForce GTX 3060 GPU. Optimization of the model was achieved through the use of the Adam optimizer (Kingma and Ba 2015), with an initial learning rate set at 5e-3.

C. Evaluation Metrics

We utilize the RI coefficient to assess the accuracy of the imputed dataset [14].

$$\mathrm{RI} = \frac{TP + TN}{n(n-1)/2} \tag{6}$$

The RI coefficient is capable of handling ordered data and identifying potential clustering structures within the data. This feature is particularly useful when processing time series data, as it helps in evaluating the accuracy of the imputed dataset by reflecting the similarity between the imputed values and the original values.

Clustering purity over calculates the frequency of occurrence of true categories in each cluster and selects the largest frequency as the purity of that cluster. Then, the purity of all the clusters is weighted and averaged to get the purity of the entire clustering result. The formula for calculating the clustering purity is as follows:

$$\mathrm{CP} = \frac{1}{k}\sum_{i=1}^{k} \max_{j} \mathrm{TP_{ij}} \tag{7}$$

The Adjusted Rand Index takes into account the effect of randomly assigned clustering results on the original clustering results, and can tell us the degree of difference between the clustering results and the randomly assigned results. The ARI index adjusts the RI index to more accurately reflect the quality of the clusters when the sample size is large. The formula for ARI is as follows:

$$\mathrm{ARI} = \frac{RI - E(\mathrm{RI})}{\max(\mathrm{RI}) - E(\mathrm{RI})} \tag{8}$$

The calculation of NMI takes into account the theoretical upper bound of the information and therefore provides a relatively accurate evaluation of the clustering quality. The formula for NMI is as follows:

$$\mathrm{NMI} = \frac{2\mathrm{MI}(\mathrm{U},\mathrm{V})}{\mathrm{H}(\mathrm{U}) + \mathrm{H}(\mathrm{V})} \tag{9}$$

Table II displays the Clustering Evaluation Metrics on 20 Incomplete Datasets of the UCR Dataset.

Table II Clustering Evaluation Metrics on 20 Incomplete Datasets of UCR Dataset

Metric	RI	CP	ARI	NMI
Arrow	0.550	0.500	0.108	0.110
Beef	0.584	0.360	-0.01	0.200
BeetleFly	0.605	0.750	0.220	0.344
BME	0.577	0.553	0.194	0.240
Car	0.609	0.383	0.001	0.009
CBF	0.566	0.469	0.070	0.091
Chinatown	0.571	0.710	0.126	0.062
Doger	0.626	0.740	0.140	0.053
ECG200	0.553	0.639	0.024	0.085
ECGFive	0.557	0.688	0.140	0.122
FaceFour	0.629	0.410	0.027	0.103
Fungi	0.844	0.280	0.040	0.302
Freezer	0.527	0.616	0.054	0.089
Herring	0.504	0.592	-0.016	0.001
House	0.501	0.622	0.043	0.032
Meat	0.549	0.220	-0.010	0.016
OliveOil	0.508	0.400	0.065	0.212
Rock	0.619	0.320	0.056	0.194
Symbols	0.785	0.500	0.282	0.478
UMD	0.549	0.484	0.010	0.026

D. Comparison with Baseline Methods

We compare Ts-CDD with a one-stage method (state-of-the-art incomplete time-series deep clustering, VaDER (de Jong et al. 2019)). We use RI and CP as evaluation metrics to assess the performance of our Ts-CDD against VaDER on 20 incomplete UCR time series datasets [15].

Table III Rand Index & Cluster Purity Comparison on 20 Incomplete Datasets of UCR Dataset

Metric	Rand Index		Cluster Purity	
Method	VaDER	Ts-CDD (ours)	VaDER	Ts-CDD (ours)
Arrow	0.551	0.550	0.508	0.500
Beef	0.172	0.584	0.2	0.360
BeetleFly	0.473	0.605	0.5	0.750
BME	0.328	0.577	0.333	0.553
Car	0.243	0.609	0.316	0.383
CBF	0.332	0.566	0.335	0.469

Metric	Rand Index		Cluster Purity	
Method	VaDER	Ts-CDD (ours)	VaDER	Ts-CDD (ours)
Chinatown	0.600	0.571	0.725	0.710
Doger	0.611	0.626	0.739	0.740
ECG200	0.534	0.553	0.640	0.639
ECGFive	0.499	0.557	0.502	0.688
FaceFour	0.253	0.629	0.295	0.410
Fungi	0.590	0.844	0.102	0.280
Freezer	0.499	0.527	0.500	0.616
Herring	0.509	0.504	0.593	0.592
House	0.508	0.501	0.579	0.622
Meat	0.322	0.549	0.333	0.220
OliveOil	0.271	0.508	0.400	0.400
Rock	0.287	0.619	0.420	0.320
Symbols	0.166	0.785	0.178	0.500
UMD	0.328	0.549	0.333	0.484
Average	0.2685	0.3263	0.3063	0.2774
Win	5	15	7	13

Table III showcases the comparison of rand index and cluster purity metrics between our model and the baseline model on 20 incomplete datasets.

The experimental outcomes demonstrate that Ts-CDD exhibits superior performance over VaDER in terms of both the Rand index (RI) and clustering purity (CP). Ts-CDD attains a higher degree of stability across diverse dataset structures, highlighting its efficacy. The considerable improvement in the mean values of both RI and CP coefficients achieved by Ts-CDD underscores its dominance over VaDER. The consistent performance of Ts-CDD indicates its robustness and adaptability, making it the preferred method for scenarios characterized by significant variations in data attributes.

V. Conclusion

In this paper, we present TsCDD-GAN, an innovative model designed to handle incomplete time series data by integrating imputation and clustering tasks. Utilizing GANs, our approach not only fills missing values but also generates data that aids in better clustering, improving the quality of clustering outcomes. Extensive experiments on UCR time series datasets show significant performance improvements, particularly in RI and ARI metrics, indicating more accurate and meaningful clusters. TsCDD-GAN also exhibits strong scalability and adaptability, making it suitable for real-world applications with dynamic data characteristics. Overall, TsCDD-GAN presents a promising solution for clustering incomplete time series data and has the potential to advance the field of time series analysis, especially where data completeness is an issue.

References

[1] Du, W. (2023). PyPOTS: a Python toolbox for data mining on Partially-Observed Time Series. *arXiv preprint arXiv:2305.18811*.

[2] Cao, W., Wang, D., Li, J., Zhou, H., Li, L., & Li, Y. (2018). Brits: Bidirectional recurrent imputation for time series. *Advances in neural information processing systems*, *31*.

[3] Luo, Y., Cai, X., Zhang, Y., & Xu, J. (2018). Multivariate time series imputation with generative adversarial networks. *Advances in neural information processing systems*, *31*.

[4] Sun, B., Ma, L., Cheng, W., Wen, W., Goswami, P., & Bai, G. (2017, October). An improved k-nearest neighbours method for traffic time series imputation. In *2017 Chinese Automation Congress (CAC)* (pp. 7346-7351). IEEE.

[5] Wang, Z., She, Q., & Ward, T. E. (2021). Generative adversarial networks in computer vision: A survey and taxonomy. *ACM Computing Surveys (CSUR)*, *54*(2), 1-38.

[6] Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, *8*(1), 6085.

[7] Du, S., Li, T., Yang, Y., & Horng, S. J. (2020). Multivariate time series forecasting via attention-based encoder–decoder framework. *Neurocomputing*, *388*, 269-279.

[8] Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). LSTM-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*.

[9] Yue, Z., Wang, Y., Duan, J., Yang, T., Huang, C., Tong, Y., & Xu, B. (2022, June). Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 36, No. 8, pp. 8980-8987).

[10] Ma, Q., Zheng, J., Li, S., & Cottrell, G. W. (2019). Learning representations for time series clustering. *Advances in neural information processing systems*, *32*.

[11] Suo, Q., Yao, L., Xun, G., Sun, J., & Zhang, A. (2019, June). Recurrent imputation for multivariate time series with missing values. In *2019 IEEE international conference on healthcare informatics (ICHI)* (pp. 1-3). IEEE.

[12] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative adversarial networks: An overview. *IEEE signal processing magazine*, *35*(1), 53-65.

[13] Dau, H. A., Bagnall, A., Kamgar, K., Yeh, C. C. M., Zhu, Y., Gharghabi, S., ... & Keogh, E. (2019). The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, *6*(6), 1293-1305.

[14] Meng, X. L., Rosenthal, R., & Rubin, D. B. (1992). Comparing correlated correlation coefficients. *Psychological bulletin*, *111*(1), 172.

[15] de Jong, J., Emon, M. A., Wu, P., Karki, R., Sood, M., Godard, P., ... & Fröhlich, H. (2019). Deep learning for clustering of multivariate clinical patient trajectories with missing values. *GigaScience*, *8*(11), giz134.