Time-Series Generative Adversarial Networks for Cyber-Physical Systems

Minh B. T. Nguyen*
Virginia Tech
Blacksburg, VA 24060
mnguyen0226@vt.edu

Md Nazmul Kabir Sikder*
Virginia Tech
Arlington, VA 22203
nazmulkabir@vt.edu

Chelsea Wang*
Virginia Tech
Arlington, VA 22203
chelseawang@vt.edu

Abstract

Modern water treatment and distribution systems rely on digital controls, sensors, actuators, and the internet of things for both monitoring and operational applications. Combining physical processes with embedded technology (cyber-physical systems (CPS)) improve the quality of water services; however, increases the likelihood of being attacked by cybercriminals. In recent years, several deep learning models have success in attacks detection on minimal cyber-physical test-bed; however, the deployment rate is low due to the lack of testing in realistic datasets. Due to privacy and ethical considerations, real datasets are not publicly available for Machine Learning model evaluation. This paper presents Synthetic cyber-physical Water data generation with with Generative Adversarial Network (GAN), inspired by the previous Time GAN framework for synthetic traffic data generator; a tool to generate almost realistic but synthetic dataset streams for capturing temporal dependencies in time series water distribution and water treatment applications. TimeGAN is capable of capturing temporal dependencies at any given time. We test our hypothesis if synthetic data can substitute real cyber-physical datasets to train models of comparable accuracy. We present that synthetic temporal-preserved multivariate time-series data can replace real time sequence one during training stage of a regression task. We test our hypothesis on two popular benchmark water distribution system dataset with appropriate experimentation and the results are presented with comprehensive explanation. Our works is published at here[1].

1 Introduction

Water Distribution Systems (WDS) are getting advanced and digital because of recent advancement of technology and communication. Unlike before, now operators can observe and control pressure in the valve, water flow, quality in chemical mixture for water purification using remote controlled sensors and actuators. These controllable units are also known as Programmable Logic Controller (PLC); thanks to the advancement of technology, maintenance and repairing became piece of cake due to advanced PLCs. Unfortunately, the communication and physical components are exposed to cyber threats. Good news is, Deep Learning (DL) methods have gained popularity and platform to understand complex system especially WDS. However, due to insufficient data, assurance and generalizability check on these DL methods have become challenging. Hence, these DL models are yet to be popular in the WDS domain. There are many reason why real data is not abandoned, and one main reason is because privacy preservation. Exposing real data lets the system more prone to be attacked. Therefore, we need tools that can provide us synthetically generated dataset which are real

[1] https://github.com/mnguyen0226/tsgans_cps

*Equal Contribution. Affiliate with Department of Electrical and Computer Engineering.
Trustworthy Machine Learning Final Project (ECE 5984 Spring 2022), Blacksburg, Virginia, USA.

enough to test DL models for generalizability and robustness check. Only then, the DL models can be trustworthy enough to be deployed in the real application.

1.1 Motivation

Cyber-physical attacks are rapidly increasing while damaging components of the distribution system. On February 5th, 2021, Florida's water treatment plant was reported to be compromised; in which the hacker altered the levels of sodium hydroxide of the 15,000-person city of Oldsmar. The chemical could have severely damaged any human tissue it touches at high levels. To keep up the research community with cyber threats in WDS community, solutions have been proposed via a platform such as BATtle of the Attack Detection ALgorithms (BATADAL). BATADAL is an international cyber-physical attacks detection competition launched in 2016 [17] that serves as a benchmark to solve cyber attacks especially concealed attacks in WDS. However, these solutions aren't yet widely adopted and brought into limelight due to the lack of testing with real cyber-physical datasets.

The real cyber-physical datasets are not publicly available; the bottleneck here is privacy violations, ethics, and expense. Additionally, they expose the critical infrastructure to the attackers. For instance, Virginia's Blacksburg-Christianburg wastewater treatment plant is unlikely to publicize the data, even if the wastewater is adequately treated. The denial of datasets is understandable since it is inconsiderate if the local community knows the percentage of people using illegal substances. Organizations even hesitate in sharing data internally due to the risk of leaking private information. Furthermore, realistic water treatment/distribution testbeds have been built, such as Secure Water Treatment (SWaT) [12] or SCADA (BATADAL competition dataset: C-town WDS) [5]; however, collecting data or testing is costly [7] [1]. Lastly, generating artificial cyberattacks on real CPS seriously violates ethical considerations since the systems may unintentionally harm the community.

The reasons mentioned above are wake-up calls for machine learning researchers like us to provide an effective, inexpensive, and comparable approach for cyber-physical water datasets.

1.2 Our Contribution

In this section we discuss about our contribution to this project. We apply TimeGAN model for preserving temporal dynamics, therefore the synthetic data sequence reflects similar properties as real data sequence. Controlled by supervised training, the TimeGAN exploits the flexibility of unsupervised learning and producing synthetic datasets. Both supervised and adversarial objectives learns joint optimization with the embedding space, the network is a reflection of the training objects's dynamics during sampling. Empirically, we generate synthetic dataset and test them qualitatively and quantitatively.

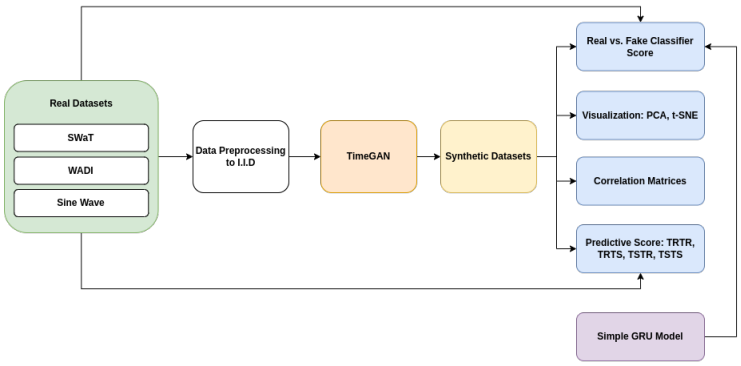

Figure 1: End-to-end Temporal-Preserved Synthetic Time-Series Data Generation Pipeline.

A major part of this project is to select appropriate dataset for our hypothesis test. For a toy example we use Sine wave as univariate time-series data. Additionally, two real datasets are used for experimentation including Secure Water Treatment (SWaT) and SCADA from the University of Singapore [7] and BATADAL Competition [17] respectively. SWaT is *"an operational test-bed for water treatment that represents a small-scale version of a large modern water treatment plant found in a large city"* [12]. The 2nd dataset is from a water distribution system C-town network from

BATADAL competition that have regular water flow information as well as concealed attacks [5] on different physical layer component including pump, PLC, sensor etc.

Both SWaT and SCADA datasets contain several data types: continuous, discrete, categorical, or numeric. We exploit the unsupervised GAN framework's versatility by using autoregressive models. We perform t-SNE and PCA analysis [18] to determine the degree to which the synthetic dataset accurately and qualitatively resembles the real dataset. Then, we quantify the ability of a post-hoc classifier to distinguish generated dataset from the real data. Furthermore, we estimate how well the generated data retains the properties of the original data by using the *"train on synthetic, test on real (TSTR)"* framework [6] to the sequence prediction problem. After a comprehensive experimentation, we observe that for SWaT and SCADA dataset synthetic data generation, TimeGAN shows good performance.

We want to test two research questions as follows,

1. Given a well developed time generative adversarial network (TimeGAN), can we generate water distribution system multivariate time sequential dataset of comparable accuracy?
2. How can we evaluate synthetic time-series data visually and qualitatively?

We develop a pipeline using TimeGAN for the SWaT and BATADAL dataset to test our research question. Figure 1 represents the block diagram of the major workflow of the model. First, the datasets are normalized and converted into independent and identically distributed (IID) random variables by taking a window (row-number) of size 24 and slide it along the rows of the dataset. The idea is to shift the window by one row at a time, thus we will obtain a batch of IID data with the shape of 24xc (c is the number of columns in the initial dataset). Next, we build TimeGAN from scratch via Tensorflow 2.x and train TimeGAN on real, processed dataset. TimeGAN will then provide synthetic, IID datasets that preserve temporal-dependencies of the original corresponding datasets. Lastly, various testing and evaluating methods will be applied to check for the quality, usefulness, and fidelity of the synthetic datasets.

2 Literature Review

Data augmentation or data generation is a popular research area. For successful data augmentation purpose, two major objectives are considered with high priority: privacy preservation and distribution relevancy with original data so that they are useful for deployment purpose. Balancing between these two objective is tricky as privacy preservation may reduce utility of the augmented data. For the scope of this project, we are more interested in capturing the later objective. We need models that not only capture variable distribution across time points but also understand complex potential variable interdependence across time. For instance, let $x_{1:T} = (x_1, ..., x_T)$ be the multivariate sequential data, we want to capture the conditional distribution of temporal transitions $p(x_t|x_{1:t-1})$ accurately.

2.1 Synthetic Data Generation on Multivariate Time-Series

In the literature, there are several approaches for augmentation of time series data. Traditional modeling approaches, on the other hand, are constrained by distribution data types; and are vulnerable to computational difficulties, limiting the reliability of synthetic data created by these models [2] [3] [16]. Additionally, GAN-based approaches are popular for enhancing performance and data generation flexibility [10] [11] [19]. The published GAN experiments are limited to a static dependency, ignoring the temporal dependency that is common in real-world data [14] [19]. On the other hands, there are attempts in partially building temporal dependence inside GAN blocks; however, there are limits on the temporal dependence in a pass that GAN generates a time [9] [10].

For water distribution system application, there are only handful amount of research on either synthetic data generation. Zhou et al [22] aims to solve the problem of scarcity of attack industrial control datasets via membership distribution, association grouping, and GANs. The authors claim that the amount of attacks is "increased by 100 percent" [22] from last couple of decades. The authors only validate the results by training and testing the accuracy percentage between the original datasets with synthetic datasets using Support Vector Machine (SVM) and eXtreme Gradient Boosting (XGBoost) [22]. Unfortunately, the framework proposed by Zhou et al is very computationally expensive, because, increasing the amount of anomaly data points (30% of the BATADAL dataset) by 100

Methods	Capture Attribute Dependence	Capture Temporal Dependence	(Continuous) Multivariate Generation	Categorical Variable
WP-GAN [13]	Yes	No	No	Yes
CTGAN [19]	Yes	Partially	Yes	Yes
ODDS [8]	Yes	Partially	No	No
MTSS-GAN [15]	Yes	Yes	Yes	No
TimeGAN [21]	Yes	Yes	Yes	No
TimeVAE [4]	Yes	Yes (better than TimeGAN)	Yes	Yes
DoppelGANger [10]	Yes	Partially	Yes	Yes
STAN [20]	Yes	Yes	Yes	Yes

Table 1: A Comparison of Synthetic Data Generation Methods

times, it is required to generate 100 times the mixed in normal data points [17]. While generating synthetic WDS data, we need to measure the similarity and diversity of the newly generated samples against the original samples; which is still a ongoing research. We use TimeGAN to fill that research gap for WDS synthetic data generation. Table 1 summarizes few popular GAN based synthetic data generation methods and their features.

3 Materials and Methods

In this section, we present the core architecture of out pipeline, the TimeGAN.

3.1 Time-Series Generative Adversarial Networks

For our study, the architecture we apply is a Time-Series Generative Adversarial Network (TimeGAN) designed and developed by Yoon et al. [21]. The pipeline tries to achieve two important objective including local and global joint distribution matching by breaking down the the sequence level objective into a step-wise objectives. Following equation provides global objective where D represents measure of distance between distributions.

$$\min_{\hat{p}} D\left(p\left(\mathbf{S}, \mathbf{X}_{1:T}\right) \| \hat{p}\left(\mathbf{S}, \mathbf{X}_{1:T}\right)\right) \tag{1}$$

Another objective is local for any t [Eq: 2].

$$\min_{\hat{p}} D\left(p\left(\mathbf{X}_t \mid \mathbf{S}, \mathbf{X}_{1:t-1}\right) \| \hat{p}\left(\mathbf{X}_t \mid \mathbf{S}, \mathbf{X}_{1:t-1}\right)\right) \tag{2}$$

TimeGAN has four major components as follows:

1. Embedding Function,
2. Recovery Function,
3. Sequence Generator, and
4. Sequence Discriminator.

TimeGAN trains the autoencoding component with the adversarial components in a way that the model parallelly learns to encode the feature, produce synthetic data and iterate over time. The latent space are provided by the embedding network where the adversarial network learns from the latent space. For supervised loss estimation both real and synthetic data latent space are synchronized.

3.1.1 Embedding and Recovery Functions

The embedding and recovery functions helps the adversarial network to learn from the lower dimensional representation by providing mapping between latent and feature space. The latent vector spaces $\mathcal{H}_{\mathcal{S}}, \mathcal{H}_{\mathcal{X}}$ are related to feature spaces S, X. The embedding function $e : \mathcal{S} \times \prod_t \mathcal{X} \rightarrow \mathcal{H}_{\mathcal{S}} \times \prod_t \mathcal{H}_{\mathcal{X}}$ then converts static and temporal characteristics to latent codes $\mathbf{h}_{\mathcal{S}}, \mathbf{h}_{1:T} = e\left(\mathbf{s}, \mathbf{x}_{1:T}\right)$.

$$\mathbf{h}_{\mathcal{S}} = e_{\mathcal{S}}(\mathbf{s}), \quad \mathbf{h}_t = e_{\mathcal{X}}\left(\mathbf{h}_{\mathcal{S}}, \mathbf{h}_{t-1}, \mathbf{x}_t\right) \tag{3}$$

Where, for static features, the embedding network is $e_{\mathcal{S}} : \mathcal{S} \to \mathcal{H}_{\mathcal{S}}$; and for temporal features, the recurrent embedding network is $e_{\mathcal{X}} : \mathcal{H}_{\mathcal{S}} \times ,H_{\mathcal{X}} \times \mathcal{X} \to \mathcal{H}_{\mathcal{X}}$. In contrary, the recovery function take temporal and static codes back to their feature characteristics.

$$r : \mathcal{H}_{\mathcal{S}} \times \prod_t \mathcal{H}_{\mathcal{X}} \to \mathcal{S} \times \prod_t \mathcal{X}, \tilde{\mathbf{s}}, \quad \tilde{\mathbf{x}}_{1:T} = r\left(\mathbf{h}_{\mathcal{S}}, \mathbf{h}_{1:T}\right) \tag{4}$$

$$\tilde{\mathbf{s}} = r_{\mathcal{S}}\left(\mathbf{h}_s\right), \quad \tilde{\mathbf{x}}_t = r_{\mathcal{X}}\left(\mathbf{h}_t\right) \tag{5}$$

Where, $r_{\mathcal{S}} : \mathcal{H}_{\mathcal{S}} \to \mathcal{S}$ and $r_{\mathcal{X}} : \mathcal{H}_{\mathcal{X}} \to \mathcal{X}$ are recovery network for temporal embedding and static network for temporal embedding respectively.

3.1.2 Sequence Generator and Discriminator

The generator here first outputs synthetic data in embedding space instead of feature space directly. Let $\mathcal{Z}_{\mathcal{S}}, \mathcal{Z}_{\mathcal{X}}$ are two vector space onto which the prior distributions are defined, then random vectors are selected from them to generate $\mathcal{H}_{\mathcal{S}}, \mathcal{H}_{\mathcal{X}}$. Then tuple of static and temporal random vectors are selected by the generating $g : \mathcal{Z}_{\mathcal{S}} \times \prod_t \mathcal{Z}_{\mathcal{X}} \to \mathcal{H}_{\mathcal{S}} \times \prod_t \mathcal{H}_{\mathcal{X}}$ function to synthetic latent codes $\hat{\mathbf{h}}_{\mathcal{S}}, \hat{\mathbf{h}}_{1:T} = g\left(\mathbf{z}_{\mathcal{S}}, \mathbf{z}_{1:T}\right)$.

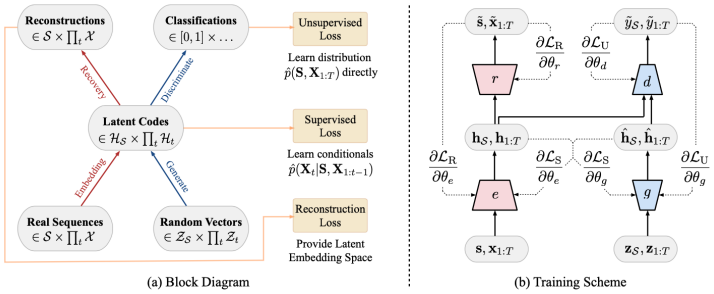

Figure 2: (a) Block diagram of different functionalities. (b) Training scheme; solid and dashed lines represent forward and backward propagation of gradients [21]

$$\hat{\mathbf{h}}_{\mathcal{S}} = g_{\mathcal{S}}\left(\mathbf{z}_{\mathcal{S}}\right), \qquad \hat{\mathbf{h}}_t = g_{\mathcal{X}}\left(\hat{\mathbf{h}}_{\mathcal{S}}, \hat{\mathbf{h}}_{t-1}, \mathbf{z}_t\right) \tag{6}$$

A recurrent network is used to implement g, where for static features, generator network is $g_{\mathcal{S}} : \mathcal{Z}_{\mathcal{S}} \to \mathcal{H}_{\mathcal{S}}$; and for temporal features, recurrent generator is $g_{\mathcal{X}} : \mathcal{H}_{\mathcal{S}} \times \mathcal{H}_{\mathcal{X}} \times \mathcal{Z}_{\mathcal{X}} \to \mathcal{H}_{\mathcal{X}}$. From a distribution of choice, random vector $\mathbf{z}_{\mathcal{S}}$ can be sampled, and a stochastic process is followed by $\mathbf{z}_t$. Next, Wiener process and Gaussian distribution are used here. Finally, the discriminator also functions from the embedding space. The static and temporal codes are received and return classifications $\tilde{y}_{\mathcal{S}}, \tilde{y}_{1:T} = d\left(\tilde{\mathbf{h}}_{\mathcal{S}}, \tilde{\mathbf{h}}_{1:T}\right)$ by the discrimination function $d : \mathcal{H}_{\mathcal{S}} \times \prod_t \mathcal{H}_{\mathcal{X}} \to [0,1] \times \prod_t [0,1]$. The $\tilde{\mathbf{h}}_*$ symbol dictates either synthetic $(\hat{y}_*)$ or real (h $\mathbf{h}_*$ embeddings; similarly, the $\tilde{y}_*$ symbol dictates either synthetic $(\hat{y}_*)$ or real (y_*) classifications data. A bidirectional recurrent network is then introduced to implement d with a feedforward output layer,

$$\tilde{y}_{\mathcal{S}} = d_{\mathcal{S}}\left(\tilde{\mathbf{h}}_{\mathcal{S}}\right) \quad \tilde{y}_t = d_{\mathcal{X}}\left(\overleftarrow{\mathbf{u}}_t, \overrightarrow{\mathbf{u}}_t\right) \tag{7}$$

where $\overleftarrow{\mathbf{u}}_t = \overleftarrow{c}_{\mathcal{X}}\left(\tilde{\mathbf{h}}_{\mathcal{S}}, \tilde{\mathbf{h}}_t, \overleftarrow{\mathbf{u}}_{t+1}\right)$ and $\overrightarrow{\mathbf{u}}_t = \vec{c}_{\mathcal{X}}\left(\tilde{\mathbf{h}}_{\mathcal{S}}, \tilde{\mathbf{h}}_t, \overrightarrow{\mathbf{u}}_{t-1}\right)$ respectively represent the sequences of backward and forward hidden states; $d_{\mathcal{S}}, d_{\mathcal{X}}$ are output layer classification functions and $\vec{c}_{\mathcal{X}}, \bar{c}_{\mathcal{X}}$ are recurrent functions. No other restrictions are applied on the architecture, a standard recurrent formulation is used to get the generator being an autoregressive model.

3.1.3 Jointly Learning to Encode, Generate, and Iterate

TimeGAN takes into account the autoregressive character of time series by combining the unsupervised adversarial loss on both actual and synthetic data with a stepwise supervised loss on the original data. The objective is to incentivize the model for acquiring knowledge of the distribution across time transitions from one point in time to the next in the historical data.

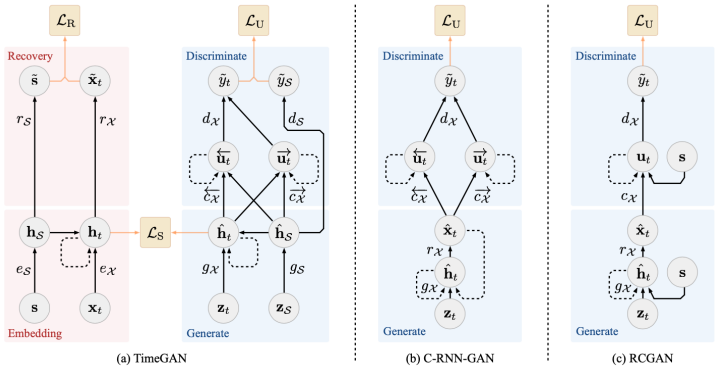

Figure 3: (a) TimeGAN with RNNs, (b) C-RNN-GAN, and (c) RCGAN. Orange lines show function application and recurrence, whereas solid and dashed lines imply loss estimation. [21]

The TimeGAN learns through three stages. Firstly, embedding and recovery functions are trained using the reconstruction loss defined below in the embedding phase.

$$\mathcal{L}_R = \mathbb{E}_{s,x_{1:T}} : [\|s - \tilde{s}\|_2 + \sum_t \|x_t - \tilde{x_t}\|_2] \tag{8}$$

The generator is fed synthetic embeddings and is tasked with creating the next synthetic vector. The unsupervised loss is constructed using the gradient that enables the discriminator to maximize (or the generator to minimize) the probability of supplying proper classifications.

$$\mathcal{L}_U = \mathbb{E}_{s,x_{1:T}}[\log y_s + \sum_t \log y_t] + \mathbb{E}_{s,x_{1:T}}[1 - \log y_s + \sum_t \log 1 - y_t] \tag{9}$$

Lastly, all components are trained jointly. The model constructs a time series embedding space by maximizing both supervised and adversarial objectives that drive it to conform to the dynamics seen during training. It combines the first two phases and adds additional loss to facilitate the generator further to capture the stepwise conditional distributions from the historical data.

$$\mathcal{L}_S = \mathbb{E}_{s,x_{1:T}}[\sum_t \|h_t - g_{\mathcal{X}}(h_S, h_{t-1}, z_t)\|] \tag{10}$$

3.1.4 Optimization

In comparison to other techniques that make use of the GAN framework, TimeGAN inherently incurs computational costs due to the addition of components to the architecture.

The following procedure is used to train reconstruction and supervised losses, in which a is hyperparameter that balances the two losses. It contains a, which reduces the dimension of the adversarial learning space and also assists the generator in discovering temporal correlations in the data.

$$min(a\mathcal{L}_S, \mathcal{L}_R) \tag{11}$$

Simultaneously, the generator and discriminator networks are trained adversarially in the following manner. b is an additional hyperparameter that compensates for the two losses, that further help the generator to minimize the supervised loss.

$$min(b\mathcal{L}_S, max\mathcal{L}_U) \tag{12}$$

4 Experiment Work

In this section we present our experimental design for testing our research questions. Here, four criteria are used to assess the quality of generated data: (1) diversity—samples should be dispersed evenly throughout the real data set; (2) fidelity—samples should be indistinguishable from the original data set; (3) usefulness—samples should be as helpful as the real data set when employed for the same prediction objectives; (4) ideal synthetic data should preserve spatio-dependency of the original dataset.

Visualization: We visualize both original and synthetic dataset using t-SNE and PCA by flatenning the temporal dimension. Both t-SNE and PCA projects multi-dimensional feature space onto a 2D space for visualization purpose. This presentation distates how well the distribution of generated samples are closely resembling to the original samples distribution in a qualitative assessment.

Discriminative Score: For similarity measurement, quantitatively, we distinguish and classify between original and synthetic dataset. For this, we manually label original data as one class and the other one as another class. A simple recurrent neural network (RNN) containing of six gated recurrent units is selected to classify real or synthetic data points.

Predictive Score: Another quantitative assessment is to investigate the usefulness of the generated synthetic data. The generated synthetic data should exhibit the predictive properties of the underlying real-world dataset. In another word, application of TimeGAN should be able to inherit conditional distribution from real data onto the synthetic data over time. Hence, we apply a simple RNN again containing six gated recurrent units for predicting next sequence information for current sequence. This test provided usefulness of the generated synthetic dataset in a quantitative manner.

Correlation Matrix: Lastly, we want to check whether the synthetic data can preserve spatio-dependency of the original multivariate time-series of CPS. We do not expect the synthetic dataset to have spatial preservation characteristic.

5 Results and Discussions

In this section, we discuss our results in terms of visual diversity assessment, fidelity assessment, usefulness, and spatio correlation check for the synthetic multivariate time-series data.

5.1 Diversity Assessment

Both PCA and t-SNE are great visual methods to see the distribution of the high-dimension dataset in 2D map. Ideally, the distribution of the synthetic samples should roughly match the real data. To begin the experimentation, for univariate time-series such as Sine wave, we can see in figure 4 that the synthetic dataset is able to resemble the distribution of the original Sine wave. This proves that TimeGAN can produce synthetic univariate time-series dataset with the same distribution of the original dataset.

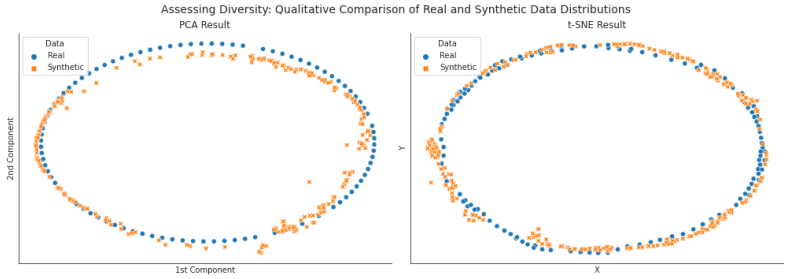

Figure 4: Visualization of (a) PCA with Sine wave, (b) t-SNE with Sine wave

Figure 5(a) shows the nature of the BATADAL dataset using t-SNE and PCA techniques, which is very noisy. We further discuss in later subsections about why this noises create a problem for TimeGAN to generate synthetic time-series dataset. Regardless, we can see that the synthetic dataset can resemble the distribution and the sparse behaviour of the original BATADAL dataset.

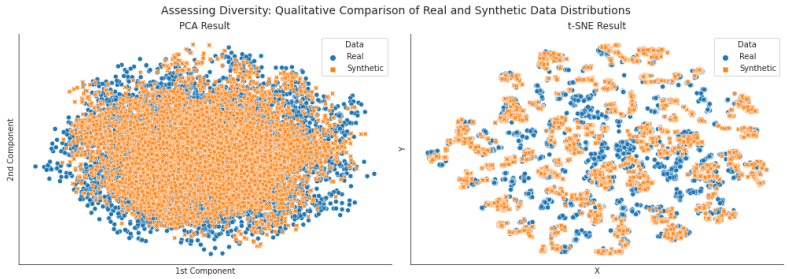

Figure 5: Visualization of (a) PCA with BATADAL dataset, (b) t-SNE with BATADAL dataset

Like the synthetic BATADAL dataset, the synthetic SWaT dataset i6 is able to resemble the distribution of the original SWaT dataset too. From figure 5 and 6, we observe TimeGAN can produce synthetic multivariate time-series dataset with the same distribution of the original dataset.

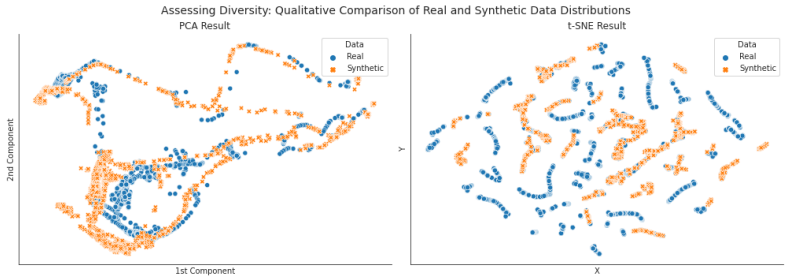

Figure 6: Visualization of (a) PCA with SWaT dataset, (b) t-SNE with SWaT dataset

5.2 Fidelity Assessment

For the fidelity assessment, we want to prove that, the synthetic data should not be indistinguishable from the real dataset. We use a simple RNN containing six gated recurrent units which serves as a classifier for real or synthetic data for each time stamps. Ideally, we want to see if the RNN is unable to classify between the real or synthetic data; that is, the training accuracy is high while the testing accuracy is low.

To begin, in terms of univariate Sine wave, we can see in figure 7 that the RNN performs slightly poorly in the testing set. This indicates that our synthetic dataset are not good enough to fool a RNN classifier. Here, the area under the curve (AUC) and testing accuracy are 0.896 and 0.798 respectively. This result can be improved further by adding more training time for TimeGAN.

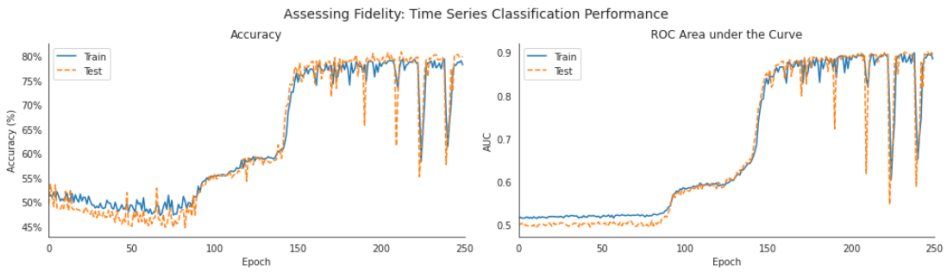

Figure 7: Accuracy and AUC scores of Sine wave

For the BATADAL dataset, in figure 5, we observe that the classifier does a good job on both AUC score and testing accuracy with 0.98 and 0.98 respectively. As we mention in the previous section, due to sparse nature of the BATADAL samples, its synthetic version are also sparse. Since, there exists barely any certain structure but mostly random noise, the classifier can easily point out the the real from synthetic data. This result can be further improved by adding more training time for TimeGAN as well. Especially, the long-trained TimeGAN can capture the data outside of the rim of PCA shown in figure 8.

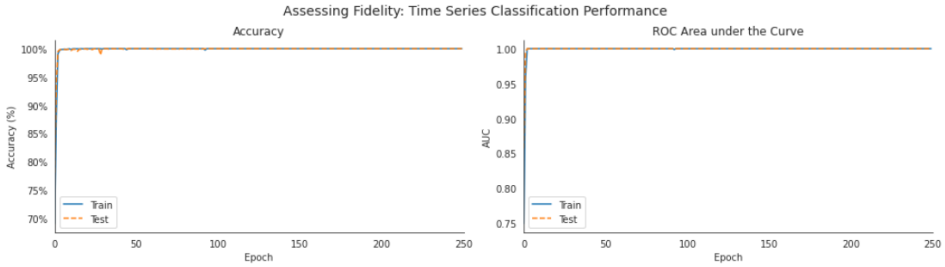

Figure 8: Accuracy and AUC scores of BATADAL dataset

For the SWaT dataset,in figure 9, we observe that the RNN performs slightly poorly in the test set. This indicates that, generated synthetic dataset may fool a RNN classifier. Here, the AUC and testing accuracy are 0.8876 and 0.863 respectively. This results can also be further improved by adding more training time for TimeGAN.

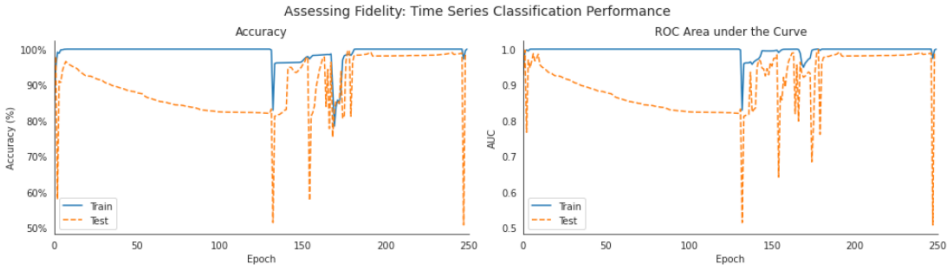

Figure 9: Accuracy and AUC scores of SWaT dataset

5.3 Usefulness

Next, we want to check whether the synthetic data are useful enough to replace the real one in predictive task. We compare the test errors of a sequence prediction model, trained on real or synthetic data, to predict the next time step for real data. There are four tests that we perform as follows,

1. Train on real dataset, test on real dataset (TRTR): The testing result for this test acts as benchmark for TSTR, since we are interested in whether synthetic dataset can replace real one in the training process.
2. Train on real dataset, test on synthetic dataset (TRTS): We are interested in whether synthetic dataset can replace real one in the testing process.
3. Train on synthetic dataset, test on real dataset (TSTR): We are interested in whether synthetic dataset can replace real one in the training process.
4. Train on synthetic dataset, Test on synthetic dataset (TSTS): We want to see how synthetic dataset performs in both training and testing process.

Amongst the four tests, we only focus on TRTR and TSTS tests since we want to observe if synthetic dataset can replace the real one for the training phase; so that we can deploy the model to the real world. For univariate time-series such as figure 10 we can see that the training accuracies on both real and synthetic datasets are similar while the testing accuracies are not too different. These results

indicate that synthetic dataset can indeed replace the real one for the training phase of univariate time-series data.

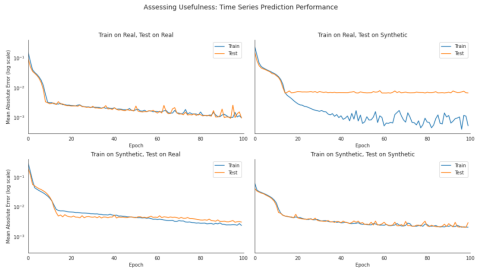

Figure 10: TRTR, TRTS, TSTR, TSTS for univariate Sine wave

For multivariate time-series such as figure 11 we observe that the training accuracy on both real and synthetic datasets are similar while the testing accuracy are not too different. These results indicate that synthetic dataset can indeed replace the real one for the training phase of multivariate time-series data

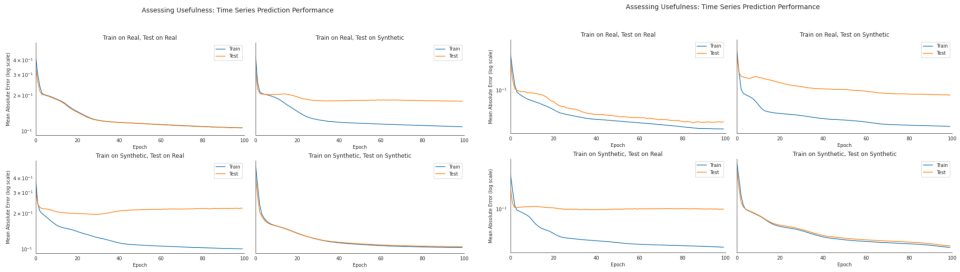

Figure 11: TRTR, TRTS, TSTR, TSTS for multivariate BATADAL and SWaT datasets

5.4 Correlation Check

Finally, we observe whether the synthetic multivariate time-series can keep the spatio-dependency of the original one. From both figure 12 and 13, we observe that the synthetic dataset are unable to preserve spatio-dependency either in BATADAL dataset or SWaT dataset. This is an open research area for future synthetic multivariate time-series generation.

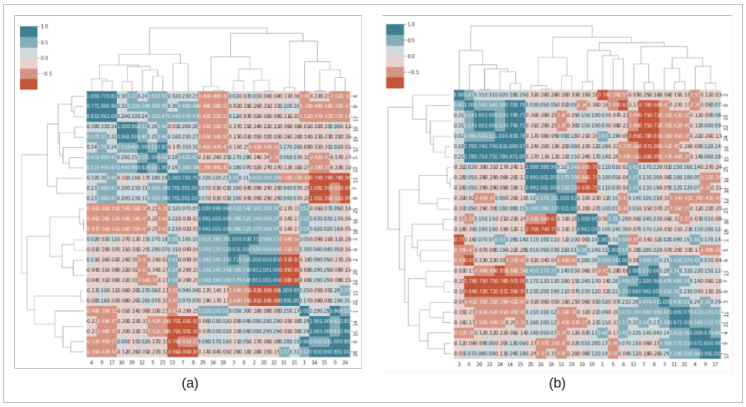

Figure 12: Correlation of multivariate time-series of original and synthetic BATADAL dataset

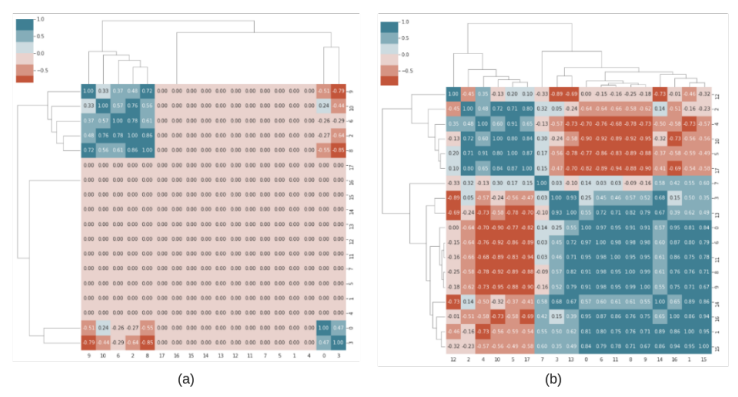

Figure 13: Correlation of multivariate time-series of original and synthetic SWaT dataset

6 Conclusion

In this project, we apply TimeGAN framework on Sine wave, BATADAL, and SWaT dataset for both univariate and multivariate time series synthetic data generation by preserving complex conditional temporal dependencies among the feature space. We observe that, by using jointly learned embedding and leveraging supervised loss contribution, the generated synthetic dataset captures conditional temporal dynamics. In future works, we will focus on spatio-dependency preservation, privacy preservation, and explanability for synthetic data generation for other cyber physical systems.

References

[1] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P Mathur. "WADI: a water distribution testbed for research in the design of secure cyber physical systems". In: *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*. 2017, pp. 25–28.

[2] Laura Aviñó, Matteo Ruffini, and Ricard Gavaldà. "Generating synthetic but plausible healthcare record datasets". In: *arXiv preprint arXiv:1807.01514* (2018).

[3] Graham Cormode et al. "Differentially private spatial decompositions". In: *2012 IEEE 28th International Conference on Data Engineering*. IEEE. 2012, pp. 20–31.

[4] Abhyuday Desai et al. "TimeVAE: A Variational Auto-Encoder for Multivariate Time Series Generation". In: *ArXiv* abs/2111.08095 (2021).

[5] Alessandro Erba et al. "Constrained Concealment Attacks against Reconstruction-based Anomaly Detectors in Industrial Control Systems". In: *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*. 2020. DOI: 10.1145/3427228.3427660.

[6] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. "Real-valued (medical) time series generation with recurrent conditional gans". In: *arXiv preprint arXiv:1706.02633* (2017).

[7] Jonathan Goh et al. "A dataset to support research in the design of secure water treatment systems". In: *International conference on critical information infrastructures security*. Springer. 2016, pp. 88–99.

[8] Steve T.K. Jan et al. "Throwing Darts in the Dark? Detecting Bots with Limited Data using Neural Data Augmentation". In: *2020 IEEE Symposium on Security and Privacy (SP)*. 2020, pp. 1190–1206. DOI: 10.1109/SP40000.2020.00079.

[9] Steve TK Jan et al. "Throwing darts in the dark? detecting bots with limited data using neural data augmentation". In: *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2020, pp. 1190–1206.

[10] Zinan Lin et al. "Using GANs for sharing networked time series data: Challenges, initial promise, and open questions". In: *Proceedings of the ACM Internet Measurement Conference*. 2020, pp. 464–483.

[11] Noseong Park et al. "Data synthesis based on generative adversarial networks". In: *arXiv preprint arXiv:1806.03384* (2018).

[12] MR Gauthama Raman, Wenjie Dong, and Aditya Mathur. "Deep autoencoders as anomaly detectors: Method and case study in a distributed water treatment plant". In: *Computers & Security* 99 (2020), p. 102055.

[13] Markus Ring et al. "Flow-based Network Traffic Generation using Generative Adversarial Networks". In: *Comput. Secur.* 82 (2019), pp. 156–172.

[14] Markus Ring et al. "Flow-based network traffic generation using generative adversarial networks". In: *Computers & Security* 82 (2019), pp. 156–172.

[15] Derek Snow. "MTSS-GAN: Multivariate Time Series Simulation Generative Adversarial Networks". In: 2020.

[16] Yi Sun, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. "Learning vine copula models for synthetic data generation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 5049–5057.

[17] Riccardo Taormina et al. "Battle of the attack detection algorithms: Disclosing cyber attacks on water distribution networks". In: *Journal of Water Resources Planning and Management* 144.8 (2018), p. 04018048.

[18] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[19] Lei Xu et al. "Modeling tabular data using conditional gan". In: *Advances in Neural Information Processing Systems* 32 (2019).

[20] Shengzhe Xu et al. "STAN: Synthetic Network Traffic Generation with Generative Neural Models". In: *International Workshop on Deployable Machine Learning for Security Defense*. Springer. 2021, pp. 3–29.

[21] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. "Time-series generative adversarial networks". In: *Advances in Neural Information Processing Systems* 32 (2019).

[22] Wen Zhou et al. "Attack sample generation algorithm based on data association group by GAN in industrial control dataset". In: *Computer Communications* 173 (2021), pp. 206–213.