Research Article

Multivariate Time-Series Data Generation in Generative Adversarial Networks

Hira Zahid[1], Tariq Mahmood[2]

1. Department of Computer Science, Institute of Business Administration Karachi, Karachi, Pakistan; 2. Institute of Business Administration Karachi, Karachi, Pakistan

Time-series data often arises during the monitoring and evaluation of ongoing industrial processes. Time series forecasting requires accurate data modelling through the description of inherent structures such as trend, cycle, and seasonality by collecting and modeling stochastically the historical data points of a time series. In this paper, we are concerned with industrial time series data that is limited and not readily available for accurate machine learning tasks, e.g., online fraud and network intrusion data. In this scenario, modeling of time series can be achieved through generative modeling activities in deep learning. Then, abundant temporal data can be generated and used in different ways to achieve application-level forecasts and predictions. We focus on the use of Generative Adversarial Networks (GANs) to model and generate limited real-world time-series data. We discover that this is a relatively new research domain with research trends generally focusing on employing real data to generate or forecast the time series through the GAN in a supervised manner. On the contrary, we adopt a novel approach that is completely unsupervised, i.e., we employ GAN to generate limited time series data from a (gaussian) noise distribution as input without any additional input vector of real data. To achieve realistic generative performance in this situation, we propose and implement a feedback mechanism through which GAN improves its performance by using historically generated time series (and never the real data). Using different experimental configurations, we demonstrate that our approach generates realistic limited intrusion detection data from the standard CIC-IDS2017 dataset.

Corresponding authors: Hira Zahid, hzahid@iba.edu.pk; Tariq Mahmood, tmahmood@iba.edu.pk

 doi.org/10.32388/XCPE04

I. Introduction

Time-series data and its applications are recently increased in different domains such as Service demand prediction in network traffic data, scenario forecast in power generation, Oil Price forecasting, Railway passenger volume forecast, crime forecasting with sequence data, Commercial building Electricity time-series prediction, missing sensor data records in weather predictions, human behavior prediction. However, an adequate amount of data is required to produce the time series prediction and forecasting framework. [1] Existing works relying on complete information of historical data are not able enough to deal with real-time scenarios because a significant portion of historical data is not available due to many reasons such as cost, time and privacy, etc. Deep learning comes up with a data generation mechanism with its deep neural networks such as Variational Autoencoder and GANs. VAE, on one hand, uses original data with a latent code sample at the input side to process its generation process whereas GANs take the random data sample and generates the synthetic data [2].

As nonlinear mapping, GAN can learn data distribution well, and the generator of GANs can be used to deal with limited data problems and imbalanced datasets. Generative Adversarial Networks (GANs) learn the data distribution in an adversarial manner. It aims at generating the synthetic data sample indifferentiable from real data sample by jointly training its two neural networks, i.e. Discriminator and Generator.

GAN was supposed to be used in image handling problems as largely shown in the previous work. Recently, the GAN framework is extended for sequential or time-series data now, called sequence generative networks and has shown promising results in capturing the dynamics of this kind of data to benefit prediction and forecasting applications plus the generated samples assist various tasks such as data augmentation and incompleteness of data. For instance, [3] proposed a deep generative model for electronic health records data to produce labelled data using real patient records. In [4], conditional GAN with LSTM layers in both discriminator and generator was used to predict the hotspots using the previous hotspots data. To extract temporal and spatial features of data, cluster analysis (DBSCAN) was also modified to cater to noise. [5] designed a data generation algorithm to resolve the issue of imbalanced data with minority class using conditional generative adversarial network. Based on the GAN framework, proposed a method [6] to predict the building electricity data with limited data by generating the target

data using the available similar data. A conditional Generative Adversarial Network with an underlying architecture of a decoder in the Generator module is used to generate future predictions to overcome the problem of historical data incompleteness for service demand prediction [7]. A cycle-consistent generative adversarial network (CycleGAN) is used in [8] to learn daily changes of the leaf and disease from hyperspectral images using the real data at the input side of the generator. For evaluation purposes, images are aligned over time using a feature-based method. Again in [9], the input to the generator involves the use of both real data and noise signals. The authors combine adaptive scales continuous wavelet transform with supervised GAN models to forecast crude oil prices Moreover, in [10], the authors propose TadGAN which is a time-series anomaly detector which is providing a dynamic threshold for classification that uses LSTM as a base model and trained with cycle consistency loss. The proposed GAN method is used to generate sufficient abnormal data to cover all possible anomalies since the decision tree model is not robust. In [11], the authors motivate themselves by the problem of scarcity of medical data and hence use GANs to generate medical synthetic data. By comparing the performance of learning systems over the latter, the authors prove the resemblance and utility of the latter to real-world data.

In a revolutionary work [7], the authors argue that GAN-based time series generation solutions are not capable of modeling properly the temporal dynamics while the supervised sequence prediction solutions are deterministic by nature. So, they combine both approaches in their proposed TimeGAN architecture. This uses data embeddings with noise instead of real data to force the GAN to follow the stochastic dynamics of the real data to produce the time-series data. Inspired by the application of GANs, we implement them to provide an intuitive approach for generating time-series data to benefit the tasks such as prediction, forecasting, and classification tasks. The proposed method leverages the benefits of the long-short-term memory (LSTM) to learn the temporal structure of data whereas GAN is used to characterize the evolving features and enhance the ability of the model to generate the data.

II. Background Knowledge

Typically, GAN consists of a generative model i.e. Generator (G) and Discriminator (D). First Discriminator is trained with real data and fake generated data by G, then outputs the probability of the sample coming from real and fake data in order to distinguish the real data in the training set from the fake data following the mini-max game [4]. Therefore, to train a discriminator D given a generator G, it is likely to maximize the probability of the discriminator predicting the data x coming from P_{data} and minimize the probability of the discriminator predicting the generated data $G(z)$ coming from P_{data}.

$$min_G max_D V(D,G) = E_x \sim P_{data}(x)[logD(x)] + E_z \sim P_z(z)[log(1 - D(G(z)))] \quad (1)$$

On the other hand, a random sequence of noise(z) from a certain probability distribution is given to the generator (G) that tries to fool the D and outputs a quality generative sample by maximizing the probability of its own outcome.

$$max_G V(G;D) = E_z \sim P_z(z)[logD(G(z))] \quad (2)$$

where represents the distribution of real data. whereas *G*(*z*;θg) consumes some noise input and produces a piece of generated data. Therefore, $P_z(z)$ denotes the distribution of noises where normal distribution *N*(0,1) and uniform distribution *U*(0,1) are the common choices *D* and *G* are iteratively optimized using the existing binary cross entropy loss that is usually available in most of the deep learning training frameworks by setting the target probabilities for real data sequence *X* as 1.0 and the target probabilities for fake data sequence *Z* as 0.0, where 1.0 represents the data was from real for with a probability of 1.0 and 0.0 represents the data was from fake for with a probability of 0.0.

III. Proposed methodology

Our proposed methodology has two combined novel features: 1) unsupervised and 2) feedback in GAN. Both Generator and Discriminator have 3- Layers of LSTM. The reasons are that: (1) LSTM is capable of exhibiting temporal dynamics compared to feed-forward networks and CNNs; (2) LSTM utilizes three gates to protect and control the cell state, which mitigates the gradient vanishing and exploding problems compared to RNNs [11], [12] ;(3) LSTM has been used in a majority of related papers. Like the above standard GAN framework in section 2, our model also optimizes two neural networks (i.e., generative network and discriminative network D) with a minimax two-player game. In the model, D tries to distinguish the real data sequence in the training data from the sequence generated by G, while G maximizes the probability of D making a mistake and this adversarial process can eventually adjust G to generate the realistic sequence of data.

Random numbers sampled from a normal distribution N(0, 1) with the dimensions of the dataset and represented input to the generator z in the form of shifted time-series that was a vector with $Z\{z_m\left(t-k\right), z_m\left(t-k-1\right), z_m\left(t-k-2\right), \ldots, z_m\left(t-k-n\right), z_m t\}$ in order to be a multivariate normal distribution. Specifically, the noise zmzmzmz_m t at time to of a particular feature is based on its three previous timestamps values. So, G generates the data $X^{\{x_m\left(t-k\right), x_m\left(t-k-1\right), x_m\left(t-k-2\right), \ldots . x_m\left(t-k-n\right), x_m t\}}$ with the input of random noise input. Then Discriminator is trained with the real data sequence of $X\left\{x_m\left(t-k\right), x_m\left(t-k-1\right), x_m\left(t-k-2\right), \ldots . x_m\left(t-k-n\right), x_m t\right\}$ and fake generated data sequence of $X^{\{x_m(t-k), x_m(t-k-1), x_m(t-k-2), \ldots . x_m(t-k-n), x_m t\}}$ where k is the time lags of the data and the feature type = 1,2,3,...m. The problem we consider can be described as: given *Z* with its previous and current values *k*, generate the target time series with the same pattern i.e. past and current values of true data sequence *X*. In this sequence, we can add *t+1, t+2, ... t+k* values for forecasting the future values too, but for now we keep it simple. To address this problem, we propose a novel generating framework, which applies the generative adversarial learning strategy to obtain the realistic adversarially generated time series *X^* to generalize any arbitrary time series.

Relying solely on the training of the generator with single noise input may not be sufficient incentive for the generator to capture the true distributions of the data [6]. To achieve this more efficiently, we introduce an additional input through feedback to further discipline learning. First, we trained the discriminator in the same standard manner and update the generator via's discriminator error. After every 10 epochs of this training, we start observing the divergences in both data distributions. First, we initially generate the data(xfake)$data(x_{fake})$ again with noise input sequence (*Z*) and calculate the KL-divergence(kl_x_{fake}_X) between the and real data (*X*) as shown in (1). Similarly, in (2) we determine the KL-divergence(kl_x_{fake}_Z) between *Z* and x_{fake}. We then add the product of kl_x_{fake}_X and product of kl_x_{fake}_Z with *X* to compute the feedback input for the generator as mentioned in (5). Repeated the same training procedure of discriminator with newly generated samples from the generator and generator with the new input noise sequence

$$\text{kl\_}x_{fake}\text{\_X} = D_{KL}(x_{fake} \| X) \qquad (3)$$

$$\text{kl\_}x_{fake}\text{\_Z} = D_{KL}(x_{fake} \| Z) \qquad (4)$$

$$z_{new} = \text{kl\_}x_{fake}\text{\_X} * Z + \text{kl\_}x_{fake}\text{\_Z} * X \qquad (5)$$

The schematic of our proposed method is shown in Fig. 1.

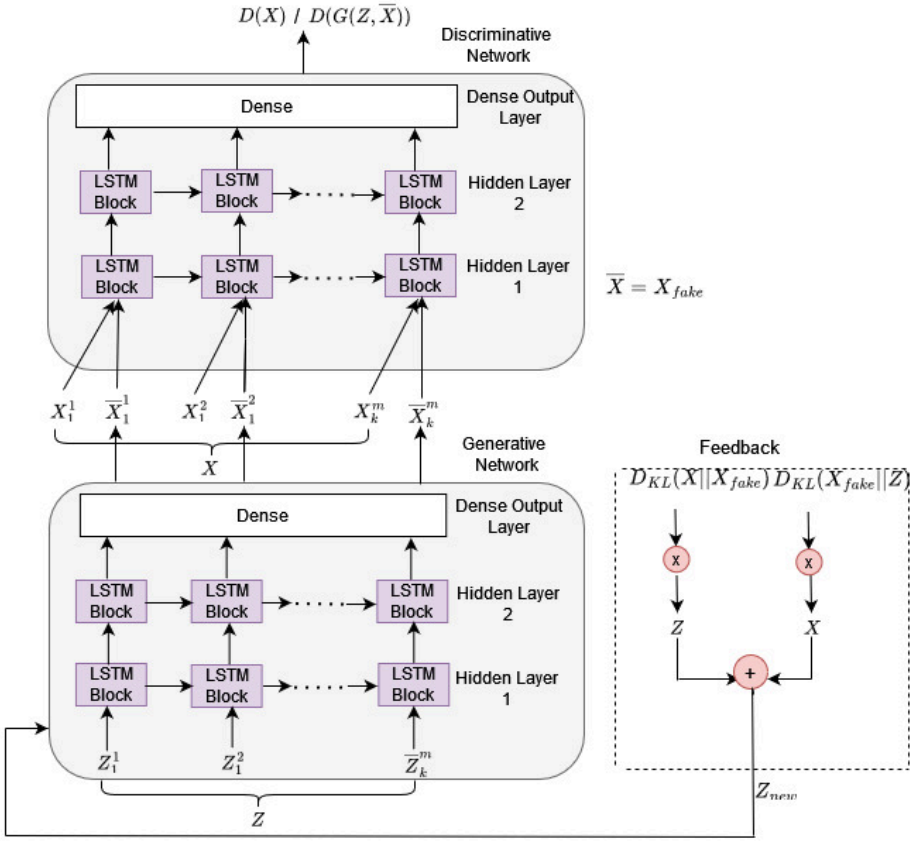

Figure 1. GAN-based Time-Series Data Generation Model

doi.org/10.32388/XCPE04

Algorithm

Generator: $G_{\square}$; Discriminator: $D_{\square}$;
Batch size: T; Model parameter: Ø ; number of epochs of adversarial process : η, Timestamps : k ; Noise data: Z ; real data : x_{real} ; Generated Data from Generator G : x_{fake} ; feedback iteration number: η_{eval} ; number of epochs of feedback = η_f

Step 1:

Transform Z into $Z\{ z_m(t-k), z_m(t-k-1), z_m(t-k-2), \ldots, z_m(t-k-n), z_m t \}$

Transform X into $X\{ x_m(t-k), x_m(t-k-1), x_m(t-k-2), \ldots. x_m(t-k-n), x_m t \}$

Initialize all the learnable model parameters ø

1. For $i=0$... η do
2. Randomly select a batch of Z and a batch of X from the real data
3. Get the predicted presentation x_{fake} of Z from Generator G
4. Train $D_{\square}$ with one batch of X_T
5. Train $D_{\square}$ with one batch of Z_T
6. Update the G via the Discriminator's error
7. If $(i+1)$ % η_{eval}; = 0;
8. Calculate KL_Div (x_{fake} , x_{real}) and Calculate KL_Div (x_{fake},z)
9. feedback = Equation (5)
10. For ($j=0$ .....η_f)
11. Generated a batch of noise data: z_T
12. Z_f = feedback + z_T
13. Repeat steps 3 till 6
14. End For
15. End For

IV. Results and Discussion

In this section, we describe a series of experiments that we have conducted to evaluate the effectiveness of our proposed method. A combination of different hyperparameters including batch size and timestamp were applied to an IDS dataset selected for this research. From this multivariate data (CIDS_2017) containing six large files we took one file which is named 'Thursday_WorkingHours-Morning-WebAttack' because of the presence of various attacked data labels. The dataset covers the time

from 8:59 am till 12:59, contains 85 features and 458968 rows which is a more suitable dataset regarding the performance of GAN. In this data, we took only the anomalous data with no labels and important features to learn the attacked pattern only. Therefore, the model was applied only on 2180 rows and 78 features. The dateTime column has been added to generate sequences of fixed-frequency dates and time spans though the Timestamp column is present however it's not uniform and cannot be directly used for pre-processing.

The GAN model has been trained using the python Library Keras[(1)], since we haven't found any single utility library to analyze time-series data using a deep-learning model. Both of the neural networks of GAN are built upon 3 Layers of LSTM followed by Leaky-ReLU with the exception of the output layer. We use batch normalization and Adam optimizer with a mini-batch size. The learning rate was set to 0.001 and epochs to 500 for both networks. Various methods are compared under the evaluation of the evaluation metric of Root Mean Square (RMSE), Mean absolute error (MAE), KL-divergence and JS-Divergence. Table I and Table II presents the results of GAN without feedback and to acquire the best hyperparameters for our model.

Methods	RMSE	MAE	KL-DIV	JS-DIV
GAN-LSTM (mini Batch16)	1.187	0.974	22.129	0.315
GAN-LSTM (mini batch=32)	1.997	0.975	21.589	0.313
GAN-LSTM (mini batch=64)	1.195	0.973	21.687	0.314

Table I. Training Results

Methods	RMSE	MAE	KL-DIV	JS-DIV
GAN-LSTM (mini batch16)	1.127	0.972	21.217	0.307
GAN-LSTM (mini batch=32)	1.133	0.965	19.846	0.298
GAN-LSTM (mini batch=64)	**1.1482**	**0.972**	**19.946**	**0.3002**

Table II. Testing Results

In these tables, we can see a smaller difference between training and testing hence there is no overfitting and underfitting problem. We picked the best mini batch size = 32 due to the stability of results during training and testing to apply our proposed methodology. However, it is quite obvious that KL-divergence which is an important metric to evaluate has a very high value as compared to other metrics because if two distributions perfectly match, $D_{KL}(p||q) = 0$ otherwise it can take values between 0 and ∞. The lower the KL-divergence value, the better we have matched the true distribution with our approximation[13]. Therefore, we introduce the KL-divergence in our method to further consider the magnitude difference between the fake and true data distribution.

A. Evaluation Metric and Baselines

There are plenty of existing error metrics, but our primary concern is with minimizing the amount of information we have to send to the distribution that preserves the most information from our original data source. *Kullback-Leibler* (KL) divergence is the expectation of the log difference between the probability of data in the original distribution with the approximating distribution.

$$D_{KL}(p||q) = E\left[logp\left(x\right) - logq\left(x\right)\right]$$

KL divergence is formally defined as follows.

$$D_{KL}(p||q) = \sum_{i=1} N\, p(xi).log\frac{p(xi)}{q(xi)}$$

Where ***q(x)*** is the approximation and ***p(x)*** is the true distribution we're interested in matching ***q(x)*** to. Intuitively this measures how much a given arbitrary distribution is away from the true distribution.

Training Model	RMSE	MAE	KL-DIV	JS-DIV
GAN-LSTM <---- JS-Div	1.075	0.789	0.663	0.032
GAN-LSTM <--- KL-Div	1.076	0.788	0.641	0.032
GAN-LSTM <---- WS-Div	1.044	0.742	0.855	0.040

Table III. Feedback Training Results

Testing Model	RMSE	MAE	KL-DIV	JS-DIV
GAN-LSTM <---- JS-Div	1.12	0.843	0.244	0.0232
GAN-LSTM <--- KL-Div	1.111	0.824	0.177	0.0203
GAN-LSTM <---- WS-Div	0.981	0.628	1.616	0.071

Table IV. Testing Feedback Results

We have applied three divergences i.e. *Jensen–Shannon* divergence (JS), *Kullback-Leibler*(KL) divergence and *Wasserstein distance* (WS) in our method as mentioned in equation 3 and equation 4. The feedback mechanism with KL-div has shown superior results over the other two divergences particularly with the evaluation metric of KL-divergence and JS-divergence as shown in Table III and Table IV. RMSE was improved by 0.01% with **JS-divergence** during training and 0.001 during testing, MAE was improved by 0.002 during training and 0.13 during testing. 0.21% in KL-div during training and 0.19 % in testing.JS-div metric in was improved by 0.003% during training and testing. With **KL-divergence and Wasserstein Distance**, RMSE was improved by 0.01 during training and 0.002% during testing, MAE was improved by 0.18 during training and 0.14% during testing. 0.215% in KL-div during training and .196 % in testing.JS-div metric in was improved by 0.003% during training and testing with both **KL-divergence and Wasserstein Distance**. MAE was improved by 0.0023 during training and 0.0034% during testing With **Wasserstein Distance**. 0.213% in KL-div during training and 0.18 % in testing.JS-div metric in was improved by 0.003% during training and testing.

The tables V, VI, VII, VIII, and IX reveal the accuracy, precision, and F1-score of LSTM, Random Forest, Xgboost, Support Vector machine, Decision Tree. We compare the proposed GAN feedback model with the above-mentioned classification approaches. Initially, we generated the fake anomalous data from the generator, applied PCA to reduce the dimensions to make it easier for the model to classify and merged with the original PCA applied benign data to attain the classical results, then perform the classification task on the original data of IDS.

	Precision	Recall	F1-score	Accuracy
BENIGN	1.00	1.00	1.00	0.992
ATTACK	0.68	0.69	0.69	

Table V. LSTM

	Precision	Recall	F1-score	Accuracy
BENIGN	1.00	1.00	1.00	0.995
ATTACK	0.78	0.90	0.84	

Table VI. Decision Tree

	Precision	Recall	F1-score	Accuracy
BENIGN	1.00	1.00	1.00	0.986
ATTACK	0.85	0.91	0.88	

Table VII. Random Forest

doi.org/10.32388/XCPE04

	Precision	Recall	F1-score	Accuracy
BENIGN	0.99	1.00	0.99	0.876
ATTACK	0.83	0.03	0.06	

Table VIII. Support Vector Machine

	Precision	Recall	F1-score	Accuracy
BENIGN	1.00	1.00	1.00	0.985
ATTACK	0.79	0.90	0.84	

Table IX. XGBOOST

Conclusion

In this paper, we introduce a novel method for time-series generation that combines the unsupervised GAN approach with the feedback mechanism. Leveraging the contributions of the unsupervised loss and trained with random noise generated data as a feedback loop based on KL-divergence, our model demonstrates consistent and significant results when compared with the classification models which shows that our model is able to generate realistic time series data. In the future, further work may investigate incorporating the prediction datasets into our model in order to generate high-quality time-series data over state-of-the-art benchmarks in generating realistic time-series data.

Footnotes

(1) https://keras.io/

Additional References

- "Time-series generative adversarial networks - Google Search." https://www.google.com/search?client=firefox-b-d&q=Time-series+generative+adversarial+networks (accessed Apr. 16, 2022).

References

1. ^*A. Koochali, P. Schichtel, A. Dengel, and S. Ahmed, "Probabilistic Forecasting of Sensory Data With Generative Adversarial Networks – ForGAN," IEEE Access, vol. 7, pp. 63868–63880, 2019, doi: 10.1109/ACCESS.2019.2915544.*
2. ^*S. Takahashi, Y. Chen, and K. Tanaka-Ishii, "Modeling financial time-series with generative adversarial networks," Physica A: Statistical Mechanics and its Applications, vol. 527, p. 121261, Aug. 2019, doi: 10.1016/j.physa.2019.121261.*
3. ^*Z. Che, Y. Cheng, S. Zhai, Z. Sun, and Y. Liu, "Boosting Deep Learning Risk Prediction with Generative Adversarial Networks for Electronic Health Records." arXiv, Sep. 05, 2017. doi: 10.48550/arXiv.1709.01648.*
4. a, b*"Extracting and Predicting Taxi Hotspots in Spatiotemporal Dimensions Using Conditional Generative Adversarial Neural Networks | IEEE Journals & Magazine | IEEE Xplore." https://ieeexplore.ieee.org/document/9023953 (accessed Apr. 14, 2022).*
5. ^*G. Douzas and F. Bacao, "Effective data generation for imbalanced learning using conditional generative adversarial networks," Expert Systems with Applications, vol. 91, pp. 464–471, 2018, doi: https://doi.org/10.1016/j.eswa.2017.09.030.*
6. a, b*Y. Pang et al., "Generative Adversarial Learning Based Commercial Building Electricity Time Series Prediction," in 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), Nov. 2019, pp. 1800–1804. doi: 10.1109/ICTAI.2019.00271.*
7. a, b*S. Ma, S. Guo, K. Wang, and M. Guo, "Service Demand Prediction with Incomplete Historical Data," in 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Jul. 2019, pp. 912–922. doi: 10.1109/ICDCS.2019.00095.*
8. ^*"Hyperspectral plant disease forecasting using generative... - Google Scholar." https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Hyperspectral+plant+disease+forecasting+using+generative+adversarial+networks&btnG= (accessed Apr. 14, 2022).*

9. ^“Oil price forecasting using supervised GANs with continuous wavelet transform features - Google Search.” https://www.google.com/search?client=firefox-b-d&q=Oil+price+forecasting+using+supervised+GANs+with+continuous+wavelet+transform+features (accessed Jun. 25, 2022).
10. ^A. Geiger, D. Liu, S. Alnegheimish, A. Cuesta-Infante, and K. Veeramachaneni, “TadGAN: Time Series Anomaly Detection Using Generative Adversarial Networks,” arXiv:2009.07769 [cs, stat], Nov. 2020, Accessed: Apr. 14, 2022. [Online]. Available: http://arxiv.org/abs/2009.07769
11. a, bS. Dash, A. Yale, I. Guyon, and K. P. Bennett, “Medical time-series data generation using generative adversarial networks,” in International Conference on Artificial Intelligence in Medicine, 2020, pp. 382–391.
12. ^W. Wu, F. Huang, Y. Kao, Z. Chen, and Q. Wu, “Prediction Method of Multiple Related Time Series Based on Generative Adversarial Networks,” Information, vol. 12, no. 2, Art. no. 2, Feb. 2021, doi: 10.3390/info12020055.
13. ^“Kullback-Leibler Divergence Explained,” Count Bayesie. http://www.countbayesie.com/blog/2017/5/9/kullback-leibler-divergence-explained (accessed Jun. 24, 2022).

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.