ALMA MATER STUDIORUM · UNIVERSITY OF BOLOGNA
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**MASTER THESIS IN
ARTIFICIAL INTELLIGENCE**

CHALLENGING THE DYNAMICS OF TIME: GENERATE AND EVALUATE REAL-WORLD TIME SERIES TO ESTIMATE NOx EMISSIONS IN A TURBO MACHINE

CANDIDATE:
Anna Valanzano

SUPERVISOR:
Prof. Michele Lombardi

CO-SUPERVISORS:
Dr. Luca Strazzera
Dr. Alessandro Maggio

2nd Graduation Session
Academic Year 2022-2023

Contents

List of Figures

Acknowledgements

I would like to thank my relator Michele Lombardi and correlator Alessandro Maggio for their feedback and recommendations during the development of this thesis. I would also like to extend my thanks to all my colleagues at the University of Bologna for standing by my side throughout these two years of my Masters program.
I would also like to express my sincere gratitude to my manager Giacomo Veneri and tutor Luca Strazzera for giving me the opportunity to do this internship and for their patience, support and guidance during this period. To my colleagues of AI team, I am grateful for the enriching experiences we have shared, and I look forward to our continued journey together.
Last but certainly not least, I want to express my deepest appreciation to my parents and to my beloved sister, Miriam. Their unconditional love, encouragement, and unwavering belief in me have been my constant motivation.

Abstract

Estimating accurate NOx emissions is essential to monitor pollution and health condition of a turbo machine. We use a virtual sensor to correctly estimate the particle pollution value in real-time, leveraging modern machine learning approaches. It is well known that machine learning heavily relies on data, but real-world applications encounter data-related issues. In this work, we address the cited business use case scenario where limited data hamper an optimal regression quality. We investigate the application of time series generative models, with a particular emphasis on evaluating their performance using a comprehensive set of quantitative and qualitative metrics. We also conduct a critical analysis of the evaluation metrics commonly employed in the literature for validating and assessing the effectiveness of generative models. The analysis highlights the limitations of these metrics, as they do not take into account the temporal dependencies present in time series data and rely heavily on the specific implementation of the evaluation model. Finally, task-specific metrics are proposed to assess the effectiveness of generated data in supporting an industrial application. By delving into those pillars, this work aims to contribute to the advancement of knowledge on temporal synthetic data generation, showing how it can impact environmental care.

Chapter 1

Introduction

1.1 Business Use Case: turbo machines and gas turbines

This thesis represents the culmination of my internship experience at Baker Hughes, a cutting-edge energy technology firm specializing in the field of turbomachinery. Simply put, a turbomachine is a mechanical device that transfers energy between a fluid (such as gas or liquid) and a rotor. Baker Hughes is dedicated to producing turbine technologies for mechanical drive and power generation applications and we design a wide range of gas turbines for oil and gas and other industrial applications. A gas turbine is a type of internal combustion engine that converts the energy from the combustion of fuel into mechanical energy. Gas turbines typically consist of three main components: a compressor, a combustor, and a turbine. The compressor draws in and compresses air, which is then mixed with fuel in the combustor and ignited. The hot combustion gases expand through the turbine, causing it to rotate and generate mechanical power. Gas turbines rely on a multitude of sensors, numbering in the thousands, to enable robust monitoring and precise control of their operational parameters. Some common types of sensors found in gas turbines include temperature sensors, pressure sensors, flow sensors, vibration sensors, speed sensors, gas composition sensors.

1.2 The issue of NOx emissions: CEMS and PEMS

Numerous modeling techniques have been explored for virtual sensing, For instance, Korpela et al. [22] proposes a comparison study including linear (ARX and ARMAX) and nonlinear (NN and SVR) modelling approaches. Instead, Lv et al. [26] proposes a novel least squares support vector machine (LSSVM)-based ensemble learning paradigm to predict NOx emission of a coal-fired boiler using real operation data. Among these, NNs have emerged as a highly effective approach for predictive emission monitoring systems, demonstrating strong capabilities for estimating pollutants such as nitrogen oxides (NOx), as stated in Radl [28]. Recurrent NNs, which can capture dynamic temporal behavior, have been specifically recommended for estimating carbon dioxide (CO2) emissions in Ćirić et al. [39], demonstrating their efficiency for air quality estimation applications that require modeling time-series data. Long short-term memory (LSTM) neural networks (introduced in Hochreiter and Schmidhuber [17]) have emerged as a promising architecture for estimating industrial emissions, as demonstrated in Xie et al. [34], Yang et al. [36].

1.3 Our proposal: Time Series Generation

Accurate monitoring of NOx emissions is essential in our business case, and this poses a significant challenge in an industrial setting, where dealing with small datasets is very common. The performance of our initial regression model based on Long Short-Term Memory (LSTM) was suboptimal according to various regression metrics, as discussed in Section 4.1. After experimenting with different model architectures and hyperparameter configurations, we determined the underperformance was likely due to limitations in the training data rather than the model itself. Specifically, we hypothesize that the available data may have been insufficient in quantity, did not encompass all relevant operating conditions of the machine. In general, large datasets are an important requirement to achieve high performance, accuracy, and generalization for any machine learning task, such as prediction or anomaly detection. While further data collection focusing on underrepresented conditions could help address the potential data issues limiting model performance, obtaining additional real-world data is currently infeasible within the specific context of this project due to constraints in cost and time. Therefore, our proposed solution to get more data is to generate synthetic data, that, according to our expectation, should increase the generalization ability of trained models by reducing overfitting. In this work, we tested three generative approaches, Gaussian Mixture modeling (GM)

(Reynolds [29]), Variational Autoencoders (VAEs) (Kingma and Welling [21]), Generative Adversarial Networks (GANs) (Goodfellow et al. [13]), and we compared them to a simple baseline model, which creates synthetic data adding random noise to the mean sensor value. We conduct experiments on two datasets - one commonly used in previous research, as described in Section 3.1, and another from our industrial application, which we also characterize in Section 3.1. We further investigate the metrics generally applied in research for evaluating time series generation (as surveyed in Yadav et al. [35], Arnout et al. [1]), with a focus on their explanatory power regarding the generation model. Specifically, for instance, we examine whether these metrics can elucidate the model's capacity to produce coherent samples over time or generate samples within the appropriate value range. Moreover, we demonstrate that these metrics can be inadequate for evaluating the overall performance of our models, as our investigation revealed that some of these metrics, such as visual and statistical metrics, do not take into account the temporal dependencies present in time series data. On the other hand, other metrics, based on the predictions of classification and regression models, may rely on the specific implementation of the evaluation model. To address these limitations, we introduced task-specific metrics that assess whether the generated data can effectively support our industrial task and provide insights into how time series generation models can meet the needs of real-world applications.
Thus, the main contributions of this work are two-fold. First, it demonstrates the potential of time series generation models for practical industrial applications. Second, it highlights the importance of using appropriate evaluation metrics when assessing time series models. Determining suitable evaluation metrics poses intrinsic challenges due to the temporal nature of time series data. Past literature has not extensively examined evaluation methodologies for these models. The inherent difficulties in evaluating time series generation models are discussed in more detail in the following paragraph, as evaluating them presents challenges given their characteristics of time series data.

1.4 Challenges

As stated in Brophy et al. [2], unlike static data, as images for instance, research on generative models for time series is still in an embryonic stage and dealing with time series implies the additional issue of properly modeling the variables temporal evolution under observation. To model successfully time-series data means that a model must, not only capture the datasets features distributions within each time-point but also, it should be able to capture the complex dynamics of those features across time. We must not forget also that each time sequence as a variable length associated. Capturing the intricate temporal dependencies and ensuring scalability and efficiency are crucial challenges. Furthermore, another big deal is the evaluation of the fake samples. While visual inspection can often suffice for static data like images, it proves inadequate for time series analysis. Even in the case of univariate time series, it is difficult to determine if the generated samples accurately represent the real data distribution by solely examining the plots and the common metrics. This difficulty further intensifies when dealing with multivariate time series.

1.4.1 Goals and Motivation: Addressing Privacy Risks

In addition to the challenges posed by privacy concerns and the scarcity of large and balanced time series datasets, there is another important motivation driving the use of generative models for time series: they might mitigate privacy risks associated with sensitive time series data. Accessing datasets in the time series domain is often challenging due to privacy concerns and the difficulty of obtaining sufficiently large and balanced datasets, as highlighted in Brophy et al. [2]. To address these challenges, synthetic data generation has gained significant attention as it offers an effective solution for preserving data privacy while maintaining key statistical properties. In this context, time series generative models play a crucial role by generating differentially private datasets, thereby eliminating the risk of linkage between the source and generated data. A wide range of methods have been used to asses the privacy risk associated with synthetic data. Choi et al. [6] performed tests for presence disclosure and attribute disclosure. In contrast, other studies (Choi et al. [6], Esteban et al. [11]) utilized a three-sample test, comparing the training, test, and synthetic data, to identify potential overfitting of the synthetic data to the training data.

1.5 Outline

Chapter 2 presents the theoretical background required to understand and conduct this thesis, including a discussion of generative models and evaluation metrics. Chapter 3 describes the datasets and implementation details of the model used in this study. Chapter 4 presents and discusses the results obtained in this thesis. Finally, Chapter 5 concludes the research by summarizing the key findings and offering suggestions for further improvements and future work.

Chapter 2

Background

2.1 Time Series Generation: problem formulation

The data we are analyzing is collected from multiple sensors attached to machines on a manufacturing production line. These sensors capture readings over time from various variables such as temperature, vibration, pressure and speed. As the measurements are recorded sequentially at intervals during ongoing machine operations, we have a multivariate time series dataset. A multivariate time series is a time series that consists of multiple variables observed simultaneously over discrete points in time. In our case, each time point consists of a vector of sensor values - one for each measured variable.
Let's assume we have a set of real time series data $X = \{X_i\}_{i \in I}$. Each X_i is a multivariate time series $X_i = \{x_{i,j,t}\}_{j \in J, t \in T}$ where i indexes the individual time series, j indexes the variable measured and t indexes the time step. Each $x_{i,j,t}$ is a bi-dimensional vector in $\mathbb{R}^{DxT}$ with $D \in \mathbb{N}$, $T \in \mathbb{R}$.
Assuming that the real data follow from an underlying distribution p_{real}, the generative model task is to learn a probability distribution p_{synth} which approximates p_{real}. Then, p_{synth} is used to generate a new dataset of fake time series, namely $S = \{S_i\}_{i \in I}$ with $S_i = \{s_{i,j,t}\}_{j \in J, t \in T}$.

2.2 A Brief Background of Generative Models

Historically, time series generation has roots in regression and initially focused on forecasting individual time steps rather than generating entire sequences. Autoregressive

(AR) models are commonly used for time series forecasting as they consider a time series to be correlated with its own lagged values. Specifically, simple AR models assume the output variable is linearly regressed on its own previous values. While AR models were widely adopted in the past, they are inherently deterministic in that future states are calculated deterministically based on past values, without accounting for any randomness. More precisely, an AR model predicts the current value based on a linear combination of the previous n timesteps' values, plus an error term. This error term represents the unpredictable or stochastic part of the time series and allows the model to incorporate some randomness. However, the random variations are assumed to be independent and identically distributed over time.

In the context of time series generation, two main families of approaches exist: statistical methods and deep learning methods (Iwana and Uchida [18]). Statistical methods explicitly model the probability distribution and dynamics of time series, often making strong assumptions about their conditional distribution. On the other hand, deep learning methods, approximate the probability distribution without relying on strong prior assumptions (Gatta et al. [12]). Neural networks, in general, have the advantage of being universal approximators and can learn various types of distributions, as stated in Lu and Lu [25]. For this reason, neural networks serve as a strong foundation for building more advanced generative models.

Various deep learning generative models have been used for synthetic data generation, from variational autoencoders (VAEs) and recurrent neural network (RNN) variants to GANs, all of which have their pros and cons (Brophy et al. [3]).

In this study, we explore three models for time series generation: Gaussian Mixture (statistical model), TimeGAN, and TimeVAE (deep learning models). Brief descriptions of these models are provided in Subsections 2.2.1, 2.2.2, and 2.2.3.

2.2.1 Gaussian Mixture Models

Gaussian Mixture Models (GMMs) are statistical models that represents a dataset as a combination of Gaussian distributions and allow to generate synthetic data by sampling from the learned distribution. This is achieved by randomly selecting a component based on the mixing weights and sampling from the corresponding Gaussian distribution. It is important to note that GMMs make the assumption that a mixture of Gaussians can effectively represent the underlying data and this assumption may not hold true for many real-world datasets, as they often have more complex distributions. The choice of the number of Gaussian components is also an important hyperparameter that can greatly affect model performance if not selected appropriately based on the data. Typically, the optimal number of components is determined through empirical evaluation of model likelihood on a validation set for different numbers of Gaussians.

2.2.2 Generative Adversarial Networks

GANs are networks consisting of a generator and a discriminator which compete with each other in a min-max game in order to learn. The discriminator tries to separate real data from fake data and the generator tries to fool the discriminator by creating increasingly realistic synthetic data. By competing against each other they both try to improve and this can result in an optimised generator that can create very realistic looking synthetic data. The discriminator has to provide the probability that a real/fake sample belongs to the real dataset, the generator really learns an approximation of the original distribution (Goodfellow et al. [13]).

There exist numerous variations of GANs, which vary in terms of network structures, loss functions, and specific design choices. Among these frameworks, the authors of Esteban et al. [10] propose a Recurrent GAN (RGAN) and Recurrent Conditional GAN (RCGAN) to produce realistic real-valued multi-dimensional time series, with an emphasis on their application to medical data. RGANs make use of recurrent neural networks in the generator and the discriminator. In the case of RCGANs, both of these RNNs are conditioned on auxiliary information. Authors of Mogren [27] propose C-RNN-GAN, a generative adversarial model that works on continuous sequential data, and apply it by training it on a collection of classical music. It is noteworthy to citet also Srinivasan and Knottenbelt [30], a new kind of time series-based GAN, namely the Time-series Transformer GAN, which combine adversarial training with the cutting-edge idea of transformer networks. Instead, Sumiya et al. [31] proposes the Noise Reduction GAN, focused on noise reduction for medical signals, a CNN GAN with a penalization term in the Generator loss which forces the generated sample to be closer to the original data than the standard GAN. Another work is Hazra and Byun [16], the Synthetic biomedical Signals GAN, which combines CNN in the Discriminator and RNN in the Generator. Another inter-

esting work is Jeha et al. [19] which proposes PSA-GAN, a GAN-based approach that exploits self-attention mechanisms to identify temporal patterns in the series.
For all of the benefits that come with GANs, they are not without their own downsides, as described in Brophy et al. [2]. One of the significant challenges of GANs lies in their inherent instability, which makes them difficult to train. GAN models suffer from issues such as non-convergence, diminishing/vanishing gradients, and mode collapse. A non-converging model does not stabilize and continuously oscillates, causing it to diverge. Diminishing gradients prevent the generator from learning anything, as the discriminator becomes too successful. Mode collapse is when the generator collapses, producing only uniform samples with little to no variety, as described in Brophy et al. [3]

2.2.2.1 TimeGAN

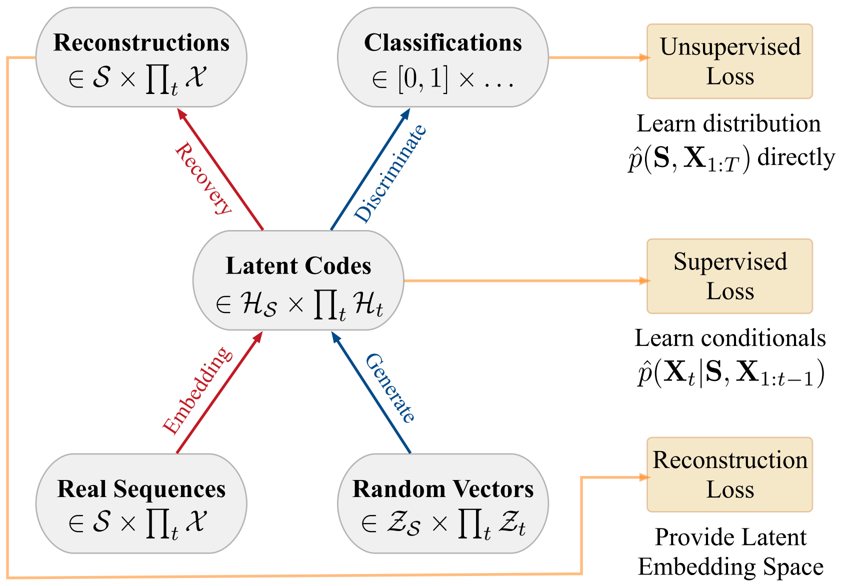

Reconstructions
$\in \mathcal{S} \times \prod_t \mathcal{X}$
Classifications
$\in [0,1] \times \ldots$
Unsupervised Loss
Learn distribution $\hat{p}(\mathbf{S}, \mathbf{X}_{1:T})$ directly
Recovery
Discriminate
Latent Codes
$\in \mathcal{H}_\mathcal{S} \times \prod_t \mathcal{H}_t$
Supervised Loss
Learn conditionals $\hat{p}(\mathbf{X}_t|\mathbf{S}, \mathbf{X}_{1:t-1})$
Embedding
Generate
Real Sequences
$\in \mathcal{S} \times \prod_t \mathcal{X}$
Random Vectors
$\in \mathcal{Z}_\mathcal{S} \times \prod_t \mathcal{Z}_t$
Reconstruction Loss
Provide Latent Embedding Space

Figure 2.1: Block diagram of component functions of TimeGAN model. Reprinted from Yoon et al.

A good generative model for time-series data should preserve temporal dynamics, in the sense that new sequences respect the original relationships between variables across time. TimeGAN model (Yoon et al. [38]) was proposed as a GAN based framework that is able to generate realistic time-series data. In addition to the unsupervised adversarial loss on both real and synthetic sequences, it introduces a supervised loss using the original data as supervision, thereby explicitly encouraging the model to capture the stepwise conditional distributions in the data. Moreover, it introduces an embedding network to provide a reversible mapping between features and latent representations, thereby reducing the high-dimensionality of the adversarial learning space. Figure 2.1 depicts the key components of the TimeGAN model. It shows real time series sequences used to train the model, random vectors that serve as input to the generator, classifications produced by the discriminator, and reconstructed generator outputs. The figure also illustrates the losses related to the inputs, showing the unsupervised loss on discriminator classifications, supervised loss on embedded data, and reconstruction loss on the reconstructed outputs.

The GAN has a three phase training process. First, the autoencoder, comprising the Encoder and Decoder, is pre-trained on the provided sequential data to optimize reconstruction. Next, the Generator and Discriminator are trained together to capture the temporal behavior and patterns in the historical data, with the Generator trying to fool the Discriminator. Finally, all four components - the autoencoder, Generator, and Discriminator - are jointly trained.

2.2.3 Variational Autoencoder

A Variational Autoencoder (VAE) (Kingma and Welling [20]) learns to encode an input data into a lower-dimensional latent representation. Unlike traditional auto-encoders, the encoder in a VAE outputs the distribution of the embedding instead of a point estimate. Let us consider some dataset $X = \{x^{(i)}\}_{i=1}^{N}$ consisting of N i.i.d. samples of some continuous or discrete variable x. The VAE presented by Kingma and Welling [21] assumes that the data are generated by some random process, involving an unobserved continuous random variable z. The process consists of two steps: (1) a value $z^{(i)}$ is generated from some prior distribution $p_\theta(z)$, (2) a value $x^{(i)}$ is generated from some conditional distribution $p_\theta(x|z)$. The encoder models the probabilistic posterior distribution $q_\theta(z|x)$, while the decoder models the conditional likelihood $p_\theta(x|z)$. The VAE is trained to minimize the reconstruction error between the input data and the reconstructed output while also maximizing the likelihood of the learned probability distribution. As stated in Brophy et al. [3], VAEs use learned approximate inference to produce synthetic samples efficiently. An inference problem is simply using the value of some variables or probability distributions to predict other values or probability distributions.

Approximate inference is when we seek to approximate a true distribution, say $p(y|x)$, by seeking an approximate distribution $q(y|x)$. However, this network approximation conducted by VAEs means that their generated data quality can be degraded compared to samples generated by GANs.

2.2.3.1 TimeVAE

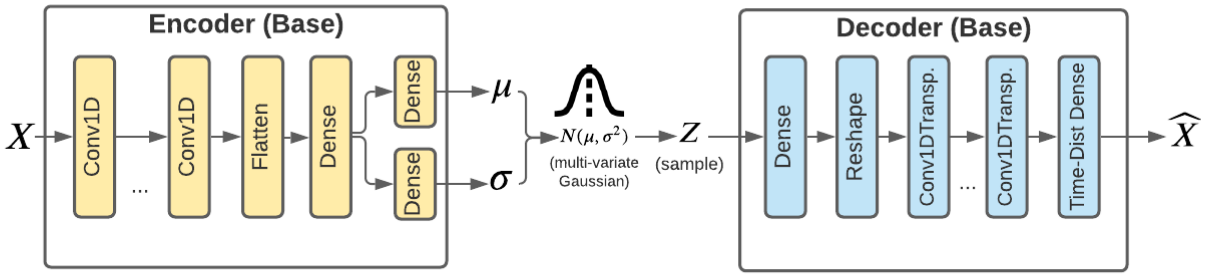


Figure 2.2: Block diagram of component functions of TimeVAE model. Reprinted from Desai et al.

Figure 2.2 shows a block diagram of component functions of TimeVAE model (Desai et al.). The encoder takes input sequences X and applies a series of convolutional layers, utilizing ReLU activation as used by the authors of TimeVAE, to extract features and reduce dimensionality. Next, the data are flattened before passing through a fully-connected (dense) linear layer. The outputs are fed into two dense layers the produce two vectors of means and variances μ and σ to parameterize the latent space. If m is the number of chosen latent dimensions, representing dimensions of the multi-variate Gaussian, then this last dense layer has $2m$ number of neurons. The size of latent space m is a key model hyper-parameter.

In the generation phase, a vector Z of length m is sampled from a learned multivariate Gaussian distribution and then fed into the decoder, which processes it through a fully connected linear layer. Then the data are reshaped into 3-dimensional array before passing through a series of transposed convolutional layers with ReLU activation. Finally, the data is forwarded through a time-distributed fully connected layer with dimensions designed to match the shape of the original signal X. The purpose of the time-distributed layer is to independently apply a dense layer to each time step of the output generated by the convolutional layers. This mechanism ensures that the previous layer returns sequences, preserving the temporal information of the data.

2.3 Evaluation

It is important to note that within the research community, there is currently no consensus on standardized evaluation metrics for assessing the quality of generated time series data. To assess the effectiveness of our proposed models in generating synthetic time series data, we employed various techniques commonly used in the literature (see Yadav et al. [35], Arnout et al. [1]). These techniques involve both standard metrics that are used for other tasks, as well as ad hoc metrics that have been specifically developed for the time series generation task.

2.3.1 Standard metrics

To measure the dissimilarity between the probability distribution of generated data and that of the real data, we employ a statistical measure known as the Maximum Mean Discrepancy (MMD). This kernel-based test, introduced in the referenced paper Gretton et al. [15] can be used as a loss/cost function in generative models as shown in Dziugaite et al. [9] For a comprehensive definition of this metric, we refer the reader to the original work in Gretton et al. [15] In addition, visual evaluation techniques were also employed. Given that we are dealing with multivariate time series data, we utilized Principal Component Analysis (PCA) (Bryant and Yarnold [4]) and t-Distributed Stochastic Neighbor Embedding (t-SNE) (van der Maaten and Hinton [33]) to reduce the data into two dimensions. This reduction allows us to plot the data in a two-dimensional space, facilitating visual exploration and interpretation. Unlike the MMD, which provides a numeric evaluation of similarity, these techniques allow for a more intuitive understanding of the similarity between the two distributions.

2.3.2 Ad-hoc metrics

The work by Esteban et al. [10] introduced two evaluation techniques, namely Train on Synthetic Test on Real (TSTR) and Train on Real Test on Synthetic (TRTS). These approaches involve training a regression model using synthetic data generated by the model and evaluating its performance on real data, as well as training the same regression model on real data and assessing its performance on synthetic data. The authors in

Yoon et al. [38] devised a discriminative score (DS) by training a post-hoc time-series classification model to differentiate between sequences from the original and generated datasets. Additionally, they computed a predictive score (PS) by training a post-hoc sequence-prediction model on the generated data and evaluating its ability to predict next-step temporal vectors for each input sequence in the real data. It is important to note that lower values of these scores (TRTS, TSTR, DS, and PS) indicate a higher level of similarity between real and synthetic data, as they are directly related to the amount of error the classification/regression models produce.

2.3.3 Regression metrics

We use four metrics to evaluate the performance of regression models: Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), 90th percentile error value, and Maximum error value (which we will refer to as Max Error). The MSE measures the average of the squared errors between predicted and actual values. If Y is an n-dimensional vector of observed values and $\hat{Y}$ is an n-dimensional vector of predicted values, then the Mean Squared Error (MSE) is computed as:

$$\text{MSE}(Y, \hat{Y}) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

The MAPE measures the average size of errors as a percentage, showing the average percentage difference between forecasts and actual outcomes. The MAPE is computed as:

$$\text{MAPE}(Y, \hat{Y}) = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right|$$

The 90th percentile error value (also called p90) represents the error threshold below which 90% of predictions fall. This provides insight into the tail of the error distribution. The Maximum error value is the single largest deviation observed across all predictions. While not a robust standalone metric, it offers context regarding the scale of potential outliers. Collectively, these metrics evaluate the model from multiple angles - how close predictions align with actuals on average (MSE, MAPE), where most errors tend to lie (p90), and the degree of potential outliers (maximum error). Lower values across all metrics demonstrate enhanced regression performance.

Chapter 3

Methodology

In the upcoming paragraphs, we discuss the datasets used for our experiments and the preprocessing phase in Sections 3.1, 3.2. Furthermore, we provide insights into the implementation details of both the regression model and generative models, as outlined in Section 3.3. All the experiments have been done with a GPU Tesla V100-SXM2-32GB.

3.1 Datasets description

We conduct experiments on two datasets to evaluate the effectiveness of our proposed model. The first dataset, the UCI Appliances Energy Prediction Dataset (reference at Candanedo [5]), which we will refer to as the "Energy Dataset", is a well-established resource for constructing regression models that accurately predict the energy usage of appliances in low-energy buildings. It has gained widespread recognition as a benchmark dataset for conducting time series generation experiments. The dataset bears significant resemblance to our own case, as it provides comprehensive measurements of essential variables like temperatures and pressures.

The second dataset, the "NOx dataset", was obtained from an industrial setting and contains data from 35 sensors which are monitoring various machine-specific information, similar to the features in the Energy dataset, but with the addition of emissions values. Nevertheless, due to intellectual property concerns, we cannot provide further technical details on our industrial dataset, whose features will be named $x_0, ..., x_{34}$, and all values have been appropriately normalized. The dataset contains $300,000$ sensor value instances collected over two years from an operating gas turbine.

Feature	Mean	Standard Deviation
x_1	0.590	0.077
x_6	0.512	0.105
x_8	0.403	0.154
x_{10}	0.194	0.085

Table 3.1: Mean and Standard Deviation of Normalized Features x_1, x_6, x_8, x_{10} in the NOx Dataset.

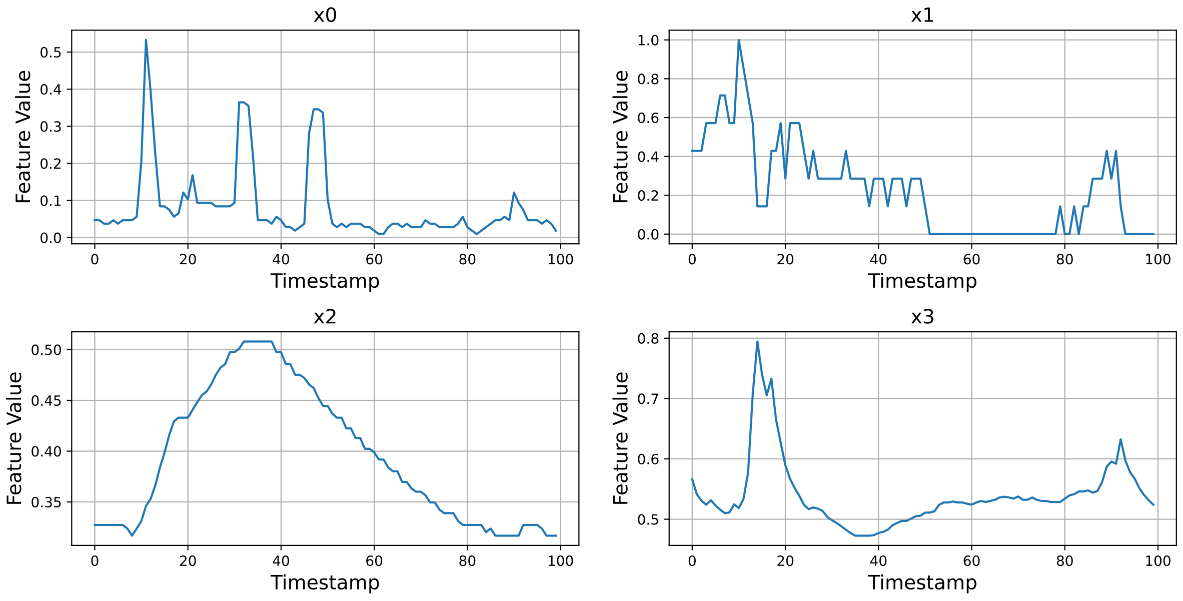


Figure 3.1: Temporal Variation of the First Four Sensors in the Energy Dataset over the Previous 100 Time Steps (Plotted on Normalized Data).

3.2 Preprocessing

For the NOx dataset, we resample the samples to have a uniform one-minute interval between recordings. To identify the most influential features for our model, we employ the Boruta algorithm, which facilitates feature selection. To gain a comprehensive understanding of this algorithm, we refer interested readers to the original work in Kursa and Rudnicki [23]. For all the features in the original dataset, it creates random copies of them (called shadow features) and train regression models based on this extended dataset. To understand the importance of a feature, the algorithm compares it to all the generated shadow features. Only features that are statistically more important than these synthetic features are retained as they contribute more to model performance. In our study, we retained 16 out of the original 35 sensors in the dataset. We do not perform any of these preprocessing operations on the Energy dataset, as it is a benchmark

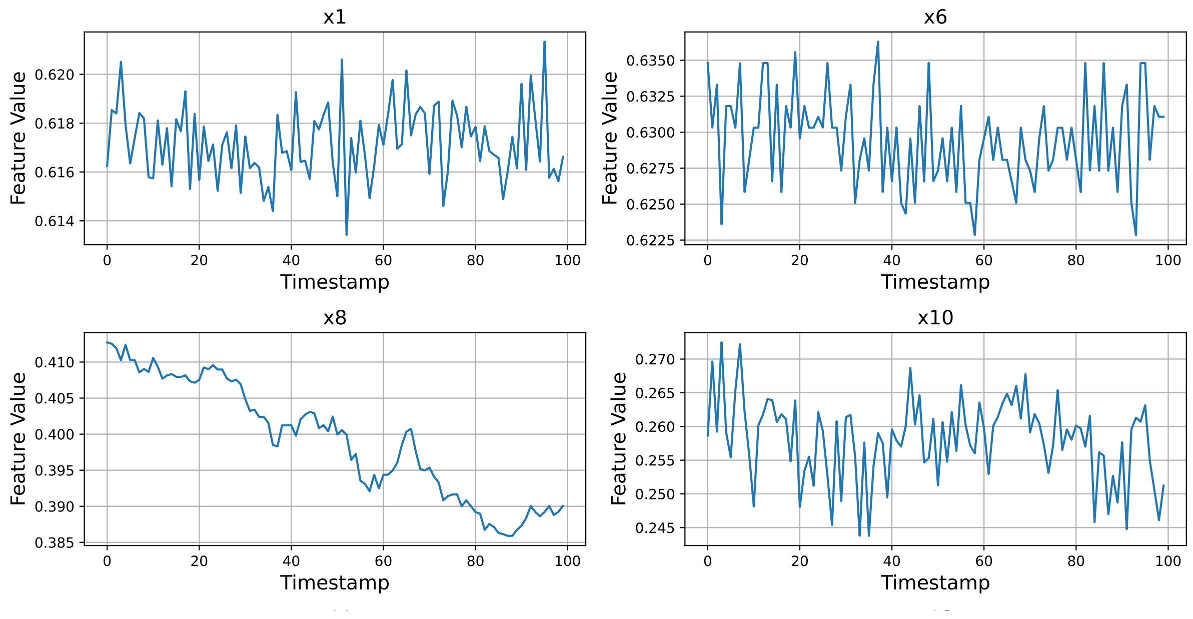


Figure 3.2: Temporal Variation of the First Four Sensors in the NOx Dataset over the Previous 100 Time Steps (Plotted on Normalized Data).

literature dataset that was already prepared and suitable for our experiments.

The two-dimensional datasets are reshaped into three-dimensional data by creating overlapping windows of n timestamps. The value of n serves as a hyper-parameter that necessitates tuning in order to identify the most suitable value. In our specific case, we have determined that a window size of 24 timestamps provides the desired outcome.

The hold out approach was used to split the data into 80% train and 20% test sets. The training and test data undergo separate scaling processes using a MinMaxScaler to transform the data to a range of 0 to 1.

3.3 Models: implementation details

We explore four different models for generating synthetic time series data: Gaussian mixture modeling, GANs, VAEs and a simple baseline model. In our exploration, we specifically consider two prominent state-of-the-art generative models for time series data, namely TimeVAE (Desai et al. [7]) and TimeGAN (Yoon et al. [38]). We utilized the architecture provided by the original authors of TimeGAN, which can be found in Yoon [37], to develop our implementation as a starting point for our implementation, making necessary adaptations and improvements to enhance its functionality and performance. We provide an overview of the implementation details for the models considered in sections 3.3.1 to 3.3.4.

3.3.1 Baseline

The simple baseline model provides a comparison point for evaluating the performance of the more complex models. It generates synthetic time series data by adding a random noise term to the mean of the sensor value in the training data. The noise term is added with a probability of 10%, and has a value between 0 and the standard deviation of the sensor measurement for all the training timestamps.

The introduction of this baseline model is necessary due to the prevalent practice in the literature of comparing different complex architectures without a reference point. For instance, studies like Yoon et al. [38] compare their architecture with RCGAN (Esteban et al. [10]), C-RNN-GAN (Mogren [27]), RNNs trained with teacher-forcing (T-Forcing) (Graves [14]), professor-forcing (Lamb et al. [24]), WaveNet (van den Oord et al. [32]), and its GAN counterpart WaveGAN (Donahue et al. [8]). Similarly, the authors of TimeVAE (Desai et al. [7]) compare their architecture against TimeGAN, RNNs trained with teacher-forcing (T-Forcing), and RCGAN. However, the absence of a baseline model in these comparisons leaves the evaluation incomplete.

Regarding the definition of the baseline model, we establish it in a specific manner to ensure that it generates examples that conform to the desired distribution but do not exhibit time correlation. If the baseline model performs favorably based on the evaluation metrics, it raises concerns that the metrics may not be effectively capturing the desired characteristics of the time series data. In such a scenario, it indicates a potential limitation or flaw in the selected metrics, as they are unable to distinguish between models with and without time correlation accurately. Conversely, if the metrics indicate poor performance for the baseline model, it suggests that the evaluation measures are successful in identifying the absence of time correlation. This outcome instills confidence in the metrics' ability to evaluate the desired characteristics and differentiate between models that exhibit time correlation and those that do not.

3.3.2 Gaussian Mixture

To begin, the model employs Principal Component Analysis (PCA) to reduce the dimensionality of the training data. This process involves representing the data using the top two principal components. To determine the optimal number of Gaussian to represent data, a grid search is performed, exploring different configurations. Once the ideal number is identified, the model proceeds to train the Gaussian Mixture Model (GMM) using the PCA data, specifically incorporating 30 distributions. The implementation of the model leverages the Scikit-learn library, specifically the GaussianMixture class, and is fitted with the default configuration of the hyperparameters. For more information, you can refer to the documentation provided by the Scikit-learn library. The trained GMM is then used to randomly generate new synthetic sample points that have similar properties and cluster patterns to the original training data.

3.3.3 TimeVAE

The Variational Autoencoder (VAE) minimizes the Evidence Lower Bound (ELBO) loss function, which comprises two components. First, it includes the negative log-likelihood of the data given the latent variable z, which is sampled from the approximate posterior distribution $q_\theta(z|x)$ (explained in Section 2.2.3). Second, it incorporates the Kullback-Leibler (KL) Divergence between the encoded latent space distribution and the prior distribution $p_\theta(x|z)$ (described in Section 2.2.3). The negative log-likelihood term measures the reconstruction error between the input data and the reconstructed output, while the KL-Divergence term encourages the learned latent representation to be close to the prior distribution. In summary, the VAE is trained to minimize the reconstruction error and maximize the likelihood of the learned probability distribution by optimizing the ELBO loss function.

A grid search was conducted to identify the optimal hyperparameters for implementing the TimeVAE model. The hyperparameters tuned include latent space dimension, batch size, early stop patience, number of training epochs, and the reconstruction weight hyperparameter. The reconstruction weight controls the weighting of the reconstruction loss in the ELBO loss function. As explained previously, the ELBO loss is the sum of the reconstruction loss and the divergence loss. By tuning the reconstruction weight hyperparameter, we can optimize the balance between reconstructing the input data and learning the latent space regularization in the TimeVAE model. The values of the hyperparameter that resulted in the best TimeVAE model performance set are reported in Table 3.2.

Hyperparameter	Value
Latent dimension	8
Reconstruction weight	3.0
Number of training epochs	1000
Early stopping patience	10
Batch size	256

Table 3.2: Hyperparameters of the TimeVAE model.

3.3.4 TimeGAN

The Generator, Discriminator, Encoder and Decoder components are implemented as deep recurrent neural networks with multiple LSTM layers to model temporal behavior. Specifically, each network consist of 3 LSTM layers followed by a dense layer. In the Generator, the final LSTM layer is connected to a dense sigmoid output layer that generates the synthetic time-series data. In the Discriminator, the last LSTM layer outputs to a dense sigmoid layer to predict the probability that the input is real versus fake. In the Encoder, the final LSTM layer is connected to a dense sigmoid layer that condenses the temporal information into a fixed-length embedding vector for each time-series. The length of this embedding vector is a hyperparameter, defaulted to 24. The Decoder mirrors the Encoder with 3 LSTM layers to reconstruct the original time-series from the embedding. Its final layer is a dense output layer with linear activation to recover the original input dimensions.
A grid search was conducted to identify the optimal hyperparameters for training the TimeGAN model. The hyperparameters tuned include the hidden dimension size of the RNN layers, batch size, number of joint training epochs, and the optimizer. The combination of hyperparameters that resulted in the best TimeGAN performance are reported in Table 3.3.

Hyperparameter	Value
Hidden dimension	24
Gamma	1
Batch size	256
Epochs (joint training)	1000
Optimizer	Adam

Table 3.3: Hyperparameters of the TimeGAN model.

Hyperparameter	Value
Learning rate	0.0001
Optimizer	Adam
Loss	Mean Square Error

Table 3.4: Hyperparameters of the LSTM Model.

3.3.5 LSTM Regression model

We develop virtual sensor aims to predict NOx emissions through a regression model based on a LSTM model [17]. The input to the model is a sequence of n data points, each data point consists of more features. The model architecture is described in Figure 3.3. The LSTM model consists of three LSTM layers followed by one dense layer. The

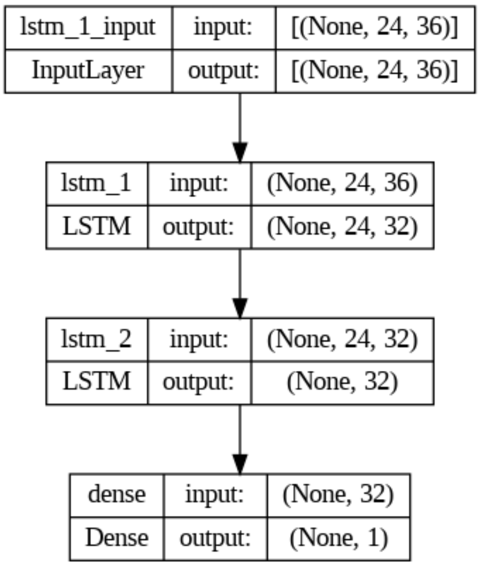

lstm_1_input | input: | [(None, 24, 36)]
InputLayer | output: | [(None, 24, 36)]
lstm_1 | input: | (None, 24, 36)
LSTM | output: | (None, 24, 32)
lstm_2 | input: | (None, 24, 32)
LSTM | output: | (None, 32)
dense | input: | (None, 32)
Dense | output: | (None, 1)

Figure 3.3: Block diagram of LSTM model

hyperparameters of the LSTM regression model are reported in Table 3.4.

3.4 Metrics: implementation details

MMD To compute MMD between the synthetic samples X and the real samples Y, we first calculate the kernel matrices K_{XX}, K_{YY}, K_{XY}, for the synthetic-synthetic, real-real, and synthetic-real sample pairs using a radial basis function (RBF) kernel. This is done with the pairwise kernels function from Scikit-learn library. Given the means of these kernel matrices - μ_{XX} for K_{XX}, μ_{XY} for K_{XY}, and μ_{YY} for K_{YY} - the MMD is computed as $\mu_{XX} - 2 * \mu_{XY} + \mu_{YY}$.

DS-PS DS and PS metrics were introduced by the authors of the TimeGAN architecture Yoon et al. [38], who also provided a public implementation. However, due to compatibility issues with previous versions of the TensorFlow libraries, we reengineered the models, striving to adhere as closely as possible to the original designs.
The DS is calculated using a RNN trained to distinguish between real and generated data. The model consists of a Gated Recurrent Unit (GRU) network with a fully connected layer for binary classification, where the two classes in consideration are real data and non-real data. The DS metric, as defined by the authors, is computed by subtracting 0.5 from the model's accuracy. This computation aligns with the official implementation of the TimeGAN model Yoon [37], where the DS metric is utilized.
The PS measures one-step ahead prediction performance. The model is trained on the generated data to predict the next time step of the target feature given n-1 input feature timestamps, and then evaluated on real data. The PS is the Mean Absolute Error between the predicted and actual target values, with lower values indicating better predictions.

TRTS-TSTR The generated samples are used to train and test an external regression model for predicting for predicting the specified target features (i.e. nox emission for nox and temperature value for energy). The model has the same architecture and hyperparameters as the model described in Figure 3.3 and Table 3.4 in Subsection 3.3.5.

Chapter 4

Results

We present the results of our analysis of nitrogen oxides (NOx) levels using regression in Section 4.1. Next, in Sections 4.2 and 4.3, we evaluate our generative pipelines using two datasets: the Energy Dataset is covered in Section 4.2, while the NOx Dataset is covered in Section 4.3. Finally, in Section 4.5 we assess the generated data through task-related metrics, specifically focused on its ability to perform downstream NOx-related tasks.

4.1 Results of the regression model

We show the regression model's performance on predicting NOx emission. Table 4.1 shows the evaluation metrics, while Figure 4.1 visualizes predictions over the first 100 test time steps. As seen in Figure 4.1, the model (orange line) captures the overall trend in the ground truth NOx emission (blue line), though it struggles with some of the minor fluctuations. Table 4.1 reports the performance metrics from attempting NOx regression

Performance Metric	**Value**
Mean Squared Error (MSE)	0.018
Mean Absolute Percentage Error (MAPE)	0.547
90th percentile (p90)	0.198
Max Error	0.605

Table 4.1: Mean metrics Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE) for NOx regression on the original scaled dataset over three attempts.

on the original scaled dataset over three tries. The resulting metrics are generally satisfying as the average error, with an MSE of 0.02 and MAPE of 0.55, is well below requirements. However, the maximum error of 0.60 and P90 percentile error of 0.20 indicate outliers account for more variation than desired.

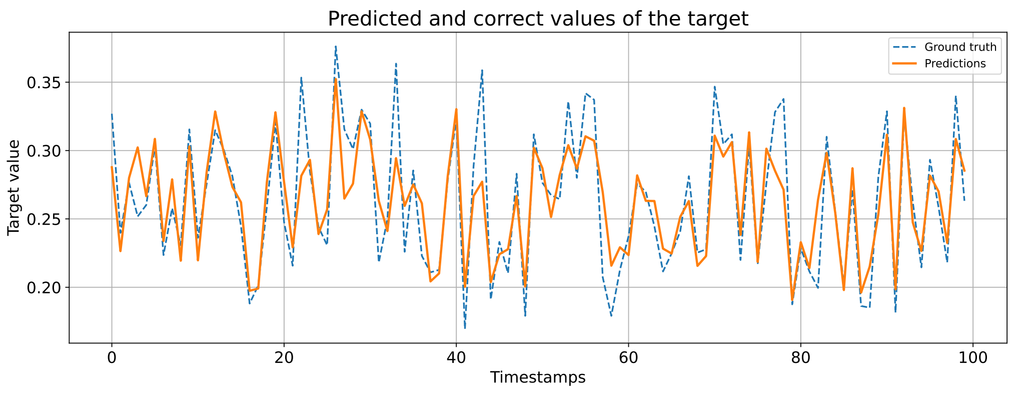


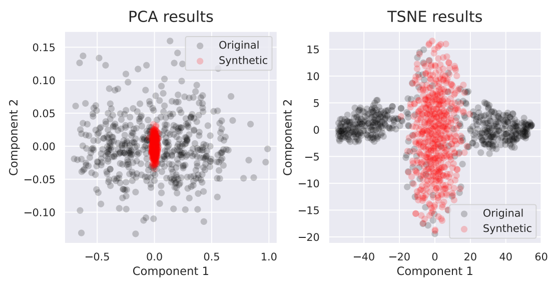
Figure 4.1: Example of Predictions of Scaled NOx Emissions over the First 100 Test Time Steps.

4.2 Results on the Energy Dataset

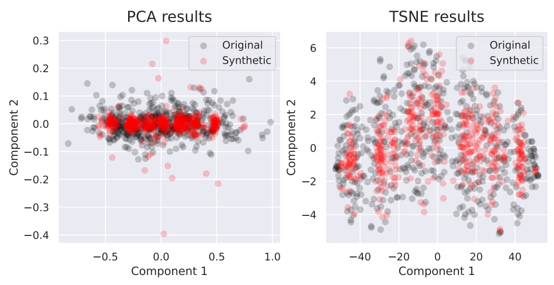
To compare the performance of the evaluated models with the results reported in literature, we show the selected evaluation metrics computed on the Energy Dataset. It is important to note that these results may be difficult to compare across studies, as the values of the DS, PS, and TSTR/TRTS metrics may depend on the specific definition of the model and the training procedure used. Therefore, our findings should be interpreted in the context of our specific model definitions and training procedures, and may not be directly comparable to the results reported in other studies.
We present the findings of our qualitative evaluation in Subsection 4.2.1 and the results of our quantitative evaluation of the models in Subsection 4.2.2.

4.2.1 Qualitative evaluation

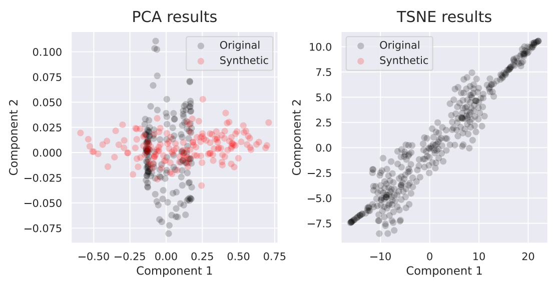
Figure 4.2 presents t-SNE and PCA visualizations for both real and synthetic data generated by the Baseline (top left), GM (top right), TimeVAE (bottom left), and TimeGAN (bottom right) models, specifically applied to the Energy Dataset. In the visualizations, the color red represents the original data, while black represents the synthetic data. As shown in Figure 4.2, drawing definitive conclusions based solely on visual inspection can be challenging. Figure 4.2a indicates that the baseline model can only generate values



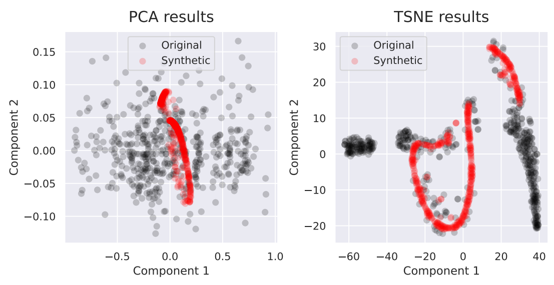
(a). Baseline model



(b). GM model



(c). VAE model



(d). GAN model

Figure 4.2: t-SNE and PCA visualization for the Baseline (top left), GM (top right), TimeVAE (bottom left), and TimeGAN (bottom right) models for Energy Dataset. Red denotes original data, and black denotes synthetic.

within a narrow range centered around the mean, with some variability from random noise. This suggests the baseline model fails to accurately capture the full complexity of the underlying distribution exhibited by the real data. A visual comparison of subfigures 4.2b-4.2d suggests that the GM model may provide the closest approximation of the underlying data distributions, relative to the other models shown.

Generated Time Series Comparison for Specific Features To develop an intuitive understanding of the data windows generated by the models, we present random windows of original and generated data in Figure 4.3 for two specific features. The Baseline model's generated data for two features are depicted in Figure 4.3a and 4.3b. It is evident that the Baseline model generates synthetic data with a mean value and standard noise. We observe two distinct situations: in the first case (as shown in Figure 4.3a), the mean value of the sensor differs from the training set, resulting in a noticeable deviation between the two plots. In contrast, in the second case (Figure 4.3b), the mean values appear to be similar, and the synthetic data closely resemble the real data. The GM model's generated data for the same features are shown in Figure 4.3c and 4.3d. It is evident that the model successfully generates data within the correct range in Figure

4.3c, while in Figure 4.3d the generated data deviate from the desired range. Indeed, in Figure 4.3d, the generated data oscillates around 0.6, deviating from the original data clustered around 0.2. Overall, a notable difference between the generated and real data is the presence of increased fluctuations in the generated samples compared to the relatively flatter nature of the real data. Additionally, the TimeVAE model's generated data for the two features can be observed in Figure 4.3e and 4.3f present examples of generated data by the TimeVAE model for two other features. Notably, the synthetic data in these examples do not exhibit fluctuations. Nevertheless, it is evident that the ranges of the generated data differ from those of the original data. Figures 4.3g and 4.3h depict examples of data generated by the TimeGAN model for two sensor features. Compared to the other models, TimeGAN appears to more effectively capture the temporal dynamics of the data, producing coherent samples over successive time steps without anomalous fluctuations. The generated time-series patterns closely mimic those of the real data, indicating TimeGAN is better able to learn the underlying relationships governing how the feature values evolve sequentially over time.

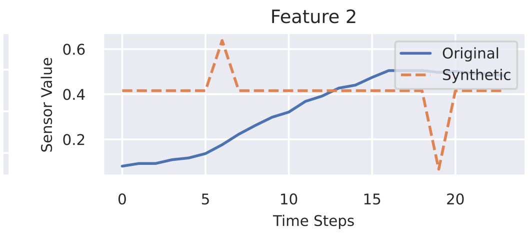
4.2.2 Quantitative evaluation

We report the quantitative evaluation metrics for the four models in Table 4.2, 4.3 and 4.4. Results in Table 4.2,4.3 and 4.4 reveal that different metrics may suggest different

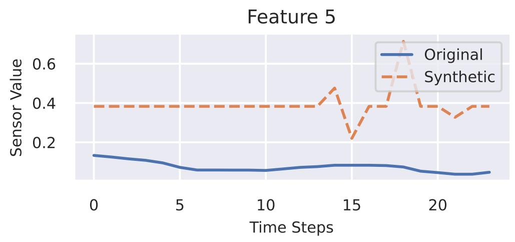
Model	**Performance Metrics**		
	DS	**PS**	**MMD**
Baseline	0.50	0.13	**0.00**
GM	0.50	**0.09**	**0.00**
TimeVAE	0.33	**0.09**	0.06
TimeGAN	**0.32**	0.16	0.03

Table 4.2: Quantitative evaluation metrics of generated data for Baseline, GM, TimeGAN, TimeVAE models for Energy Dataset. Metrics are Computed on Scaled Data. Lower values for all metrics indicate better model performance. Bold indicates best performance.

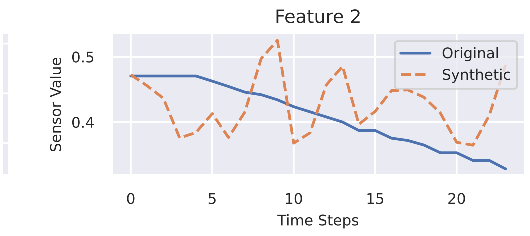
models as the best performer. For instance, TimeGAN achieves the best results in terms of DS, which is a measure of the classification model's ability to differentiate between real and generated data. This result is consistent with the fact that GANs are designed to train the generator to produce synthetic data that closely resembles real data, "cheating"



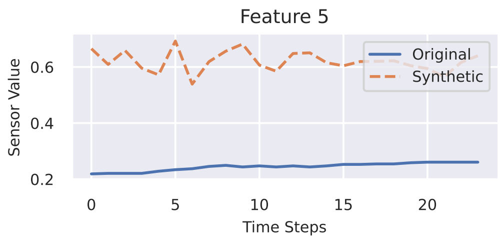
(a). Comparison of generated (orange) and real (blue) data for the second feature using the Baseline Model.



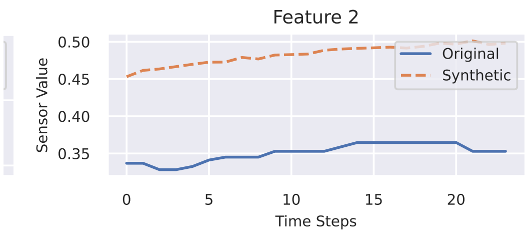
(b). Comparison of generated (orange) and real (blue) data for the fifth feature using the Baseline Model.



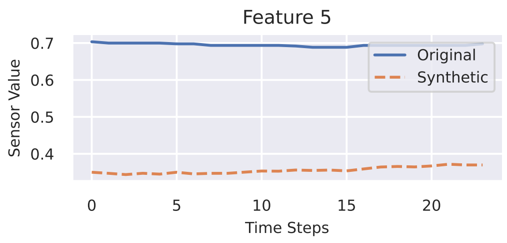
(c). Comparison of generated (orange) and real (blue) data for the second feature using the GM Model.



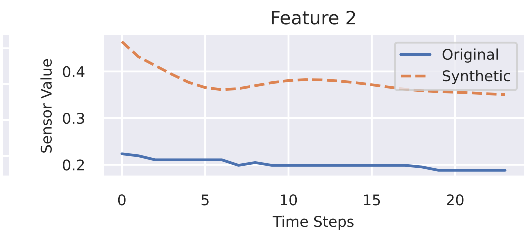
(d). Comparison of generated (orange) and real (blue) data for the fifth feature using the GM Model.



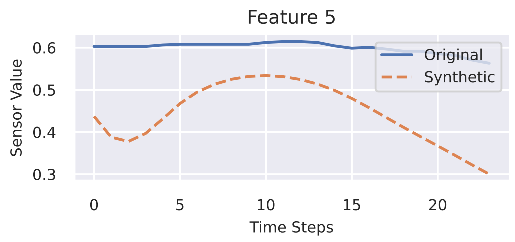
(e). Comparison of generated (orange) and real (blue) data for the second feature using the TimeVAE Model.



(f). Comparison of generated (orange) and real (blue) data for the fifth feature using the TimeVAE Model.



(g). Comparison of generated (orange) and real (blue) data for the second feature using the TimeGAN Model.



(h). Comparison of generated (orange) and real (blue) data for the fifth feature using the TimeGAN Model.

Figure 4.3: Comparison between the Generated (Orange) and Real (Blue) Data for the Second and Fifth Features of the Energy Dataset. A Randomly Selected Window of 24 Time Steps is Displayed, Showcasing Normalized Real Data and Data Generated by the Four Analyzed Models.

Model	**TSTR**			
	MSE	**Max Error**	**P90**	**MAPE**
Baseline	0.03	0.65	**0.28**	0.83
GM	**0.01**	**0.46**	0.37	**0.51**
TimeVAE	0.17	0.60	0.56	0.88
TimeGAN	0.03	0.49	0.38	1.61

Table 4.3: Comparison of TSTR metrics for generated data from the Baseline, GM, TimeGAN, and TimeVAE models on the Energy dataset. Metrics are Computed on Scaled Data. For all metrics, lower values indicate better model performance. The best performing model for each metric is highlighted in bold.

Model	**TRTS**			
	MSE	**Max Error**	**P90**	**MAPE**
Baseline	0.004	0.59	0.05	0.08
GM	**0.003**	0.51	0.32	**0.07**
TimeVAE	0.065	0.54	0.40	0.40
TimeGAN	0.016	**0.18**	**0.25**	0.16

Table 4.4: Comparison of TRTS metrics for generated data from the Baseline, GM, TimeGAN, and TimeVAE models on the Energy dataset. Metrics are Computed on Scaled Data. For all metrics, lower values indicate better model performance. The best performing model for each metric is highlighted in bold.

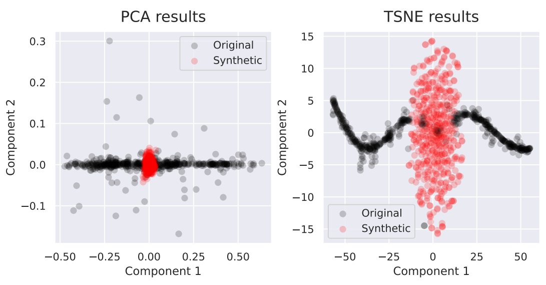
therefore the discriminator Goodfellow et al. [13].

As the DS measures how good the synthetically created data are, it can be used in theory to evaluate the performance of any generation model (not only time series generative models). Instead, the PS is particular to time-series data, as it should demonstrate the ability of a generative model to accurately predict the next time step in a sequence based on the previous observed time steps. According to the evaluation results, both the GM model and TimeVAE model achieved the best performance in terms of predictive score (PS).

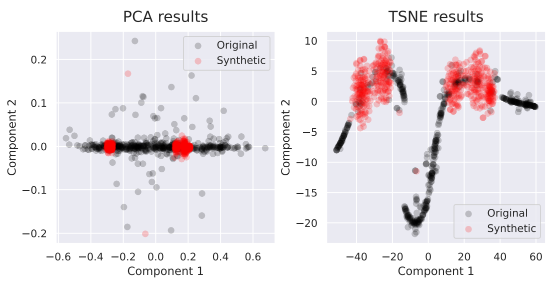
The MMD scores indicate that, from the perspective of measuring the discrepancy between distributions, all models achieved relatively low values. However, the Baseline and GM models exhibited the smallest discrepancies, reaching the lowest MMD scores of all the evaluated approaches. One possible explanation for this phenomenon is that the Baseline model's good performance may be attributed to its ability to produce a realistic distribution by relying on the mean of sensor values from the training dataset, augmented with additional noise.

TRTS should evaluate how realistic the generated data are, in other words how well the distributions within the real data were learnt, while TSTR should evaluate the diversity of the synthetic data and how well the variations in the real data are captured Yadav et al. [35]. Based on the evaluation results of the TSTR, the TimeVAE model achieved the best scores on all metrics except for the p90 metric. Based on the evaluation results of the TSTR, the best models are TimeGAN and GM. Across all models, the mean error metrics for regression were consistently lower when training on real data and testing on synthetic data, compared to training on synthetic and testing on real data. This result is plausible, as models generally perform best when both their training and test data are drawn from the same underlying distribution. By training exclusively on real data, the models may have been better able to learn representations tailored to the true statistical patterns and relationships present in the real-world sample, rather than having to generalize across domains.

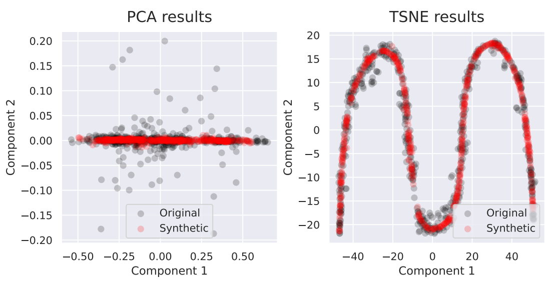
Upon evaluating generative models using all the previously presented metrics, the Baseline model's performance was found to be comparable to, or only slightly worse than, more complex models. This result is somewhat surprising since more complex models are generally expected to outperform simpler models. Therefore, we can conclude that



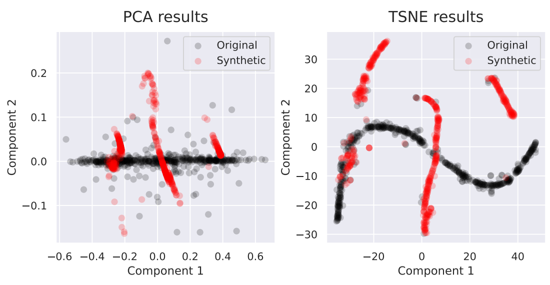
(a). Baseline model



(b). GM model



(c). VAE model



(d). GAN model

Figure 4.4: t-SNE and PCA visualization for the Baseline (top left), GM (top right), TimeVAE (bottom left), and TimeGAN (bottom right) models for Energy Dataset. Red denotes original data, and black denotes synthetic.

these metrics can be helpful in measuring certain aspects of a generative model's performance, but they may not always be reliable indicators of the overall power of a model. Moreover, we emphasize that the results of metrics such as PS, DS, TSTR, and TRTS depend on the specific definition of the classification/regression model, and this fact makes these metrics even less reliable.

4.3 Results on the Nox Dataset

We present the findings of our qualitative evaluation in Subsection 4.3.1 and the results of our quantitative evaluation of the models in Subsection 4.3.2. Furthermore, we introduce a novel analysis that deviates from the existing literature. In Subsection 4.3.3, we compare the statistical information of the generated data with respect to the real dataset.

4.3.1 Qualitative evaluation

A quantitative evaluation was performed for the NOx dataset, similar to what was done for the Energy dataset. The results of this evaluation are reported in Figure 4.4. Figure 4.4a shows that the baseline model used for the NOx dataset faced similar challenges as it did for the energy dataset. It fails to accurately capture the complexity of the underlying distribution observed in the real data. Instead, it only produced an approximation of the mean value along with some random noise. Furthermore, upon analyzing Figure4.4b and 4.4d, it becomes evident that both TimeGAN and GM faced greater challenges in generating synthetic data for the NOx dataset compared to the Energy dataset. In contrast, the TimeVAE model exhibited notably better performance in this evaluation.

Generated Time Series Comparison for Specific Features Similar to the approach used for the Energy Dataset, we have randomly chosen windows of original and generated data for two specific features from the Nox Dataset. These selected windows are visualized in Figure 4.5. However, in the case of the Nox Dataset, the insights we can derive from these random plots are relatively limited compared to the Energy dataset. We can observe that the real data exhibits more fluctuations compared to the synthetic data. This is consistent with the fact that the NOx dataset is more irregular and noisy with respect to the energy consumption data, as we can see in Figures 3.1 and 3.2. These variations over time are difficult to accurately capture and replicate in synthetically generated data, and the generative models tend to smooth out the fluctuations present in the real-world data.

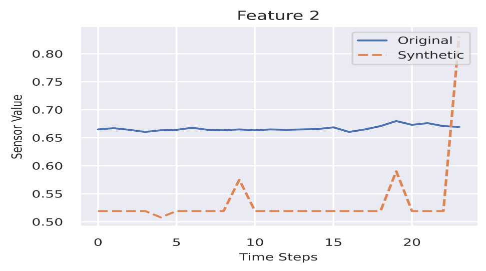
4.3.2 Quantitative evaluation

We report the quantitative evaluation metrics for the four models in Tables 4.5, 4.6 and 4.7. According to PS and DS we can say that with the NOx dataset the performance of

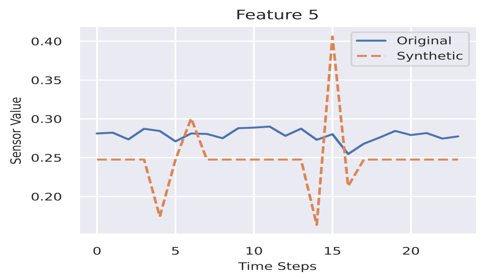
Model	**Performance Metrics**		
	DS	**PS**	**MMD**
Baseline	0.50	0.04	0.00
GM	0.16	0.04	0.00
TimeVAE	**0.01**	0.13	0.00
TimeGAN	0.44	**0.03**	0.00

Table 4.5: Comparison of predictive score (PS), discriminative score (DS), and maximum mean discrepancy (MMD) metrics for generated data from the Baseline, GM, TimeGAN, and TimeVAE models on the NOx dataset. Metrics are Computed on Scaled Data. For all metrics, lower values indicate better model performance. The best performing model for each metric is highlighted in bold.

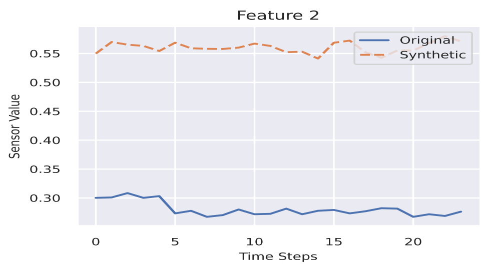
TimeVAE and GM improves, while the performance of TimeGAN decreases. Moreover,



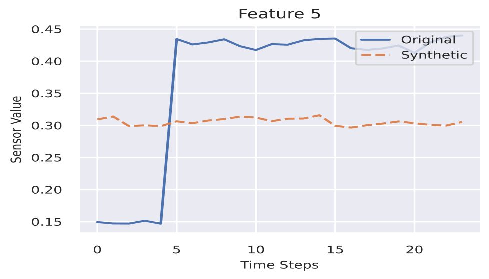
(a). Comparison of generated (orange) and real (blue) data for the second feature using the Baseline Model.



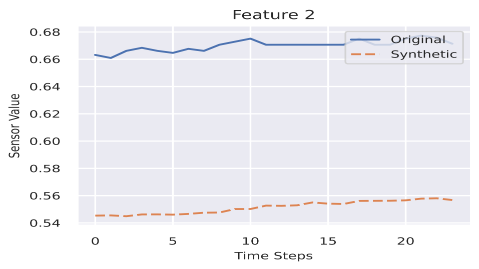
(b). Comparison of generated (orange) and real (blue) data for the fifth feature using the Baseline Model.



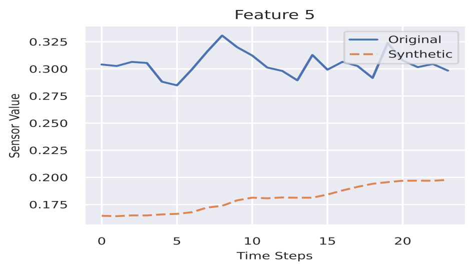
(c). Comparison of generated (orange) and real (blue) data for the second feature using the GM Model.



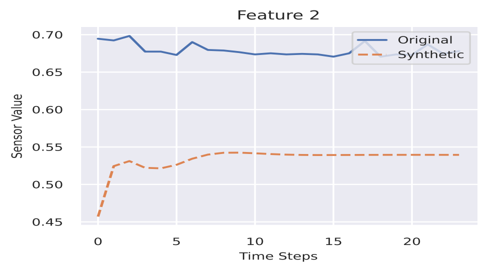
(d). Comparison of generated (orange) and real (blue) data for the fifth feature using the GM Model.



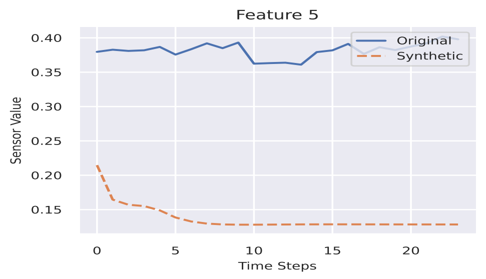
(e). Comparison of generated (orange) and real (blue) data for the second feature using the TimeVAE Model.



(f). Comparison of generated (orange) and real (blue) data for the fifth feature using the TimeVAE Model.



(g). Comparison of generated (orange) and real (blue) data for the second feature using the TimeGAN Model.



(h). Comparison of generated (orange) and real (blue) data for the fifth feature using the TimeGAN Model.

Figure 4.5: Comparison between the Generated (Orange) and Real (Blue) Data for the Second and Fifth Features of the NOx Dataset. A Randomly Selected Window of 24 Time Steps is Displayed, Showcasing Normalized Real Data and Data Generated by the Four Analyzed Models.

the MMD metric alone cannot adequately discriminate between the models. As shown in

Model	TSTR			
	MSE	**Max Error**	**P90**	**MAPE**
Baseline	0.023	0.610	0.204	0.677
GM	**0.022**	0.600	**0.202**	**0.667**
TimeVAE	0.079	**0.341**	0.315	1.648
TimeGAN	0.063	0.513	0.306	1.124

Table 4.6: Comparison of TSTR metrics for generated data from the Baseline, GM, TimeGAN, and TimeVAE models on the NOx dataset. Metrics are Computed on Scaled Data. For all metrics, lower values indicate better model performance. The best performing model for each metric is highlighted in bold.

Model	TRTS			
	MSE	**Max Error**	**P90**	**MAPE**
Baseline	0.018	0.402	0.135	0.344
GM	**0.016**	0.140	0.138	**0.325**
TimeVAE	0.074	0.510	0.349	0.675
TimeGAN	0.056	**0.238**	**0.238**	0.489

Table 4.7: Comparison of TRTS metrics for generated data from the Baseline, GM, TimeGAN, and TimeVAE models on the NOx dataset.Metrics are Computed on Scaled Data. For all metrics, lower values indicate better model performance. The best performing model for each metric is highlighted in bold.

Tables 4.6 and 4.7, also the TSTR and TRTS metrics identified different best-performing models when evaluating performance on the NOx and energy datasets. When evaluating the models on the NOx dataset, the TRTS metric scores tended to be lower than the corresponding TSTR metric scores, as was also seen with the Energy dataset.

4.3.3 Comparing statistics

To evaluate how well the generative models capture the distribution of the real data, we compared the mean and standard deviation statistics between the real and synthetic datasets. The mean and standard deviation provide a simple summary of the central tendency and spread of the data distributions. By comparing these statistics, we can assess whether the generative models are able to match these basic properties of the real data distribution. The mean and standard deviation comparisons between the real and synthetic datasets are shown in Tables 4.8 and 4.9, respectively.

Feature	Real	Baseline	GM	TimeVAE	TimeGAN
x_1	0.590	0.589	0.588	0.590	0.570
x_6	0.512	0.515	0.514	0.512	0.438
x_8	0.403	0.396	0.396	0.403	0.397
x_{10}	0.194	0.195	0.195	0.194	0.162

Table 4.8: Comparison of Mean for Normalized Features x_1, x_6, x_8, x_{10} in the NOx Dataset: Real vs Synthetic Data Generated by Baseline, GM, TimeVAE, and TimeGAN Models.

Feature	Real	Baseline	GM	TimeVAE	TimeGAN
x_1	0.077	0.097	0.020	0.077	0.005
x_6	0.105	0.127	0.069	0.105	0.027
x_8	0.154	0.176	0.129	0.154	0.012
x_{10}	0.085	0.094	0.040	0.085	0.004

Table 4.9: Comparison of Standard Deviation for Normalized Features x_1, x_6, x_8, x_{10} in the NOx Dataset: Real vs Synthetic Data Generated by Baseline, GM, TimeVAE, and TimeGAN Models.

The analysis of correlation between NOX emissions (represented by feature x_{35}) and the other features in the dataset, as well as its preservation in synthetic data, reveals interesting insights. In Figure 4.6, we present the correlations observed in real data, while Figures 4.7, 4.8, and 4.10 illustrate the correlations in synthetic data generated by the Baseline, GM, and TimeGAN models, respectively. In Figure 4.7, it is clear that the cor-

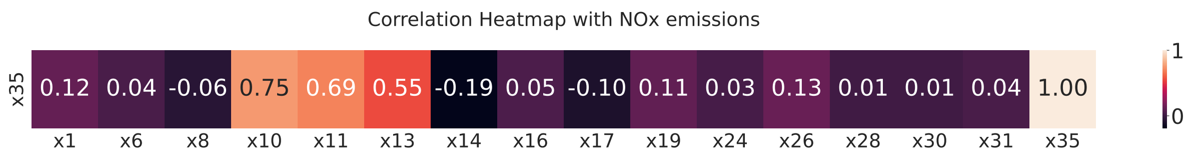


Figure 4.6: Correlation between Each Feature and the NOx Emission Feature x_{35} in the Real Dataset (Normalized Data).

relation values for the data generated by the Baseline model are all zero. This indicates the absence of a linear association between the variables being compared, specifically between NOX emissions and the other features in the dataset. The lack of significant linear correlation suggests that the Baseline model fails to capture and reproduce the same relationships observed in the real data. Therefore, the synthetic data generated by the Baseline model does not accurately reflect the underlying correlations present in the original dataset. In Figure 4.9 and 4.10, we can observe that the GM and TimeVAE mod-

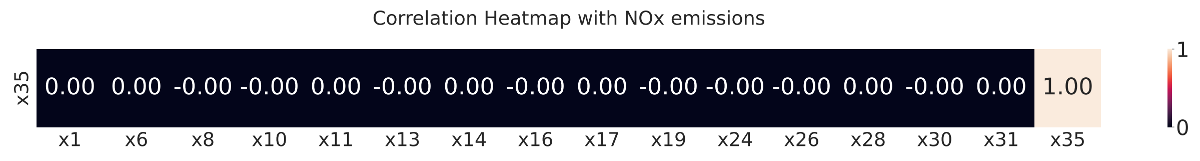


Figure 4.7: Correlation between Each Feature and the NOx Emission Feature x_{35} in the Synthetic Dataset generated by the Baseline model.

els exhibit a higher degree of correlation preservation compared to the Baseline model. Specifically, the highest correlation values consistently appear in the same three variables as in the real data: x_{10}, x_{11}, x_{13}.This finding suggests that both the GM and TimeVAE models have a better ability to capture and preserve the underlying relationships between these specific variables and NOX emissions compared to the Baseline model. The fact that the highest correlation values align with those in the real data indicates a stronger fidelity in reproducing the correlations between these variables. These findings suggest

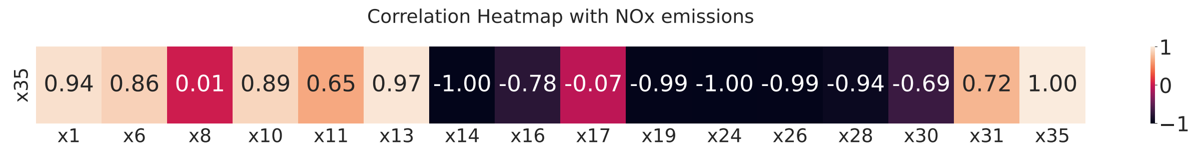


Figure 4.8: Correlation between Each Feature and the NOx Emission Feature x_{35} in the Synthetic Dataset generated by the GM model.

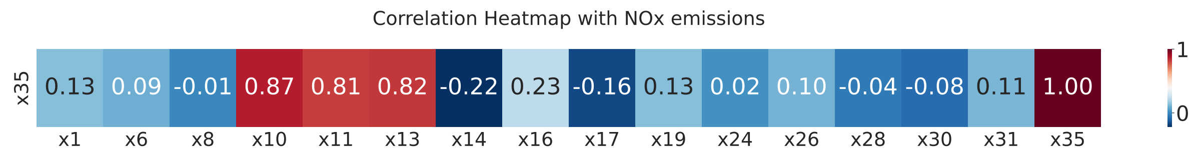


Figure 4.9: Correlation between Each Feature and the NOx Emission Feature x_{35} in the Synthetic Dataset generated by the TimeVAE model.

that the TimeVAE model outperforms the other models in capturing the underlying relationship between these specific variables and NOX emissions. While all models exhibit a modest capture of the correlation between NOX emissions and x10, the GM, Baseline, and GAN models fail to capture the correlation for features such as x17, x19, and x24, which reaches approximately -1.

We present autocorrelation and partial autocorrelation plots for each feature in the dataset in Figure 4.11. In Figure 4.11, it can be observed that the autocorrelation values in the real dataset consistently hover around 1 for all lags. This high autocorrelation sug-

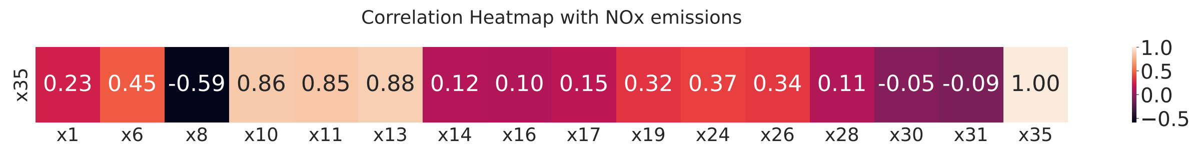


Figure 4.10: Correlation between Each Feature and the NOx Emission Feature x_{35} in the Synthetic Dataset generated by the TimeGAN model.

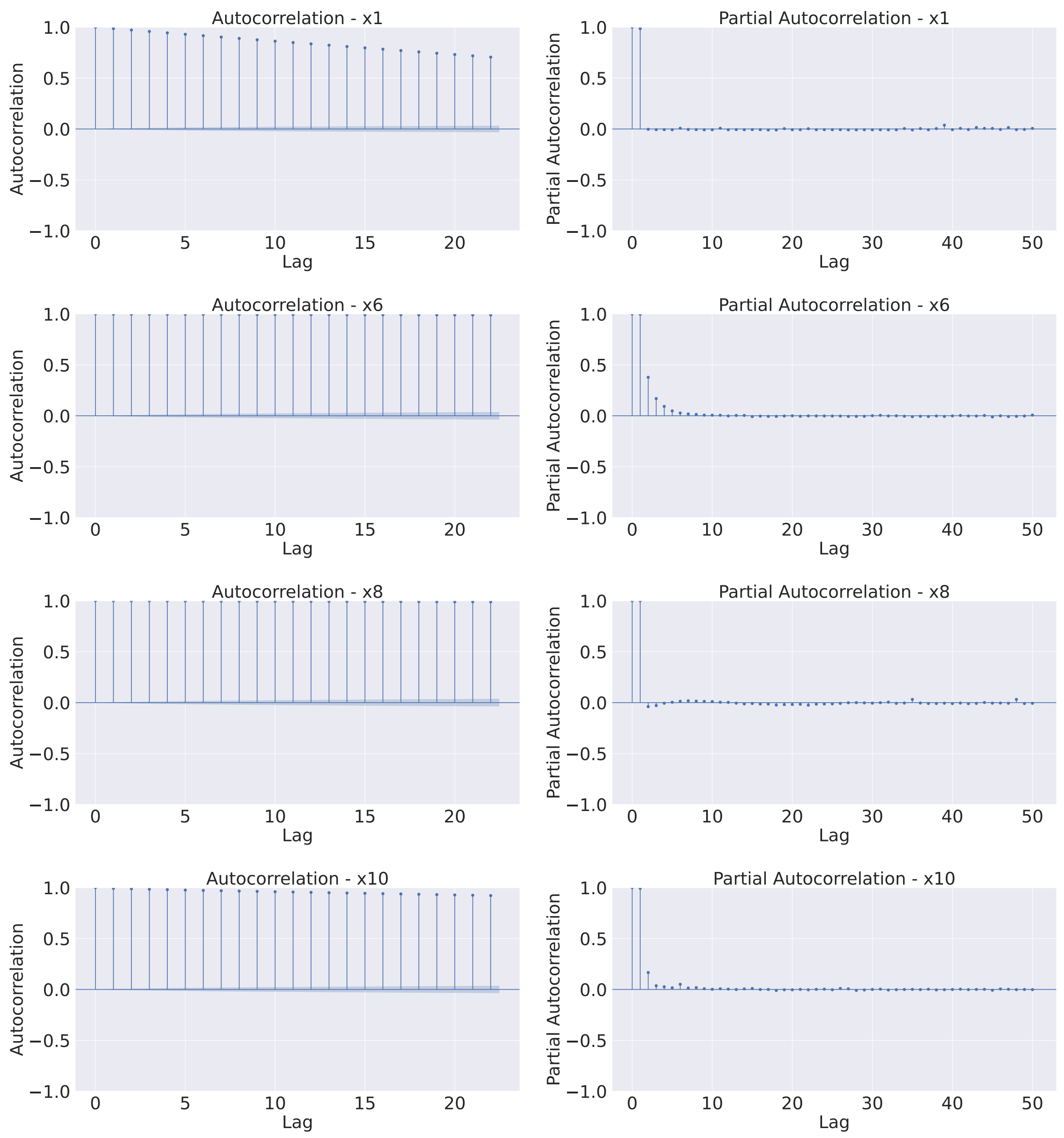


Figure 4.11: Autocorrelation and Partial Autocorrelation Plots for Four Normalized Features (x1, x6, x8, and x10) in the Real Dataset.

gests a strong dependence of the variable's values on their own past values, indicating a consistent pattern of increase or decrease.

We present the autocorrelation plots for synthetic data generated by the Baseline, GM, TimeVAE, and TimeGAN models in Figures 4.12, 4.13, 4.14, and 4.15, respectively. Notably, there is a significant difference observed between the complex models (TimeVAE and TimeGAN, as shown in Figures 4.14 and 4.15) and the simpler ones (the Baseline and GM models, depicted in Figures 4.12 and 4.13). For TimeVAE and TimeGAN, the

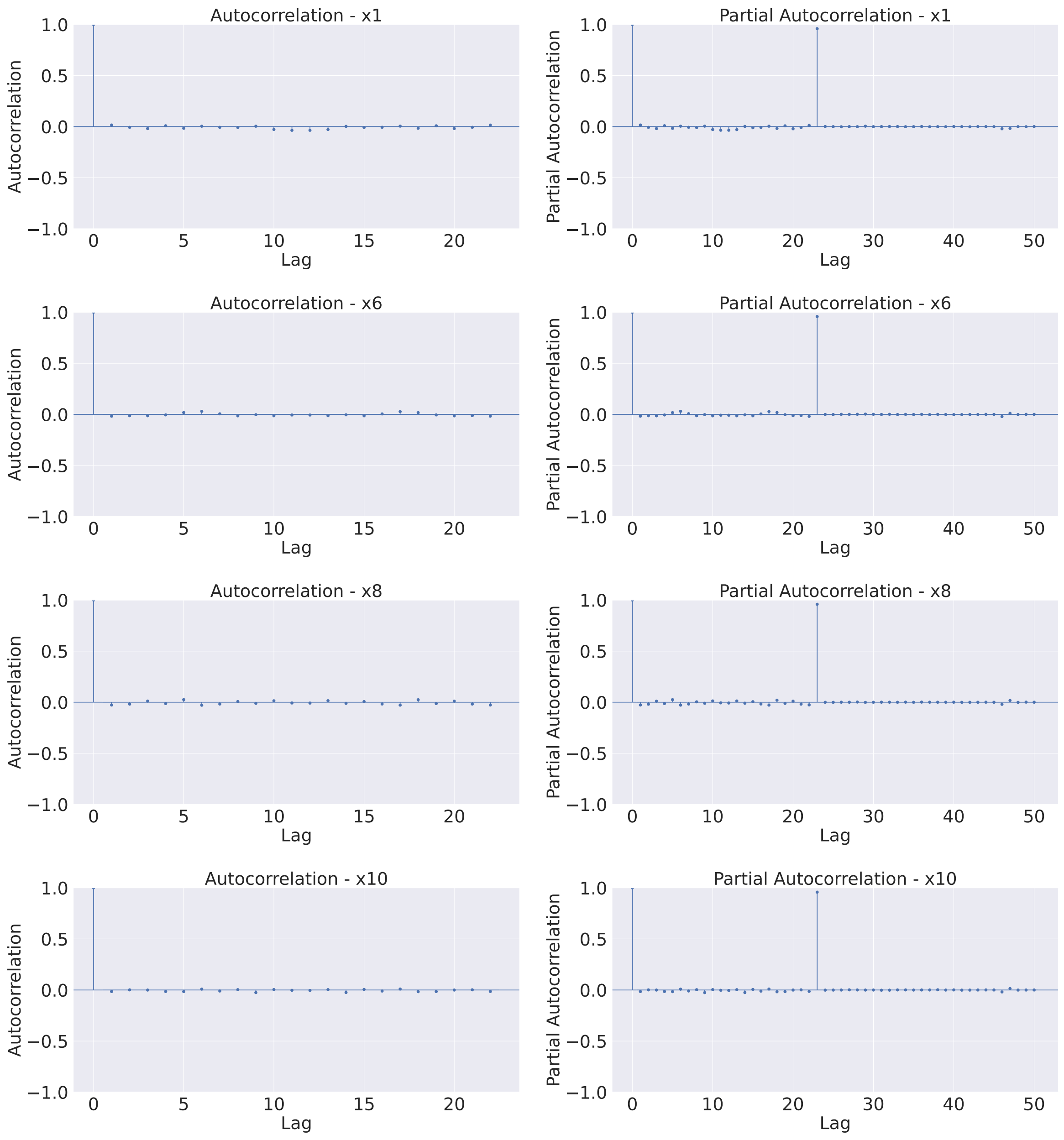


Figure 4.12: Autocorrelation and Partial Autocorrelation Plots for Four Metrics are Computed on Scaled Data. Features (x1, x6, x8, and x10) in the Synthetic Dataset Generated by the Baseline Model.

autocorrelation values are initially high and gradually decrease with increasing lags until reaching zero for the last lags within the selected window length (24 timestamps). On

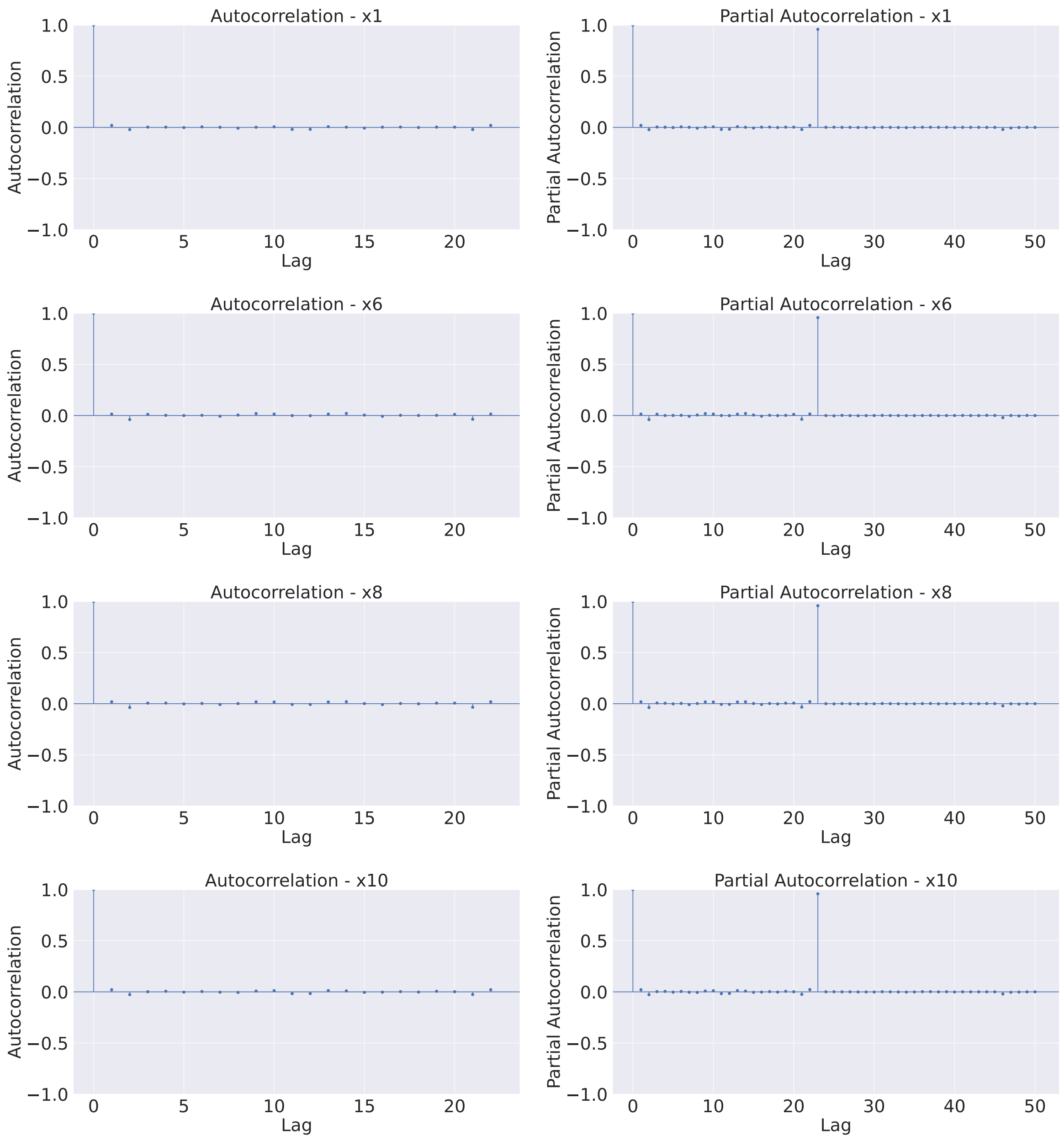


Figure 4.13: Autocorrelation and Partial Autocorrelation Plots for Four Normalized Features (x1, x6, x8, and x10) in the Synthetic Dataset Generated by the GM Model.

the other hand, the simpler models exhibit almost zero autocorrelation throughout. We present the histograms depicting NOx emissions in both real data and the synthetic data generated by the four considered models, as shown in Figure 4.16.

Upon observation, it is apparent that the Baseline model fails to capture the variations present in the real data distribution. The distributions generated by the VAE model exhibit a closer resemblance to the real data distribution, while the GAN model also displays a notable similarity. On the other hand, the GM model appears to generate values

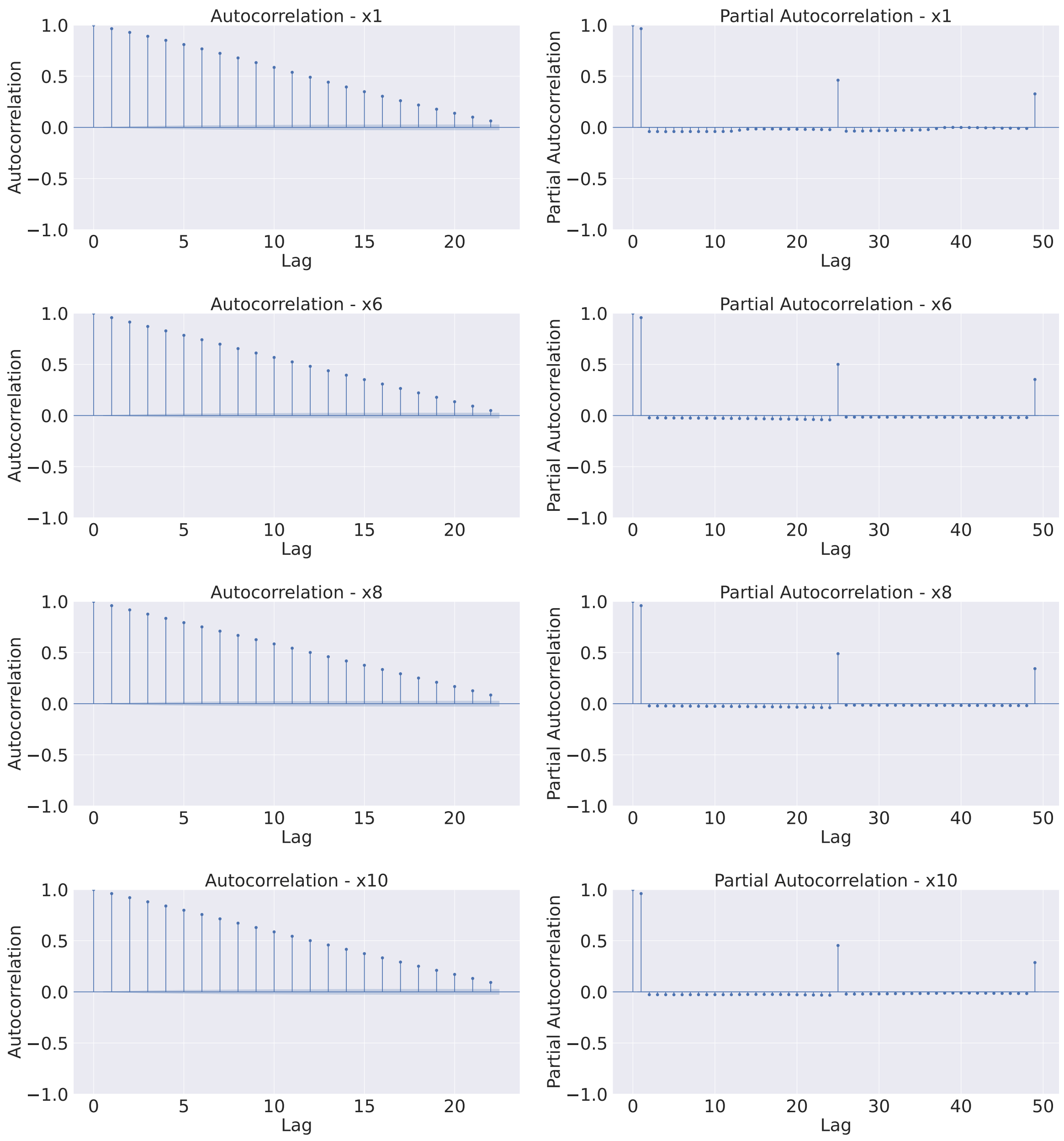


Figure 4.14: Autocorrelation and Partial Autocorrelation Plots for Four Normalized Features (x1, x6, x8, and x10) in the Synthetic Dataset Generated by the TimeVAE Model.

concentrated around two distinct points, specifically 0.375 and 0.395.

4.4 Reliability in visualization results

It is important to consider several factors when using t-SNE and PCA for visualization. Firstly, t-SNE is a stochastic method, meaning that multiple t-SNE plots generated from the same dataset can differ. This randomness stems from the algorithm's random initial-

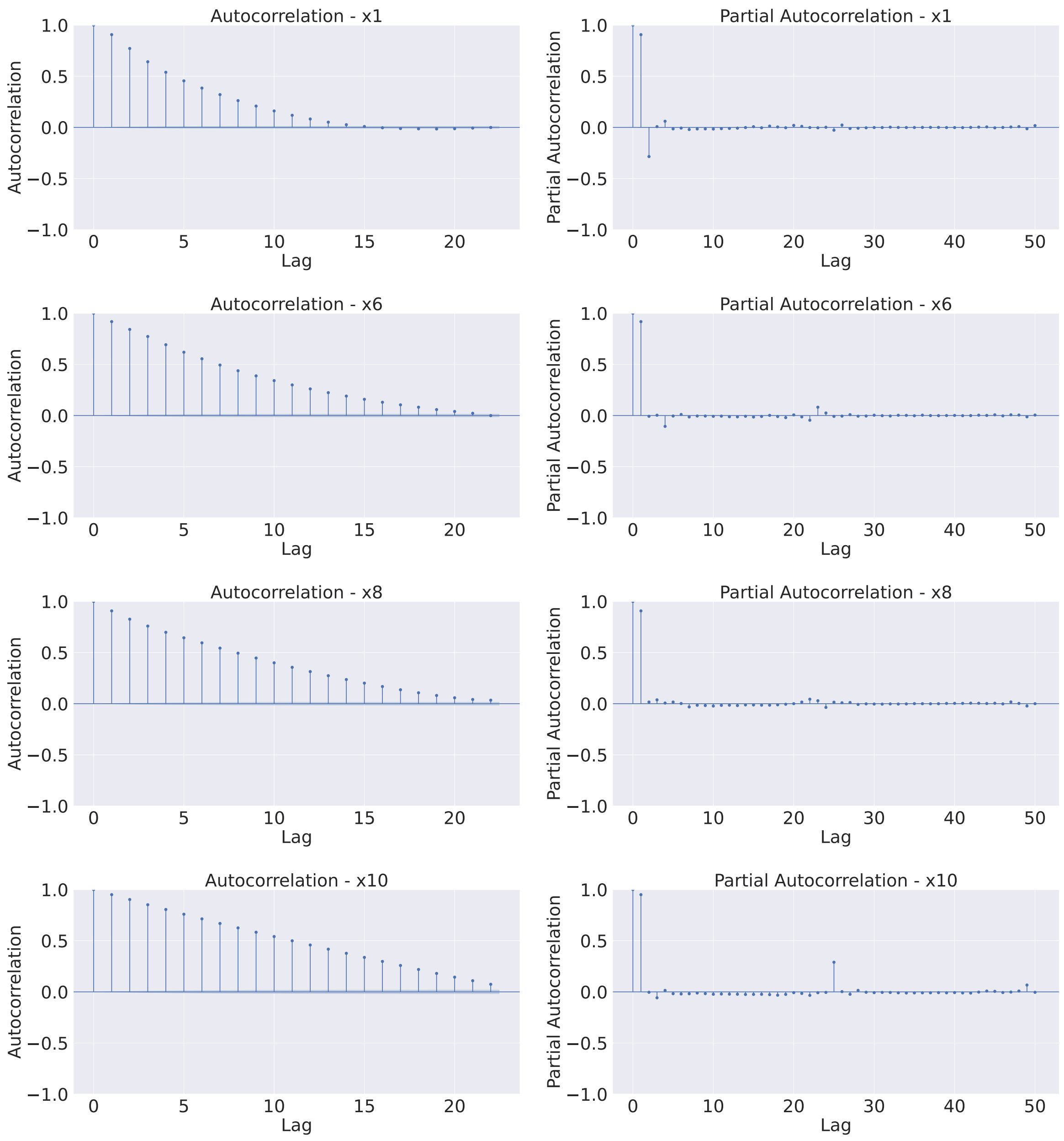
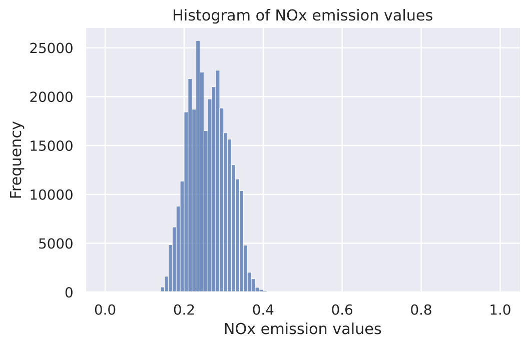
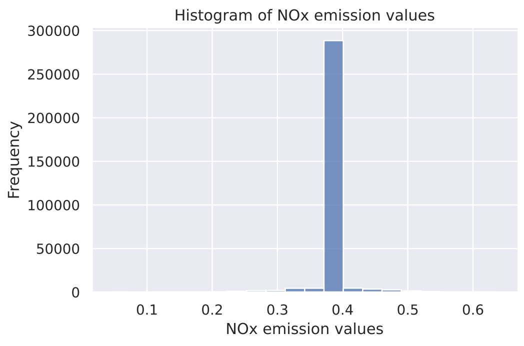


Figure 4.15: Autocorrelation and Partial Autocorrelation Plots for Four Normalized Features (x1, x6, x8, and x10) in the Synthetic Dataset Generated by the TimeGAN Model.

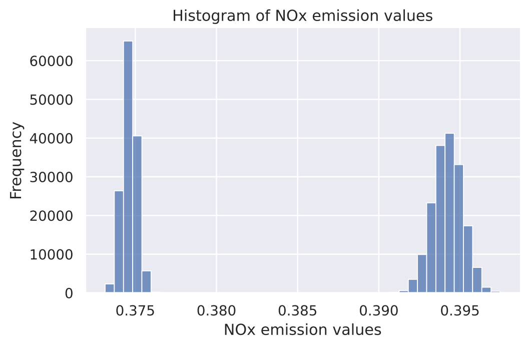
ization. On the other hand, PCA, in contrast, does not involve random initialization or sampling, and its computations are deterministic mathematical operations. In addition to the stochastic nature and random initialization, it is crucial to note that visualization is typically performed on a subset of the data. Since plotting thousands of points can be impractical, a representative batch of data is selected for visualization. Consequently, the specific subset chosen for visualization can influence the resulting t-SNE plot. Furthermore, when reducing data to two dimensions for visualization, some information is



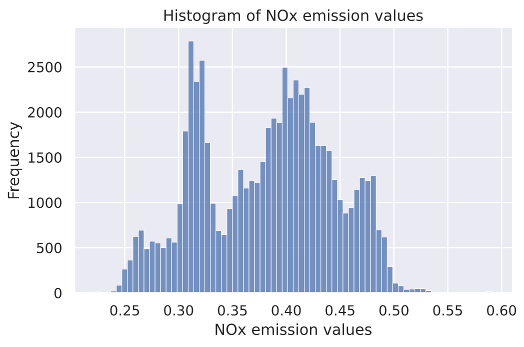
(a). Real Data



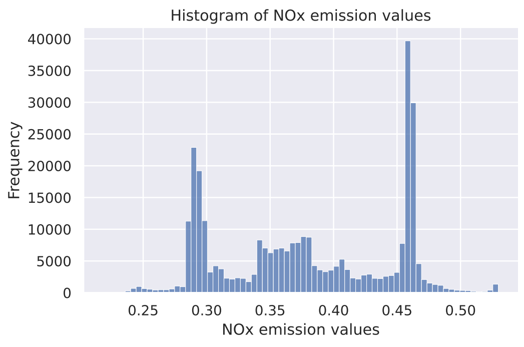
(b). Baseline Model



(c). GM Model

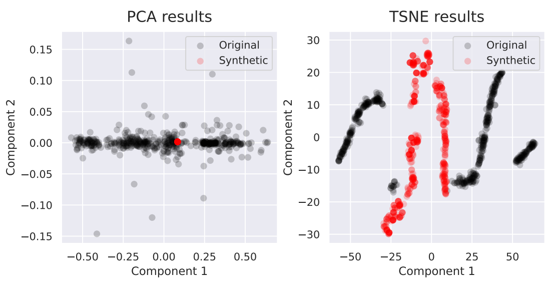


(d). VAE Model

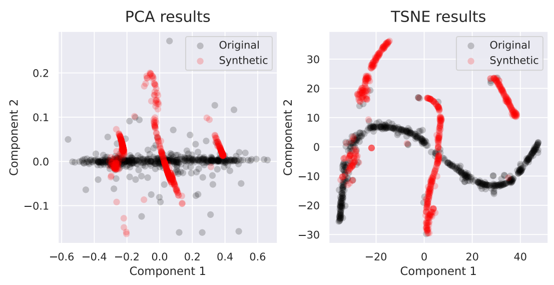


(e). GAN Model

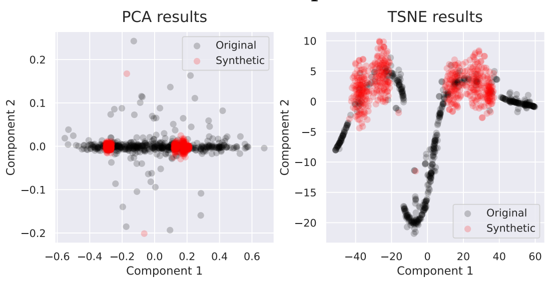
Figure 4.16: "Histograms of Normalized NOx Emissions in Real Data (a) and Synthetic Data Generated by the Four Considered Models: Baseline Model (b), GM Model (c), VAE Model (d), and GAN Model (e).



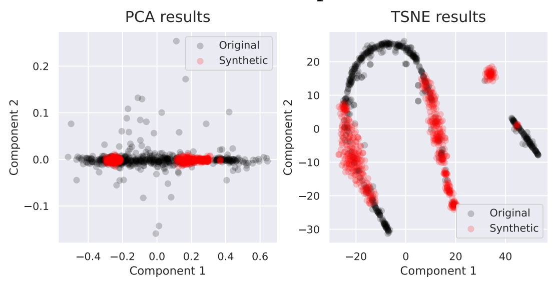
(a). TimeGAN experiment 1



(b). TimeGAN experiment 2



(c). GM experiment 1



(d). GM experiment 2

Figure 4.17: t-SNE and PCA Visualization of the NOx Dataset for TimeGAN (top left and top right) and GM (bottom left and bottom right) models. The visualizations depict two different experiments conducted with the same generation pipelines. The original data is represented in red, while the synthetic data is represented in black.

inevitably lost. Time-dependent relationships and other higher-dimensional structures may not be fully captured in the resulting two-dimensional representation. Based on our experience, we have observed that results can vary even when the generative procedure remains unchanged. Figure 4.17 presents an example of t-SNE and PCA visualizations of the NOx Dataset for TimeGAN (top left and top right) and GM (bottom left and bottom right) models. These visualizations illustrate two different experiments conducted using the same generation pipelines. Notably, Figure 4.17a and Figure 4.17b, as well as Figure 4.17c and Figure 4.17d, demonstrate distinct results, highlighting the variability between the experiments.

4.5 Task related metrics on NOx Dataset

To address the limitations of the evaluation metrics discussed in Sections 4.2 and 4.3, we decided to move towards a task-related metric that could better capture the performance of the evaluated models in our specific application. As we described in Chapter 1, our main goal was to develop time series generation models to improve industrial models with scarcity of data. Therefore, we integrated the generated data into the NOx

dataset, and evaluated the resulting performance metrics of our regression model to estimate NOx emissions. If the model generates coherent synthetic data, we expect that the performance of the regression model will improve when trained on both real and synthetic data. Therefore, the task-related metric will be the performance metrics of the regression model when trained on the combined set of real and synthetic data.
Table 4.1 presents the performance metrics for our industrial regression task on the scaled NOx dataset without adding synthetic data.
Tables 4.10, 4.11, 4.12 and 4.13 present the respective metrics (MSE, MAPE, p90, Maximum Error) for the Regression Model with the addition of of synthetic data. We show how the results of the regression change as we progressively add increasing percentages of synthetic data and we report the mean and the variances of the regression metrics over 3 attempts. For example, a percentage of 30% means that we add an amount of synthetic data equal to 30% of the real data size. Moreover, the bottom row shows the average performance across all percentages of synthetic data.

Percentage	**MSE**			
	Baseline	**GM**	**TimeVAE**	**TimeGAN**
30 %	0.023	**0.017**	0.022	**0.014**
50 %	0.023	0.019	0.020	**0.021**
mean	0.023	0.018	0.021	0.016

Table 4.10: Mean Squared Error (MSE) of the Regression Model with varying percentages of synthetic data added to real data. Metrics are Computed on Scaled Data. Bold results indicate better performance compared to the results obtained without data augmentation, where the MSE was **0.018**.

Results in Tables 4.10, 4.11, 4.12 and 4.13 indicate that adding synthetic data can lead to improved performance of the regression model across all the evaluated metrics. Notably, the Baseline model consistently yields the poorest results in comparison to the other evaluation metrics assessed in Section 4.3.2. Interestingly, TimeGAN consistently outperforms the other models, emerging as the optimal choice for enhancing regression metrics. Adding data generated by TimeGAN, we alwayis improve the performance of all the regression metrics. The addition of synthetic data from GM, TimeVAE, and TimeGAN exhibits minimal adverse effects on model performance, suggesting that the generated

Percentage	MAPE			
	Baseline	**GM**	**TimeVAE**	**TimeGAN**
30%	0.636	0.541	0.609	**0.485**
50%	0.620	0.573	0.612	**0.593**
mean	0.628	0.557	0.611	0.488

Table 4.11: Mean Absolute Percentage Error (MAPE) of the Regression Model with varying percentages of synthetic data added to real data. Metrics are Computed on Scaled Data. Bold results indicate better performance compared to the results obtained without data augmentation, where the MAPE was **0.540**.

Percentage	p90			
	Baseline	**GM**	**TimeVAE**	**TimeGAN**
30%	0.219	**0.197**	0.211	**0.187**
50%	0.218	0.205	0.215	**0.211**
mean	0.219	0.201	0.213	0.191

Table 4.12: 90th percentile error value (p90) of the Regression Model with varying percentages of synthetic data added to real data. Metrics are Computed on Scaled Data. Bold results indicate better performance compared to the results obtained without data augmentation, where the p90 was **0.198**.

Percentage	Max Error			
	Baseline	**GM**	**TimeVAE**	**TimeGAN**
30%	0.588	0.607	0.592	0.617
50%	0.589	0.600	0.592	0.593
mean	0.589	0.604	0.592	0.605

Table 4.13: Max Error of the Regression Model with varying percentages of synthetic data added to real data. Metrics are Computed on Scaled Data. Bold results indicate better performance compared to the results obtained without data augmentation, where the Max Error was **0.605**.

data closely aligns with the true distribution of real data. In terms of determining the appropriate amount of data to add, we conducted experiments with varying percentages and observed that excessively increasing the amount of added data led to a decrease in performance. This trend is evident across all the evaluated metrics, as we consistently obtained inferior results when using 50% of additional data compared to using 30%.

Chapter 5

Conclusions

We introduce a novel approach for monitoring NOx emissions through the utilization of generative models, addressing the data scarcity challenge commonly encountered in industrial settings. Our study encompasses a comprehensive exploration of various generative techniques, assessing their performance on two distinct datasets: one sourced from existing literature and another derived from our specific industrial application.

In our analysis, we delve into the complexities associated with evaluating generated time series data. We highlight the limitations of conventional evaluation methods commonly employed in the literature, emphasizing that they may not consistently serve as reliable indicators of overall model performance. To address this issue, we propose the incorporation of additional, simpler comparison metrics to assess the fidelity of generated data. These metrics include correlation with NOx emissions, autocorrelation, as well as basic statistical measures such as mean and standard deviations. Furthermore, we advocate for the inclusion of visual plots showcasing the generated data over time, providing an intuitive assessment of the model's ability to produce coherent data within a given window.

To cater to the specific needs of industrial applications, we put forth a task-specific evaluation metric. This metric underscores the potential of generative models while emphasizing the importance of developing evaluation metrics tailored to the unique requirements of the intended application. Through our comprehensive analysis, we highlight the necessity of considering multiple evaluation metrics when assessing generated time series data. Notably, our findings reveal that the incorporation of more complex models yields superior results, contradicting the outcomes obtained by conventional metrics commonly employed in the literature.

In future work, the authors plan to explore the potential of synthetic data to improve the performance of other industrial models, particularly anomaly detection models, and explore other evaluation metrics that considers temporal correlation between generated samples.

Bibliography

[1] Hiba Arnout, Johanna Bronner, and Thomas Runkler. Evaluation of generative adversarial networks for time series data. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2021. doi: 10.1109/IJCNN52387.2021.9534373.

[2] Eoin Brophy, Zhengwei Wang, Qi She, and Tomas Ward. Generative adversarial networks in time series: A survey and taxonomy, 2021.

[3] Eoin Brophy, Zhengwei Wang, Qi She, and Tomás Ward. Generative adversarial networks in time series: A systematic literature review. *ACM Comput. Surv.*, 55 (10), feb 2023. ISSN 0360-0300. doi: 10.1145/3559540. URL https://doi.org/10.1145/3559540.

[4] Fred Bryant and Paul Yarnold. Principal-component analysis and exploratory and confirmatory factor analysis. 01 2001.

[5] Luis Candanedo. Appliances energy prediction. UCI Machine Learning Repository, 2017. URL https://doi.org/10.24432/C5VC8G.

[6] Edward Choi, Siddharth Biswal, Bradley A. Malin, Jon Duke, Walter F. Stewart, and Jimeng Sun. Generating multi-label discrete electronic health records using generative adversarial networks. *CoRR*, abs/1703.06490, 2017. URL http://arxiv.org/abs/1703.06490.

[7] Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. Timevae: A variational auto-encoder for multivariate time series generation, 2021.

[8] Chris Donahue, Julian J. McAuley, and Miller S. Puckette. Synthesizing audio with

generative adversarial networks. *CoRR*, abs/1802.04208, 2018. URL http://arxiv.org/abs/1802.04208.

[9] Gintare Karolina Dziugaite, Daniel M. Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization, 2015.

[10] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. 2017.

[11] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans, 2017.

[12] Federico Gatta, Fabio Giampaolo, Edoardo Prezioso, Gang Mei, Salvatore Cuomo, and Francesco Piccialli. Neural networks generative models for time series. *Journal of King Saud University - Computer and Information Sciences*, 34, 07 2022. doi: 10.1016/j.jksuci.2022.07.010.

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[14] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL http://arxiv.org/abs/1308.0850.

[15] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL http://jmlr.org/papers/v13/gretton12a.html.

[16] Debapriya Hazra and Yung-Cheol Byun. Synsiggan: Generative adversarial networks for synthetic biomedical signal generation. *Biology*, 9(12):441, 2020.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[18] Brian Kenji Iwana and Seiichi Uchida. An empirical survey of data augmentation for time series classification with neural networks. *PLoS ONE*, 16(7):e0254841, 2021. doi: 10.1371/journal.pone.0254841.

[19] Paul Jeha, Michael Bohlke-Schneider, Pedro Mercado, Shubham Kapoor, Rajbir Singh Nirwan, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. PSA-GAN: Progressive self attention GANs for synthetic time series. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Ix_mh42xq5w.

[20] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019. doi: 10.1561/2200000056. URL https://doi.org/10.1561%2F2200000056.

[21] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[22] Timo Korpela, Pekka Kumpulainen, Yrjö Majanne, and Anna Häyrinen. Model based nox emission monitoring in natural gas fired hot water boilers. *IFAC-PapersOnLine*, 48(30):385–390, 2015. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2015.12.409. URL https://www.sciencedirect.com/science/article/pii/S2405896315030517. 9th IFAC Symposium on Control of Power and Energy Systems CPES 2015.

[23] Miron Kursa and Witold Rudnicki. Feature selection with boruta package. *Journal of Statistical Software*, 36:1–13, 09 2010. doi: 10.18637/jss.v036.i11.

[24] Alex Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks, 2016.

[25] Yulong Lu and Jianfeng Lu. A universal approximation theorem of deep neural networks for expressing probability distributions. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3094–3105. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/2000f6325dfc4fc3201fc45ed01c7a5d-Paper.pdf.

[26] You Lv, Jizhen Liu, Tingting Yang, and Deliang Zeng. A novel least squares support vector machine ensemble model for nox emission prediction of a coal-fired boiler. *Energy*, 55:319–329, 2013. ISSN 0360-5442. doi: https://doi.org/10.1016/

j.energy.2013.02.062. URL https://www.sciencedirect.com/science/article/pii/S0360544213001953.

[27] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training, 2016.

[28] B. J. Radl. Neural networks prove effective at nox reduction. *Modern power systems*, 20:59–62, 2000. URL https://api.semanticscholar.org/CorpusID:114468151.

[29] Douglas A. Reynolds. Gaussian mixture models. In *Encyclopedia of Biometrics*, 2009. URL https://api.semanticscholar.org/CorpusID:1063711.

[30] Padmanaba Srinivasan and William J. Knottenbelt. Time-series transformer generative adversarial networks, 2022.

[31] Yuki Sumiya, Kazumasa Horie, Hiroaki Shiokawa, and Hiroyuki Kitagawa. Nr-gan: Noise reduction gan for mice electroencephalogram signals. In *Proceedings of the 2019 4th International Conference on Biomedical Imaging, Signal Processing*, pages 94–101, 2019.

[32] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016. URL http://arxiv.org/abs/1609.03499.

[33] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

[34] Peiran Xie, Mingming Gao, Hongfu Zhang, Yuguang Niu, and Xiaowen Wang. Dynamic modeling for nox emission sequence prediction of scr system outlet based on sequence to sequence long short-term memory network. *Energy*, 190:116482, 2020. ISSN 0360-5442. doi: https://doi.org/10.1016/j.energy.2019.116482. URL https://www.sciencedirect.com/science/article/pii/S0360544219321772.

[35] Parul Yadav, Manish Gaur, Nishat Fatima, and Saqib Sarwar. Qualitative and quantitative evaluation of multivariate time-series synthetic data generated using

mts-tgan: A novel approach. *Applied Sciences*, 13(7), 2023. ISSN 2076-3417. doi: 10.3390/app13074136. URL https://www.mdpi.com/2076-3417/13/7/4136.

[36] Guotian Yang, Yingnan Wang, and Xinli Li. Prediction of the no emissions from thermal power plant using long-short term memory neural network. *Energy*, 192: 116597, 11 2019. doi: 10.1016/j.energy.2019.116597.

[37] Jinsung Yoon. Timegan: Codebase for time-series generative adversarial networks (timegan). https://github.com/jsyoon0823/TimeGAN, 2019. GitHub repository.

[38] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf.

[39] Ivan T. Ćirić, Žarko M. Ćojbašić, Vlastimir D. Nikolić, Predrag M. Živković, and Mladen A. Tomić. Air quality estimation by computational intelligence methodologies. *Thermal Science*, 16(suppl. 2):493–504, 2012.