IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 17, 2022, accepted May 16, 2022, date of publication May 25, 2022, date of current version June 22, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3177906

Assessing Deep Generative Models on Time Series Network Data

MUHAMMAD HARIS NAVEED[1], UMAIR SAJID HASHMI[1], (Member, IEEE), NAYAB TAJVED[1], NEHA SULTAN[1], AND ALI IMRAN[2], (Senior Member, IEEE)

[1]School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad 44000, Pakistan
[2]AI4Networks Research Center, School of Electrical and Computer Engineering, The University of Oklahoma, Norman, OK 73019, USA

Corresponding author: Umair Sajid Hashmi (umair.hashmi@seecs.edu.pk)

This work was supported in part by the National Science Foundation (NSF) under Grant 1619346 and Grant 1923669, and in part by the Qatar National Research Fund (QNRF) under Grant NPRP12-S 0311-190302.

ABSTRACT To achieve zero touch automation in next generation wireless networks through artificial intelligence (AI), large amounts of training data is required. This training data is publicly unavailable and is a major hindrance in research on AI applications to wireless communication. One solution is using limited real data to generate synthetic data that can be used in lieu of real data. Generative Adversarial Networks (GAN) have been used successfully for this purpose. In this paper, we choose two publicly available GAN - based models and one deep learning - based auto-regressive model. We then compare their performance at generating synthetic time-series wireless network traffic data. We also assess the impact of data scarcity on the generated data quality by varying the level of data available to the models for training. Moreover, in order to assess the usefulness of this generated data, we compare the performance of a gradient boosting regressor trained solely on generated data, real data, and a mix of both at forecasting network traffic. Our experiments show that the GANs perform better than the auto-regressive approach in each aspect considered in this work and forecasting models trained to predict network load based on data generated by these GANs yield error rates comparable to models trained on real data. Finally, augmenting small amounts of real data with generated data leads to minor performance gains in some cases.

INDEX TERMS Machine learning, GAN, TimeGAN, PAR, DoppleGANger, time series, forecast analysis.

I. INTRODUCTION

The application of artificial intelligence (AI) in medicine, power systems, image processing and other domains has become commonplace. Although motivated by gains and benefits of zero touch automation presented in earlier studies such as [1], the realization of AI enabled gains in wireless networks is yet to be witnessed in the real world. This is set to change as the world moves to sixth generation networks (6G) and beyond, in which networks will be able to perform self–configuration, self-optimization and self–healing via real-time AI on network parameters [2]. Paired with the rapid proliferation of new and diverse paradigms, such as AR (Augmented Reality) and VR (Virtual Reality), new connectivity use cases and applications such as IoT (Internet of Things) [3], URLLC (Ultra Reliable Low Latency Communication) [4], and holographic communications, the potential of harnessing the vast data produced by these systems in supervised machine learning (ML) problems to perform usage prediction, optimal resource allocation, anomaly detection and other such applications is substantial [5]–[7]. The ultimate desirable goal is to achieve AI enabled zero touch automation in next generation networks with the aim to minimize operational cost, overcome operational complexity and human errors, thereby maximizing resource efficiency and Quality of Experience (QoE).

Unfortunately, sufficient telecommunications network data required for executing sophisticated ML models is either not available, or is too scarce for effective ML model training and execution [8], [9]. This is largely due to privacy concerns and the hesitance of the telecom industry to open-source data that could potentially be used by their competition. Another challenge in getting ample training data is the large amount of technical effort required to get data out of silos within the operators where it remains trapped. When data is made available, it is usually locked behind non–disclosure agreements or released to specific research groups. This is a major impediment to research in this domain and partly responsible

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Moinul Hossain.

for the lag in applying ML techniques that we see in the communication systems domain compared to other domains where data is more freely available, such as image processing.

One solution to this deadlock is to generate synthetic data that is faithful to the properties of the original data, but is different from the original data in terms of actual values. This paper tests the latest deep learning based generative models or deep generative models (DGMs). While there are many types of DGMs, such as Auto-regressive models, Variational Autoencoders etc., here we focus on generative adversarial networks (GAN), owing to their significant success in this domain. GAN is a generative model that has delivered impressive results in generating synthetic but near realistic images and videos [10]. We apply two variants of the GAN model to three distinct internet activity time-series data sets and compare their performance against an auto-regressive method. We then test the performance of the generated data in downstream supervised machine learning applications, specifically forecasting internet traffic levels. We also analyze how, if at all, is the performance of these methods affected by the amount of data available. While the idea that more data equals better performance is valid, it is interesting to see on how much data can GANs give usable results. Finally, since evaluation of GAN performance is an open problem, and the metrics that do exist are geared towards evaluating image output, we use a variety of direct and indirect metrics to comprehensively evaluate the quality of the generated synthetic data. Simply put, we assess how data sparsity impacts several deep generative models' ability to produce high quality synthetic time-series data, and then assess how using this synthetic data improves or degrades performance of a forecasting model on real, unknown test data.

With this study, we offer the following contributions:

- A performance analysis of publicly available deep generative models, with particular focus on GANs, designed to generate realistic time-series data on a scarce telecommunications data set is conducted using a select group of indirect metrics, which assess the quality, fidelity and practical usefulness of the generated time-series data. We observe that the range of values and structure of a given time-series is retained in the GAN-generated series, but the same cannot be said for the auto-regressive model's generated time-series data.
- Analysis of the impact of data scarcity on the performance of these techniques in generating realistic time series synthetic data. While it seems intuitive that increasing the training data would improve the generated data quality, we observe that this is not always the case, and that longer time-series sequences with long-term trends are harder for GAN models to learn faithfully.
- Quantitative study of the utility of generated data in downstream predictive modelling using supervised ML approaches. We observe that for simple forecasting purposes, the error between the generated and real data trained model is between 1 to 4 percent for our best performing model.

The paper is organized as follows: Section II discusses relevant literature in this domain, Section III describes the data sets we used, Section IV describes different time-series generative models and the architectures we are using. Section V explains our methodology while Section VI explains the experimental setup employed in our simulations as well as the results. The paper is concluded in Section VII with a summary of the work and possible future research directions.

II. RELATED WORK

GANs were originally designed for simple image data, but since their inception have seen great advancements [10].

Although the extent of applicability of GANs in wireless communications domain is still being explored, some recent studies have investigated this particular research theme. To model the channel effects in an end-to-end wireless network in a data-driven way, [11] proposes to use a Conditional GAN (CGAN). A novel architecture using GAN in [12] is designed to directly learn the channel transition probability (CTP) from receiver observations. Authors in [13] leverage a GAN to create large amounts of synthetic call data records (CDRs) containing the start hour and duration of a call. The authors show a marked improvement in future call duration prediction accuracy using real data augmented with GAN generated synthetic data points.

There have also been GAN-based approaches proposed for generating different types of time-series data. Navid Fekri *et al.* introduces a Recurrent GAN (R-GAN) [14] for generating realistic energy consumption data by learning from real data. Hyland *et al.* proposed a Recurrent Conditional GAN (RCGAN) [15] to produce realistic real valued multi-dimensional medical time series data. Olof Mogren proposes a continuous recurrent neural network (C-RNN) based GAN model [16] that works on sequential data to synthesize music using a long short-term memory (LSTM) NN for both generator and discriminator.

Many studies have tried to solve the training data scarcity problem using GANs. Changhee Han *et al.* [17] introduced data augmentation in medical images using GANs. They concluded that data augmentation can boost diagnosis accuracy by 10%. Similar work has been done on images of skin lesions [18] and brain tumor MRIs [19]. Similarly, SimGAN [20] shows a 21% performance improvement in eye-gaze estimation by using an architecture similar to GANs.

Several evaluation techniques have been proposed for data generated by GANs. Ali Borji [21] analyzes more than 24 quantitative and 5 qualitative measures for evaluating generative models. He concludes that each have their own strengths and limitations. He also proposes that the evaluation techniques should be application specific. The most common GAN metrics are Frechet Distance [22] and inception score [23], but many more have been proposed, such as the fidelity and diversity generative model metrics explained in [24]. All in all, as noted in [25], GAN performance evaluation remains an open and challenging research problem.

To the best of our knowledge, our work is the first to compare several modern GAN models vs an auto-regressive approach as well as the first to stress test the given models with scarce data. It is also one of the few papers that looks at telecommunications data instead of energy, financial or medical time-series data.

III. TELECOM ITALIA DATASET

A. BACKGROUND

As mentioned before, telecommunications data is rarely open-sourced or easily available. When it is available, it is limited to a few research teams that have signed Non - Disclosure Agreements with service providers. Telecom Italia (renamed TIM Group in 2019) recognized that this situation hampered independent researchers who couldn't access data for analysis and model training purposes. So, in 2014, it organized the 'Telecom Italia Big Data Challenge'. Participants were given access to telecommunications, weather, social media, news and electricity consumption data of Milan and the province of Trento from November 2013 to January 2014, with the non - telecommunication data sets provided by other industrial partners [8].

We will be focusing on the telecommunication activity data provided in this set, specifically the call detail records (CDRs), which contain time-series data on internet usage in all regions of the Milan Metropolitan area from November to December 2013. The time range considered in our work is from Monday, November 4, 2013, to Sunday, December 22, 2013. Since the overall challenge made use of data from multiple domains provided by different companies, all of which used different spatial and temporal aggregation methods, Telecom Italia standardised them all onto a single 100×100 grid for Milan, with each square covering about 235×235 metres of area. The internet usage is measured in 'number of connections created' in 10-minute intervals. However, for ease of analysis and training we down-sample these measurements to an hourly interval.

B. REGIONS

In order to judge how well our selected generation methods work for time-series data of different nature, we look at data from three regions with interesting activity patterns. By 'interesting', we mean that the network activity from these regions contains daily and weekly seasonality as well as an overall trend over the selected timespan. These selected regions are **Bocconi university**, a private university, the **Navigli district**, a wedge between two canals that boasts upscale restaurants, bars, art galleries and is popular with tourists. The third region is the **Duomo cathedral** and its surroundings, which is a religious and tourist hotspot.

Fig. 1 and 2 illustrate the coverage area and the geographical placement of the selected regions respectively. Fig. 3 illustrates the activity patterns of each region over seven weeks. We can observe a daily seasonality in Bocconi, with usage peaking around midday and almost disappearing at night. There is an expected decline in internet usage on the weekends, which is due to low activity levels in the University campus. In Navigli, we do not observe a dramatic decrease in activity on any particular day, but we observe two peaks per

9901	9902	···	9999	10000
9801	···	···	9899	9900
···	···	···	···	···
101	102	···	···	200
1	2	3	···	100

FIGURE 1. A sample figure taken from the telecom Italia dataset [8] which divides the coverage area of radio base station into grids which helps in identifying the geographical location of the user.

FIGURE 2. Administrative map of Milan with our selected areas highlighted.

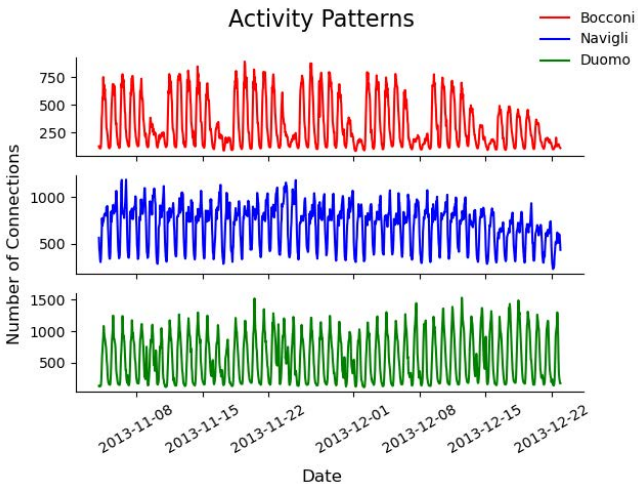

FIGURE 3. Internet activity over seven weeks in our selected areas. Note the differences in trend and seasonality over time across all three localities.

day, one at midday, and a higher one around midnight, indicating the district is a nightlife hub. Lastly, Duomo resembles a sinusoidal pattern, but a closer look shows that there's a slight uptick in activity every Friday and Saturday towards twilight. Thus, all three regions are distinct time-series data sets and it will be interesting to see whether the performance of GANs will differ due to this fact. This is also important since time-series data can possess many different trends and patterns thus, any generative models must be able to perform well across the board in order to be useful.

IV. TIME SERIES GENERATIVE MODELS

We now provide an overview of the structure of time-series data and the various methods used to generate them, including a more in-depth explanation of the specific techniques that we're using in this work. A time series is an ordered sequence of values of a variable taken at equally spaced time intervals. Thus, any analysis or generation of time series must take into account that data points taken over time may have an internal seasonality and trend [26]. Mathematically, most time series' can be written as:

$$x_t = s_t + g_t + e_t \tag{1}$$

where s_t = seasonality, g_t = trend, and e_t = residuals.

Here, $t = 1, 2, 3, \ldots, N$ represents the time index at which observations have been recorded.

A. DECOMPOSITION - BASED METHODS

As shown above, a time-series generally comprises of seasonality, trend and residual terms. One method of time series generation involves decomposing a time series into its components, then adding a deterministic and stochastic component which is constructed by optimizing weights for the trend and seasonality components and by modelling the residuals via some statistical model [27]. Another approach uses bootstrapping on the residuals obtained after decomposing to create augmented signals and then combines them with trend and seasonality to create a new time series [28].

B. AUTO-REGRESSIVE MODELS

Auto-regressive models try to forecast future values of a series based on past values of the same series and a stochastic term. In the simplest auto-regressive generative model, the conditional distributions $p(x_i|x_{<i})$ correspond to a Bernoulli random variable and the model learns a function that maps preceding variables $x_1, x_2, \ldots x_{i-1}$ to the mean of this distribution resulting in (2) from [29]:

$$p_{q_i}(x_i|x_{<i}) = Bern(f_i(x_1, x_2 \ldots x_{i-1})). \tag{2}$$

New data can then be generated by sampling from the conditional distribution learnt by the model. The use of the Bernoulli distribution means that this model cannot learn all types of distributions. This weakness is corrected by models like Neural Auto-regressive Distribution Estimator (NADE) [30], that use neural networks for parameter estimation, which is more efficient than the simple approach described earlier.

1) PROBABILISTIC AUTO-REGRESSIVE MODEL (PAR)

One implementation of an auto-regressive generative model is PAR, which is a model in the Synthetic Data Vault collection of generative models [31]. PAR uses recurrent neural networks to model the distributions. Since RNNs are designed to handle sequential data, it is much more suited to our task of time series generation.

Given a function h employed within an RNN, we calculate its hidden states $h_i = h(h_{i-1}, p_{i-1}, \theta)$ for each time-instance, depending on the prior corresponding hidden state h_{i-1}, preceding input value p_{i-1} and the network hyper parameters θ. These three variables establishing the hidden state h_i construct a set of parameters $\theta(h_i)$ which represent a distribution with density $\theta(h_i)(p_i)$, resulting in the following conditional distribution:

$$q_\theta[p_{t_0:T}|p_{1:t_0-1}] = \prod_{i=t_0}^{T} \ell_{\psi_{h_i}}(p_i) \tag{3}$$

where $q_\theta[p_{t_0:T}|p_{1:t_0-1}]$ is a parametric distribution specified by learn-able parameters θ, which are obtained by using past values $p_{1:t_0-1}$ from a time series $p = (p_t)$; $1 \leq t_0 \leq T$; to forecast future values $p_{t_0:T}$ This conditional distribution can then be used to generate new synthetic data similar to the original input data. We will be evaluating PAR against our chosen GAN - based models to evaluate whether the GAN framework yields any noticeable performance improvements against this simpler deep learning-based model. A more comprehensive review of time series data generation techniques, specifically for data augmentation in time series classification and clustering tasks, can be found in [32].

C. GAN - BASED METHODS

We now move to our main focus, the GAN-based models, in this section. We will give a brief overview of the GAN design, its structure, and its potential issues. First introduced in 2014 [10], GANs consist of two competing agents, typically neural networks, referred to as the generator and discriminator. The generator network attempts to model a noise vector z to fit the probability distribution of the input data, whereas the discriminator attempts to accurately classify the generated data from the real data. Loss convergence of generator and discriminator terminates the training period. Essentially, the two networks are jointly involved in a 2 – player min-max game up until the discriminator fails to distinguish between the real data and the generated data, a point denoted by attainment of Nash equilibrium [33].

Mathematically, this process is expressed in [10] as:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data(x)}}[\log(D(x)] + \mathbb{E}_{z \sim p_z(z)}[1 - \log(D(G(z))] \tag{4}$$

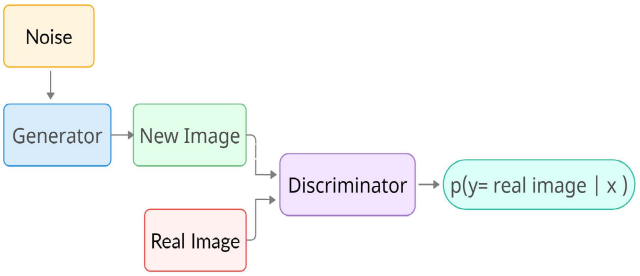

FIGURE 4. Simple GAN structure.

Here x is the input data, $log(D(x))$ is the predicted output of the discriminator for x_i, $log(D(G(z))$ is the output of the discriminator on the GAN generated data $G(z)$. The aim is to maximize the ability of the discriminator to identify real data from generator produced data, so we maximize this value, whereas the generator part of the equation tries to minimize the discriminator's ability to correctly classify real and fake data. This translates to maximizing the first term and minimizing the second term of (4). The basic GAN structure is illustrated in Fig. 4.

GANs initially became popular for their ability to produce high-quality image data while avoiding the problems associated with using Markov chains or approximating unsolvable likelihood functions. This is achieved by training the GANs via backpropagation and using dropout regularization. The drawback is that GANs are often hard to train, and suffer from problems like overfitting (reproducing input data), mode collapse (generating samples from only one class in the data) and training instability.

The use of GANs to generate tabular or time-series data is less common than generating images or video. One model used to generate tabular data is the CTGAN (Conditional Tabular GAN) [34]. Whereas, the TimeGAN [35] and DoppelGANger [36] are frameworks that modify the traditional GAN architecture to make it more suitable for time-series data. These last two models are what we use in this work. A brief overview of them is given below.

1) TIMEGAN

The auto-regressive models mentioned above are good for capturing the temporal dynamics of a sequence, but are deterministic in nature as opposed to generative. Conversely, GAN architectures such as the RGAN used in [15] do not really take into account the inter-row dependencies of time series data. The TimeGAN solves this problem by combining the GAN framework with an auto-regressive setup. Both are trained jointly with the unsupervised GAN loss guided by the supervised auto-regressive loss. Additionally, the model makes use of an embedding network to map high level features to a low-level latent feature space. The generator network also first produces samples in the latent space which are then converted back to the original feature space via a recovery network. This is done to reduce high dimensionality

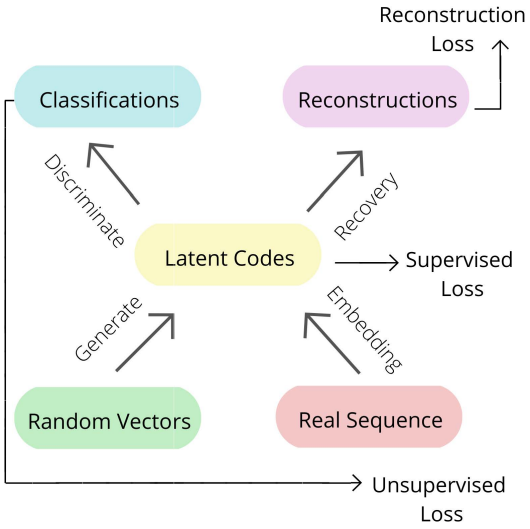

FIGURE 5. Simplified block diagram of TimeGAN.

in the adversarial learning space. A high-level structure of the TimeGAN is shown in Fig. 5.

The recovery and embedding parts are trained via a supervised and reconstruction loss. The reconstruction loss ensures the learnt latent representation is correct and the supervised loss aids the generator in learning the temporal dynamics of the data. The generator and discriminator are then trained in a typical adversarial fashion. The losses used to train the Embedding, Recovery, Generator and Discriminator networks can be found as Eqs. (7), (8) and (9) in [35].

2) DOPPELGANGER

The DoppelGANger introduces several new ideas to solve typical GAN problems like overfitting as well as problems faced when generating longer, more complex time-series data [36]. One change in design that is most relevant to our work, is how much of the data is generated in a single instance. Since RNNs produce a single measurement in a single pass and for a time-series of length L perform L passes, they tend to forget prior values and struggle to capture long term correlations in a time-series. Furthermore, authors in [36], in Section 4.1, page 5 state that even LSTMs, which were designed to correct the above-mentioned problem with RNNs, empirically struggle to perform well when the length of a time-series surpasses a few hundred records.

DoppelGANger solves this by modifying the RNN structure to produce S values in a single pass. This reduces the overall number of passes required to generate the entire series, but the quality of the generated samples also deteriorates as S increases. It uses the Wasserstein Loss as opposed to the regular GAN loss function since the former leads to more stable training in this case. The optimization function for this GAN architecture may be expressed as:

$$\min_{G} \max_{D_1, D_2} L_1(D_1, G) + L_2(D_2, G) \tag{5}$$

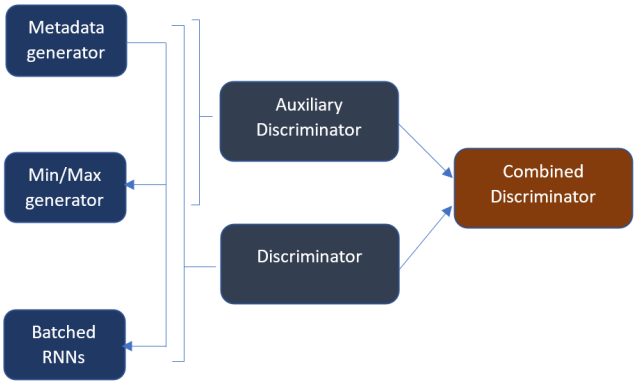

FIGURE 6. Simplified block diagram of DoppelGANger.

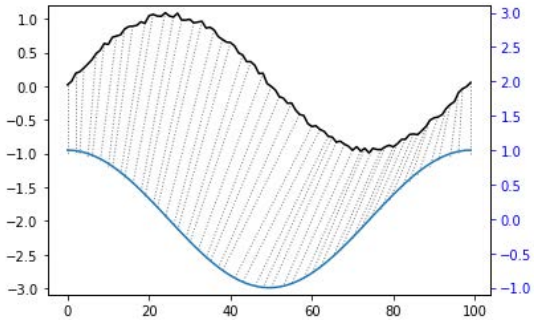

FIGURE 7. Aligning two dummy series in DTW [38].

where L_i for i = 1, 2 is the Wasserstein Loss, described in [36] as:

$$L_i = \mathbb{E}_{x\sim p_x}[T_i(D_i(x)] - \mathbb{E}_{z\sim p_z}[D_i(T_i(G_z))] - \lambda\mathbb{E}_{\hat{x}\sim \hat{p_x}}[(||\nabla_{\hat{x}} D_i(T_i(\hat{x})||_2) - 1)^2] \quad (6)$$

where $T_1(x) = x$, $T_2(x) = tx + (1 - t)(G(z))$ and t is a value from the uniform distribution.

A basic block diagram for a DG is illustrated in Fig. 6. A more detailed structure with explanation can be found in Fig.7 of [36].

V. EVALUATION METRICS

Evaluating the quality of GAN generated data is an open research problem. Unlike discriminative models which can be evaluated on fairly robust metrics like accuracy, precision and F1 score among others, generative models have no such counterparts. In the case of images, visual inspection is relied on to determine whether the produced image is good quality. This is not feasible with tabular forms of data. Another method is to indirectly evaluate the quality of the generated data by seeing how well it performs when substituted in place of real data in supervised tasks such as classification and forecasting.

Study in [21] provides a comprehensive survey of potential metrics that can be used to evaluate GANs, most of our chosen metrics indirectly evaluate the quality of the generated data either via use in forecasting models, qualitative assessments, or quantitative measures such as distance. Our criteria for choosing evaluation metrics is based on the following three principles:

- The metrics should favor the generated data that is most similar to the original data.
- The metrics should reward models that generate diverse examples and are not prone to common GAN problems such as overfitting, mode collapse, and mode drop.
- The metrics should be computationally inexpensive and as simple to interpret as possible.

A. QUALITATIVE ASSESSMENT

Perhaps the simplest way to determine how similar our data is to the original, true data is to simply visualize it. While this approach does leave out hard numbers, it gives us a quick high-level view of the shape and spread of the generated data distribution. In this work, we will analyze histograms and auto-correlation function (ACF) plots for this purpose. The histograms will allow us to judge whether the GANs produce data that faithfully captures the range of values present in the original data as well as its distribution. The ACF plots are based on calculating the correlation of a time-series with itself at different, equidistant points in time (referred to as lags). In our case, we choose a lag of up to 168 since we have hourly data over multiple weeks and expect network activity patterns to repeat. In general, our aim is to see how similar the ACF plot and histograms of the generated data are to real data.

B. KULLBACK-LEIBLER DIVERGENCE

The Kullbeck-Liebler Divergence (or KL - Divergence, or KLD) measures the number of extra bits needed to represent a true distribution P with a code written to represent distribution Q which is an approximation of P. Thus, KL - Divergence can be interpreted as the inefficiency caused by using an approximate distribution Q rather than P. Note that this does not mean that KLD is a distance measure; it is not since it is asymmetric and does not obey the triangular inequality. While the use of KL - Divergence is uncommon in comparing time-series data, we use it to evaluate the distributions of the two data sets rather than their relationship in time. This can be done by calculating the KLD after discretizing the continuous time-series and using the bin counts to create probability distributions.

The mathematical form of the KLD is shown below:

$$D_{KL}\ (P\ ||\ Q)\ = \sum_{x\in X} P(x)\ log\frac{P(x)}{Q(x)}. \quad (7)$$

P(x) represents our true distribution (the real data), whereas Q(x) represents the approximate distribution (the generated data). Since the metric is asymmetric, the position of the two series in the formula is important and interchanging the position changes the results.

C. DYNAMIC TIME WARPING (DTW)

First introduced in 1978, Dynamic Time Warping is a class of algorithms that can be used to compare two ordered

sequences against each other [37]. These could be speech sequences, music or any other time ordered sequence. It was originally used in spoken word recognition since it could align the time axis of two sequences, say, the words *now* and *noow* and calculate the Euclidean distance between them. This is opposed to directly using a distance metric which would give a large value in the comparison of any two such sequences, even though they are identical. In this work, we employ an R implementation of the algorithm [38].

In mathematical terms, DTW finds a warping function $\phi(k)$, described in [38] as:

$$\phi(k) = (\phi_x(k), \phi_y(k)), \tag{8}$$

where ϕ_x and ϕ_y remap the time indexes of the reference series x and test series y. Given these warping functions, we find the average distance between these warped x and y series. The aim of the algorithm is to align the two series in such as a way so as to reduce the distance between the series' as much as possible. Thus, the optimization problem is given in [38] as:

$$D(x, y) = min_{\phi} d_{\phi}(x, y). \tag{9}$$

The left-over distance is the inherent difference between two sequences. Fig. 7 illustrates how the DTW algorithm aligns two time-series sequences.

D. TRAIN SYNTHETIC TEST REAL (TSTR) AND DATA AUGMENTATION

TSTR is an indirect evaluation technique where a predictive model is trained on synthetic data and verified on real data. The dataset generated by GAN is used for training a model which is then tested on examples from real dataset. It was proposed by [15]. We use this technique to evaluate the telecommunications data generated by our selected models using a simple Gradient Boosting Regressor. Firstly, we partition the original dataset into a train and test set. The model is trained on the training set and tested on the held-out test set. This process is Train on Real, Test on Real (TRTR). The test set produced during TRTR is passed to the same model trained on synthetic data. The model's performance is assessed using mean absolute percentage error (MAPE) which has the following mathematical representation:

$$MAPE\ (y, \hat{y}) = \frac{100}{n} \sum_{i=1}^{n} \frac{|y - \hat{y}|}{y}. \tag{10}$$

Here, y and $\hat{y}$ represent the true and forecasted data points, respectively.

Additionally, we also augment a small amount of real data with different amounts of synthetic data to study any improvements in forecasting accuracy. The total data is fixed at 5 weeks, but the number of real and synthetic weeks in the data changes. The forecasted values are then compared against the real values to calculate the MAPE. Following this TSTR pipeline, illustrated in Fig. 9, allows us to compare how much prediction accuracy is affected by replacing the real data with synthetic data in a downstream application such as network load forecasting.

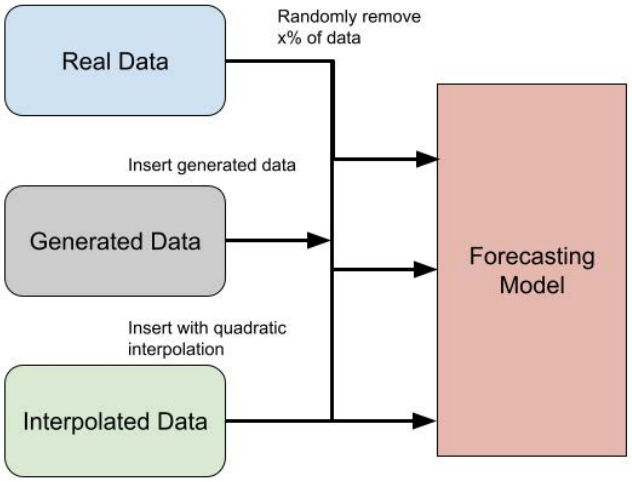

FIGURE 8. Block diagram of the imputation utility process.

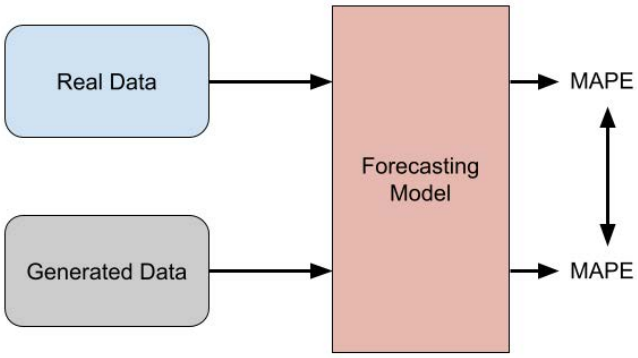

FIGURE 9. Block diagram of the TSTR process.

E. IMPUTATION UTILITY

In machine learning problems, missing data poses a serious threat to the learning ability of the model. It can introduce bias, make data handling onerous, and reduce efficiency. Missing data is often filled in, or imputed, using a variety of statistical and ML - based techniques. These include: i) carrying the last available reading forward, ii) filling with mean values, iii) imputation via K-nearest neighbours and iv) interpolation.

In this work, to introduce scarcity, we delete data from the sequence randomly. After we have obtained our GAN generated data, we impute the missing data using the corresponding sample points from our synthetic data. Similarly, we fill the same points via quadratic interpolation. Finally, we pass the sequence with missing values, the sequence with imputed GAN generated values, and the sequence with interpolated values to the same model and assess how the forecasting accuracy is affected. Fig. 8 shows the imputation utility methodology. The $\%x$ denotes the amount of data removed, which takes on the value of 20, 40, 60 and 80 percent.

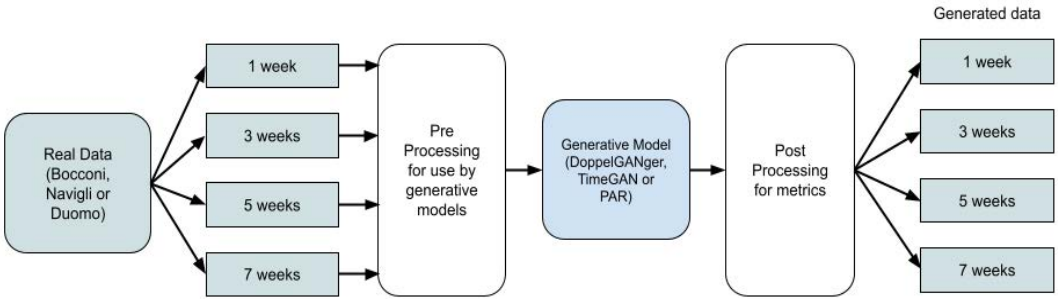

FIGURE 10. Block diagram depicting our data division and generation process.

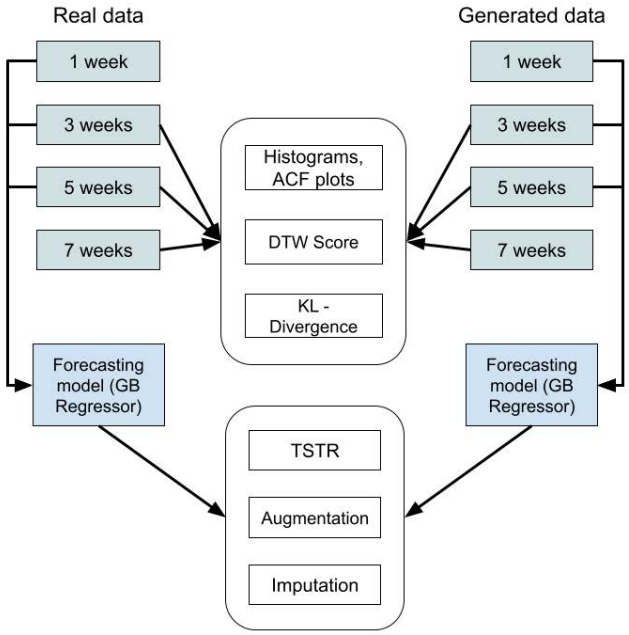

FIGURE 11. Block diagram depicting our comparative methods and outlining which techniques use what amount of data.

VI. RESULTS AND DISCUSSION

A. PRE-PROCESSING & SETUP

Figures 10 and 11 show our data generation pipeline and how real and generated data is compared with our comparative methods respectively.

We break each regions' data into four segments and one test week, details of which are given in Table 1. The test week in Table 1 is the week we forecast for. For conciseness, we report the qualitative assessment, KL - Divergence and Dynamic Time Warping of 3, 5 and 7 weeks of data only.

For the Train Synthetic Test Real (TSTR), augmentation and imputation assessments, a forecasting model is trained on 1, 3 and 5 weeks of data only and then used to forecast over the test week, which is chronologically the 6th week in the time-series. If we use 7 weeks of data as well, that would require making the test week the 8th week. The problem is that the 8th week lies in the Christmas season and its readings

TABLE 1. Data organization scheme.'Weeks' represents the number of weeks of data that is being used. 'Timespan' shows the date range and 'length' is the number of recordings in the data.

Weeks	timespan	length
1	11/4/2013 - 11/10/2013	168
3	11/4/2013 - 11/24/2013	504
5	11/4/2013 - 12/08/2013	840
7	11/4/2013 - 12/22/2013	1176
test week	12/9/2013 - 12/15/2013	168

are radically different from our training data. Thus, using 7 weeks of data is not appropriate in these assessments.

To perform the forecasting, we avoid statistical models like auto-regressive integrated moving average (ARIMA) or error trend and seasonality (ETS) since they require significant tuning with domain knowledge of the data set. Also LSTMs or other deep learning based architectures are not employed owing to a lack of training data. Instead, a simple gradient boosting regressor provided by scikit-learn [39] is used in this work. It is possible to use Tree - based models for this purpose, as demonstrated in [40]. In order to train the model, we extract several features of the time-series by hand, these are briefly explained below:

- **T-1:** The time-series with lag 1.
- **Hours:** The hour at which the sample is taken.
- **1st difference:** Difference between consecutive time-series values.
- **2nd difference:** Difference between consecutive values of the 1st difference.

Thus, by using the features above as inputs and training to predict the difference, we are able to 'tabularize' the time-series data and use it to train a decision tree - based algorithm to make forecasts. The model uses 100 estimators, 0.05 learning rate, and max tree depth of 12.

B. MODEL TRAINING

1) DOPPELGANGER

Detailed instructions on how to prepare data for use with the DoppelGANger can be found on the GitHub repository of [36]. In this section, we will only share the details relevant to reproducing our results.

The DoppelGANger is designed to work with time-series that also have corresponding static features/metadata (like

TABLE 2. An example of the required structure for three weeks of data before it is passed to the DoppelGANger.

Week of year	0	1	2	3	...	168
45	100	150	200	250	...	400
46	10	15	20	25	...	40
47	1	5	2	4	...	6

TABLE 3. Hyper-parameters used for each data set for the DoppelGANger.

Hyper-parameter	Bocconi	Navigli	Duomo
epochs	10,000	2,000	2,000
aux discriminator	False	False	False
self - norm	False	False	False
sequence length	24	12	12

ISP provider, area etc.) However, the data sets we're working with are univariate time-series with no metadata.

To solve this, we create a dummy metadata variable based on the week of year corresponding to our data. Since our data has hourly resolution, there are 168 readings in each week. This dummy metadata is then scaled between 0 and 1. Note that it is also possible to use a constant value in the place of metadata as well.

Table 2 illustrates how the data should be structured when using three weeks of data. The same can be extended to five and seven weeks as well. Table 3 lists the hyper-parameters used in our work. Any other hyper-parameters not mentioned here used default values.

2) TIMEGAN

As in the case of DoppelGANger, instructions on how to use TimeGAN can be found in the GitHub repository corresponding to [35]. The code for the TimeGAN does not provide a way to get the generated time-series with timestamps and corresponding values. Therefore, there are certain modifications required before it can be used. These changes are listed below:

- The input data cannot contain timestamps. Thus, in order to recreate timestamps for the generated data, pass 'day of year' and 'Hour' as features along with the internet connections value. If the data spans multiple years, then a 'year' feature will also be included.
- The TimeGAN outputs generated data in normalized form. Thus, the normalization function in the code needs to be modified to return the maximum value and difference between the minimum and maximum values. These values can then be used to denormalize the generated data.
- The generated data is in the form of a 3D array in the shape of (Number of samples, sequence length, Number of features). In order to recreate the generated data in the form of the original data, iterate through each sample, denormalize it, and then append all the samples together.

We observed that the same parameters worked well for all three data sets in the case of the TimeGAN. Please note that

TABLE 4. Hyper-parameters used for the TimeGAN, DoppelGANger and PAR.

Hyper-parameter	Value
sequence length	10,000
module	GRU
hidden dimensions	28
number of layers	3
iterations	50,000
batch size	64

TABLE 5. Hyper-parameters used for the TimeGAN.

Hyper-parameter	Value
sequence length	10,000
module	GRU
hidden dimensions	28
number of layers	3
iterations	50,000
batch size	64

TABLE 6. Hyper-parameters used for PAR.

Hyper-parameter	Value
epochs	1,000
segment size	None

hyper-parameter 'sequence length' in Table 3 is not the same as 'sequence length' in Table 5.

3) PAR

PAR (Probabilistic Auto-Regressive) is part of the Synthetic Data Vault [31]. PAR is the easiest model to train out of all three used in this work. The input data is simply the internet usage values against the corresponding timestamps. Table 6 lists the hyper-parameter values we used.

We generate 1 sample from the trained PAR model. This generated data is identical to the input data but filled with synthetic values. More information on how PAR works can be found in the documentation of SDV's GitHub repository.

C. QUALITATIVE ANALYSIS

1) HISTOGRAMS

For Bocconi, a quick look at Fig. 12 shows the performance disparity between the three models. It is clear that both DoppelGANger (DG) and TimeGAN (tgan) perform well in terms of capturing the full range of values present in the original data. However, a closer look shows that the TimeGAN struggles to capture outlier values such as the values greater than 700. We see that the DoppelGANger does a slightly better job at capturing less frequent values. The PAR model misses large parts of the original distribution, and even generates negative values at 7 weeks of data.

In the case of Navigli, these observations are more marked in a similar analysis of the Navigli area, shown in Fig. 14. We observe an increasing overlap between the two distributions as the training data is increased. PAR performs better here too compared to Bocconi, although it still doesn't

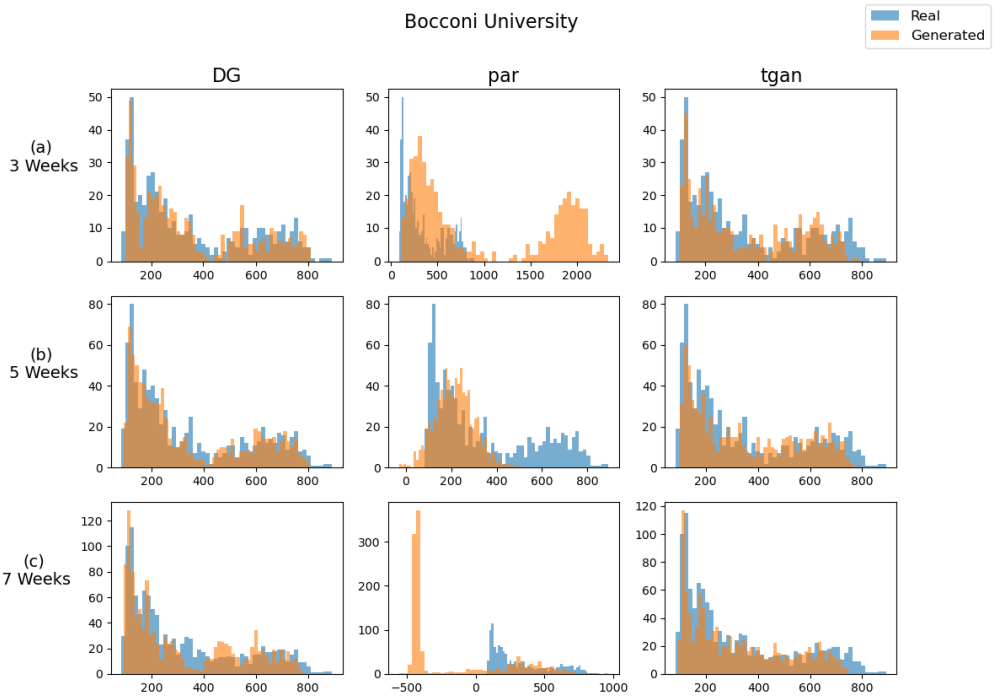

FIGURE 12. (a)–(c) show the distribution of values in the generated data when the models were trained on 3, 5 and 7 weeks of real data respectively, for Bocconi. Note that DG and tgan are short for DoppelGANger and TimeGAN respectively.

capture the multi-modal distribution properly, it doesn't output any negative values either.

In Duomo in Fig. 16 both DoppelGANger and TimeGAN perform really well. PAR performs decently when trained and compared with three weeks of data, but performance quickly deteriorates; on five weeks of data, it produces negative values and at seven weeks of data it underestimates the right-skew of the original distribution.

In short then, both TimeGAN and DoppelGANger generated data lies between the range of the original input data, whereas PAR's performance fluctuates significantly and its generated data contains negative values that weren't present or even possible in the input data.

2) ACF PLOTS

While the histograms give an idea about the distribution of the values of a time-series, they do not provide any information on how well the generated series captures the temporal correlation of the original series. For this, we create a series of auto-correlation function (ACF) plots in a similar fashion to the histograms. Fig. 13 shows the ACF plots for Bocconi region. The plots depict slightly better performance of the DoppelGANger than the TimeGAN in capturing the extreme correlation peaks. However, PAR fails to capture the temporal dependence between the series.

The same idea is repeated in the Navigli district in Fig. 15, where the performance of the TimeGAN deteriorates compared to that of DoppelGANger, while PAR again struggles with capturing temporal dynamics of the input time-series. We observe an improvement in how well DoppelGANger captures the negative correlation peaks as we increase the training data but the opposite applies in case of the TimeGAN. Compare that to Fig. 17 that shows the ACF plots for Duomo, and where the plots for DoppelGANger and TimeGAN are nearly identical. Thus, it seems that the models struggle to capture the correlations across a more chaotic time-series like Navigli but easily reproduce the correlations in places like Duomo and Bocconi, which have more regularly repeating activity patterns.

D. QUANTITATIVE ANALYSIS

Now that we have a cursory idea of the quality of the generated data, we will quantify its usefulness through some direct and indirect metrics which have already been explained in the preceding sections.

1) DYNAMIC TIME WARPING

The DTW results in Table 7 show several interesting things and corroborate some of the observations made in the prior section. In all three regions, PAR's generated series are extremely different from the ground truth. While looking at Bocconi, we see that the DTW score increases as the generated series get longer. This may be explained by Fig. 18, where we see that the real data decreases in amplitude towards the end. This trend is not captured by DoppelGANger but is somewhat reflected in the TimeGAN's output leading to an improved DTW score for the TimeGAN on 7 weeks of data and the opposite for the DoppelGANger. While not shown

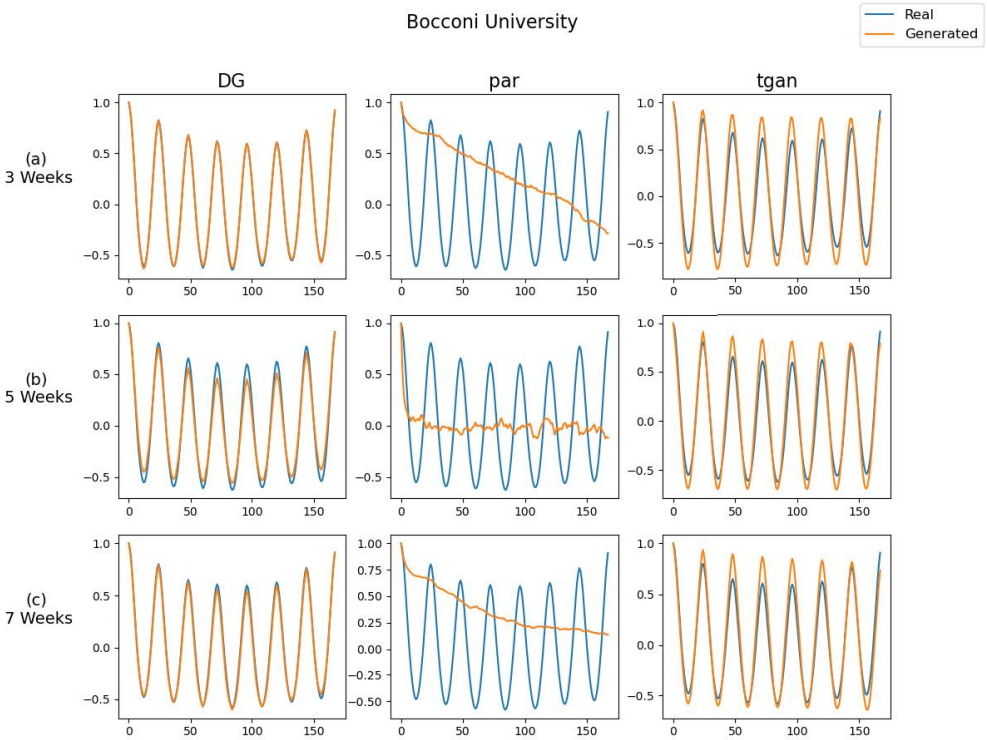

FIGURE 13. (a)–(c) show the ACF plots of real vs. generated data when the models were trained on 3, 5 and 7 weeks of real data respectively, for Bocconi. Note that DG and tgan are short for DoppelGANger and TimeGAN respectively.

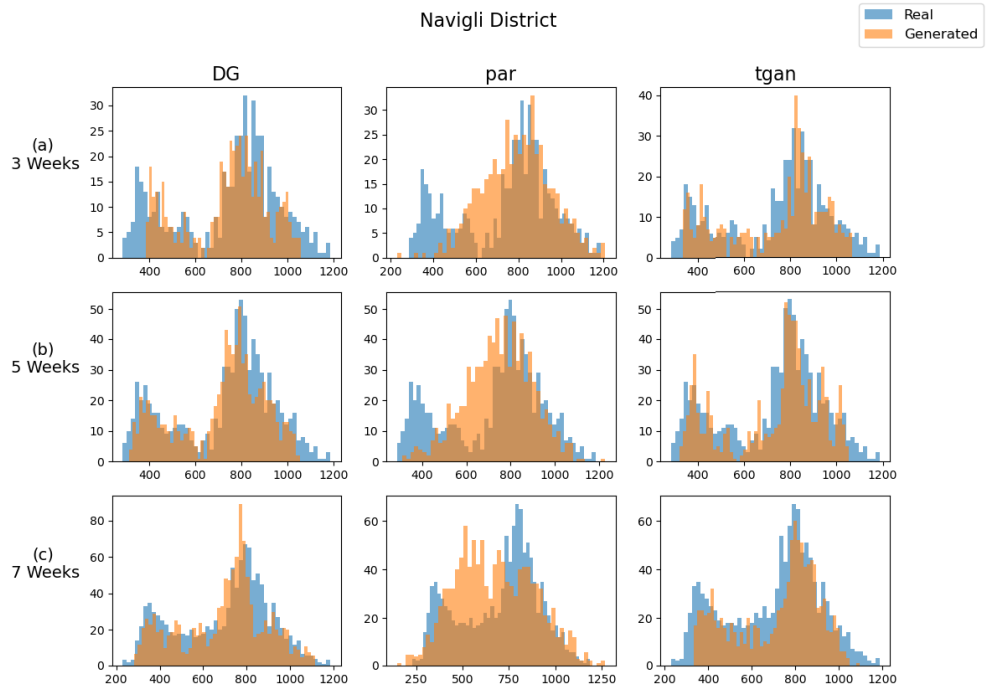

FIGURE 14. (a)–(c) show the distribution of values in the generated data when the models were trained on 3, 5 and 7 weeks of real data respectively, for Navigli. Note that DG and tgan are short for DoppelGANger and TimeGAN respectively.

here, the same explanation also applies to Navigli. Duomo on the other hand, features no such long-term downtrend, which is why we see a decrease in DTW scores as the amount of available data is increased.

2) KL - DIVERGENCE

The KL - Divergence calculations for all three regions are shown in Table 8. There is no significant pattern to be seen, other than that differences for DoppelGANger

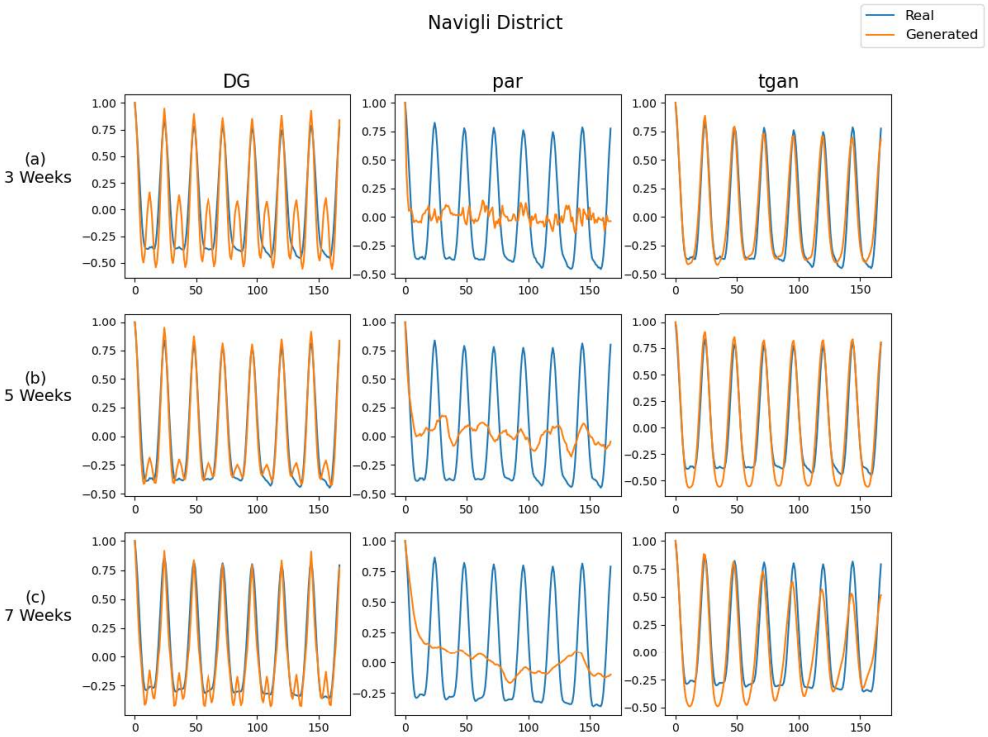

FIGURE 15. (a)–(c) show the ACF plots of real vs generated data when the models were trained on 3, 5 and 7 weeks of real data respectively, for Navigli. Note that DG and tgan are short for DoppelGANger and TimeGAN respectively.

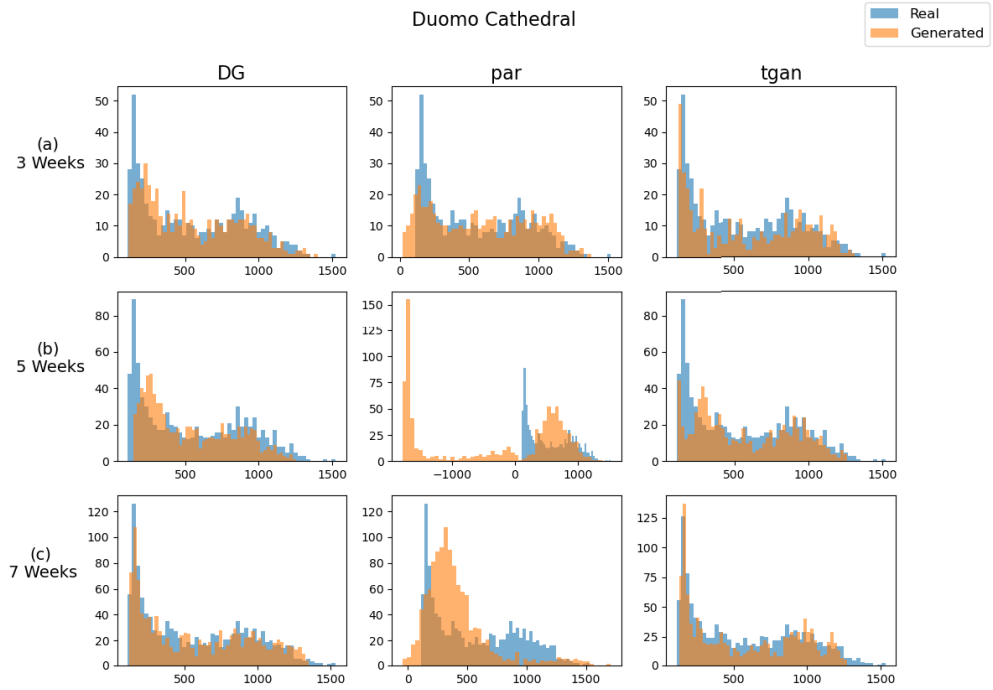

FIGURE 16. (a)–(c) show the distribution of values in the generated data when the models were trained on 3, 5 and 7 weeks of real data respectively, for Duomo. Note that DG and tgan are short for DoppelGANger and TimeGAN respectively.

and TimeGAN tend to stay consistent, whereas PAR generated data outputs the largest values and demonstrates the most volatility. In general, from an information theory perspective, we would not need a great many more bits to represent the generated distribution as the real distribution.

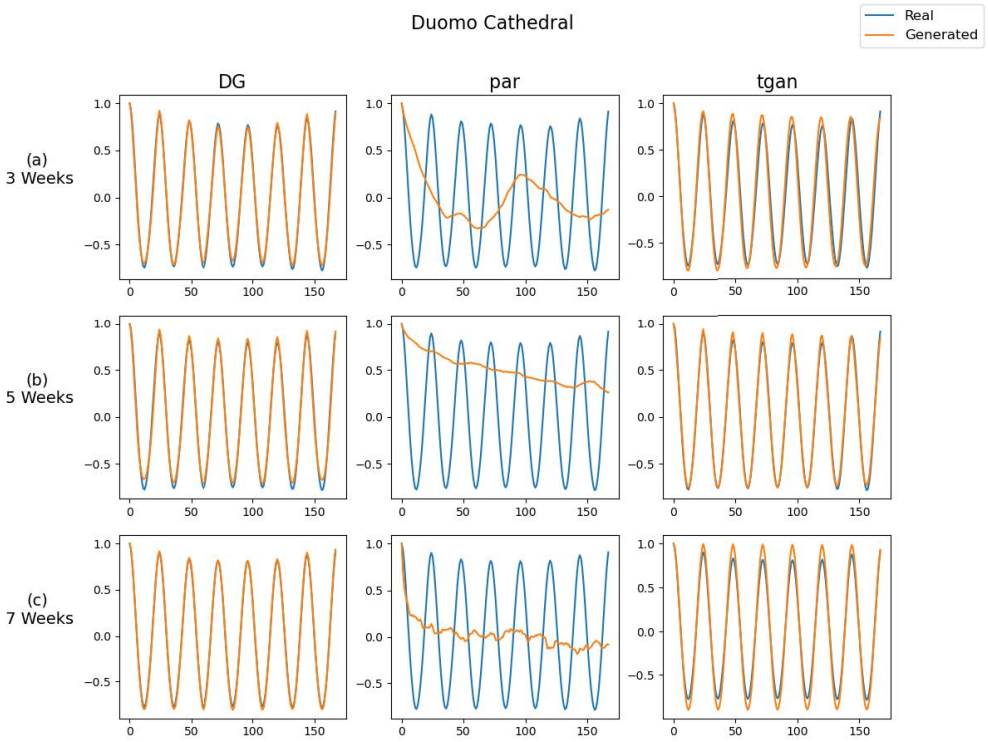

FIGURE 17. (a)–(c) show the ACF plots of real vs generated data when the models were trained on 3, 5 and 7 weeks of real data respectively, for Duomo. Note that DG and tgan are short for DoppelGANger and TimeGAN respectively.

TABLE 7. DTW - based similarity scores (lower the better) for Bocconi, Navigli and Duomo. 'Weeks' indicates the amount of data used in the calculations in terms of number of weeks.

	Bocconi			Navigli			Duomo		
Weeks	DoppelGANger	PAR	TimeGAN	DoppelGANger	PAR	TimeGAN	DoppelGANger	PAR	TimeGAN
3	19.28	260	30.65	35.2	67.93	32.37	42.38	95.74	47.52
5	21.89	67	36.7	31.58	53.04	34	38.73	459.25	48.4
7	25.28	238.54	31.47	40.19	54.91	34.26	36.14	109.4	36.8

TABLE 8. KL - divergence (lower the better) for Bocconi, Navigli and Duomo. 'Weeks' indicates the amount of data used in the calculations in terms of number of weeks.

	Bocconi			Navigli			Duomo		
Weeks	DoppelGANger	PAR	TimeGAN	DoppelGANger	PAR	TimeGAN	DoppelGANger	PAR	TimeGAN
3	0.01	0.08	0.02	0.006	0.12	0.009	0.02	0.1	0.049
5	0.01	0.13	0.032	0.014	0.10	0.004	0.05	0.15	0.06
7	0.03	0.22	0.034	0.034	0.04	0.017	0.05	0.24	0.05

TABLE 9. Train synthetic test real (TSTR) and train real test real (TRTR) prediction accuracies (in MAPE) Bocconi, Navigli and Duomo. 'Weeks' indicates the amount of data used in the calculations in terms of number of weeks. Note that DG and tgan are short for DoppelGANger and TimeGAN respectively.

	Bocconi				Navigli				Duomo			
Weeks	DG	tgan	PAR	TRTR	DG	tgan	PAR	TRTR	DG	tgan	PAR	TRTR
1	12.21	15.72	28.16	10.48	7.6	10.33	15.46	7.34	12.5	13.2	26.5	9.34
3	12.25	16.54	26.66	8.47	7.51	7.89	24.58	7.20	11.08	12.4	22.1	7.68
5	12.59	16.21	20.7	7.26	7.24	8.1	16.54	6.84	10.11	18.3	30.8	7.14

3) TRAIN SYNTHETIC TEST REAL (TSTR)

Looking at Table 9, we can make two observations. The first can be seen in the results for Bocconi where we see that increasing the amount of data leads to negligible changes in accuracy for both DoppelGANger and TimeGAN. PAR goes against this trend but we've already seen above that PAR's generated data is inconsistent so this is understandable. In any case, compared to the TRTR values, we see a 5% and 9% error difference between models trained on five weeks of data generated by TimeGAN and DoppelGANger respectively,

TABLE 10. Augmented data performance for Bocconi, Navigli and Duomo (For reference, TRTR for 5 weeks' real data is 7.26%, 6.84% and 7.14% for Bocconi, Navigli and Duomo respectively). All values are in MAPE. 'R + S (Weeks)' means real + synthetic number of weeks, so '1 + 4' means 1 real and 4 synthetic weeks of data.

	Bocconi			Navigli			Duomo		
R + S (Weeks)	DoppelGANger	TimeGAN	PAR	DoppelGANger	TimeGAN	PAR	DoppelGANger	TimeGAN	PAR
1 + 4	9.88	15.04	13.45	7.41	7.67	8.5	9.6	10.9	13.85
4 + 1	7.61	8.46	7.73	6.76	7.07	6.78	8.5	7.63	8.59
2 + 3	8.84	11.27	10.4	7.42	7.43	7.66	8.8	9	11.7
3 + 2	7.88	7.70	8.83	6.93	6.53	6.37	7.9	8.8	7.86

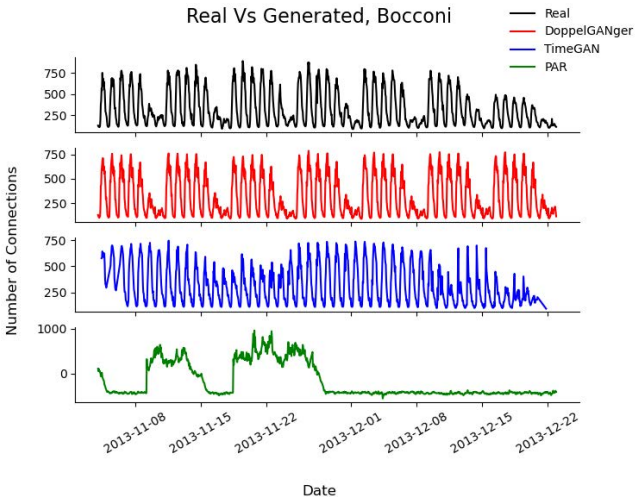

FIGURE 18. Generated data vs. real data, Bocconi, shown over 7 Weeks for all three models.

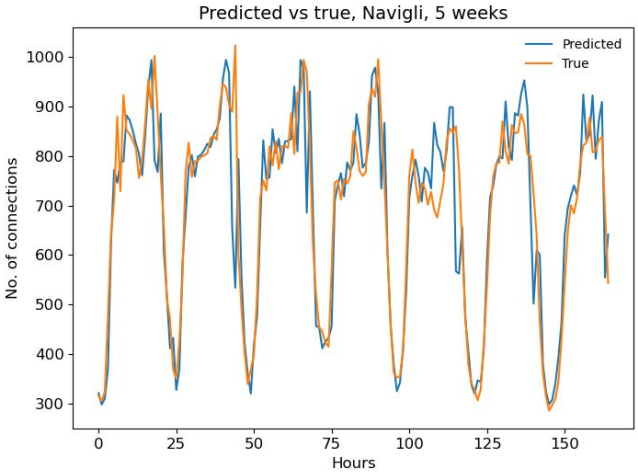

FIGURE 19. True vs. predicted, Navigli, trained on 5 weeks of TimeGAN data.

this shows that for forecasting a series' such as Bocconi, which has weekly and daily seasonality, the generated data is somewhat less useful. The advantage of the DoppelGANger over the TimeGAN comes because the TimeGAN cannot predict the troughs on the weekends well, largely because it fails to capture them during generation as well.

The situation in Navigli and Duomo is somewhat better. In the former the accuracy for DoppelGANger and TimeGAN improves as we increase the amount of generated data. A plot of the real vs predicted value is given in Figure 19 for TimeGAN. These accuracy values are also close to the accuracy of the real data, indicating that both can produce synthetic data comparable to the true data. In Duomo, this trend only applies to the DoppelGANger, with an error difference less than or equal to 3% between it and the true data.

Overall, the DoppelGANger generates significantly better quality data than the TimeGAN or PAR across all three regions.

4) PERFORMANCE WITH AUGMENTED DATA SET

Augmenting the real data with synthetic data yields results shown in Table 10. There are no across the board improvements, and the improvements observed are minimal. For instance, in Bocconi, adding 4 weeks of DoppelGANger synthetic data to 1 week of real data does improve performance over just using 1 week of real data as shown in Table 10, but the improvement is only 0.6%. In Navigli, adding two weeks from either PAR, DoppelGANger or TimeGAN to three weeks of real data yields improvement, but as before this improvement is small; less than 0.5%. In Duomo, we see no improvement in any combination of real and synthetic data at all.

The above results show that combining synthetic data with real data can yield performance benefits in a time-series forecasting framework. These improvements are minor, but that can be expected given that we are working with scarce data.

5) IMPUTATION UTILITY

Tables 11, 12 and 13 display the prediction errors for five models trained on mixed data from all three generative models, as well as interpolated data. For comparison, we also train a model on the smaller data set created by randomly removing observations. We see that the performance of all models deteriorate as we use them to fill larger gaps in the real data set. However, the DoppelGANger's performance appears the most stable, followed by the TimeGAN although it exhibits higher errors overall. We also observe the model trained on PAR imputed data performs worse than a model trained on true unimputed data, but its performance suddenly improves when the original data is reduced by 90%. We can say that PAR's uncharacteristically good performance in this

TABLE 11. Imputing missing data via interpolation and with generated data for Bocconi university. All values are MAPE. The total amount of data used is five weeks or 840 hourly readings.

Bocconi University					
%age data removed	DoppelGANger	TimeGAN	PAR	Missing	Interpolated
20	7.22	11.25	14.61	10.62	7.72
40	8.23	14.25	19.84	16.83	8.36
60	9.99	15.45	24.24	31.67	10.34
80	11.51	14.28	21.71	53.91	13.23
90	11.35	14.98	15.13	54.04	13.93

TABLE 12. Imputing missing data via interpolation and with generated data for Navigli district. All values are MAPE. The total amount of data used is five weeks or 840 hourly readings.

Navigli District					
%age data removed	DoppelGANger	TimeGAN	PAR	Missing	Interpolated
20	6.88	6.54	9.91	8.42	7.08
40	6.36	7.47	18.71	9.46	6.9
60	7.02	7.94	21.66	16.08	7.76
80	6.94	8.74	22.83	27.65	11.57
90	7.24	7.8	14.85	27.74	11.31

TABLE 13. Imputing missing data via interpolation and with generated data for Duomo cathedral. All values are MAPE. The total amount of data used is five weeks or 840 hourly readings.

Duomo Cathedral					
%age data removed	DoppelGANger	TimeGAN	PAR	Missing	Interpolated
20	9.68	10.8	64.47	10.83	7.12
40	11.23	14.7	84.37	16.09	8.41
60	12.94	22.33	71.15	29.37	8.54
80	12.4	20.83	36.65	72.56	10.26
90	12.13	22.36	30.24	91.13	10.65

case is probably due to our model learning some spurious correlations that happen to allow it to predict well on the test data despite the poor training data. The Navigli district recordings show similar trends, except that the TimeGAN and DoppelGANger perform equally well, and the gap between the GAN imputed data sets and the interpolated data is wider, at around 4% in favor of the DoppelGANger and TimeGAN.

This is not the case in Duomo, where the DoppelGANger is the clear winner. However, simply interpolating over the missing data with a common technique like quadratic interpolation seems to impute the data just as well. The interpolated model's performance deteriorates at 80% and 90% missing data levels, where the DoppelGANger imputed data yields a roughly 2.5% accuracy gain in Bocconi, a 4% gain in Navigli and a degradation of 2% in Duomo. These results emphasize that GAN based models really start yielding true benefits at high sparsity levels, where interpolation schemes fade off.

The key takeaway here is that the DoppelGANger's performance for different types of time series data is more reliable, since using it to impute random gaps in the true data yields the most accuracy, in some cases outperforming the model trained entirely on true data (when 40% data is removed in Navigli) The TimeGAN performs well for Navigli, but does relatively poorly in Bocconi and Duomo. Finally, PAR's output is more or less random, and it only captures the value distribution of the original data sets reasonably well.

VII. CONCLUSION

In this work, we compared three publicly available time-series generative models against each other using an actual mobile network data set. Two methods are based on the GAN - architecture, while one is a deep learning based auto-regressive model. We see that the GAN based architectures are superior to the auto-regressive approach across an array of numerical and graphical measures. We used the generated data to train a supervised machine learning algorithm and assessed its performance on unseen real data. These experiments revealed that models trained on GAN - based DoppelGANger and TimeGAN generated data were competitive with a model trained on true data, but the DoppelGANger was most superior in performance across all three regions. Finally, our simulations revealed that increasing the training data did not always prove beneficial but, in some cases, degraded the generative model's performance and that some models, like the DoppelGANger, perform very well even on relatively little data. We then saw that augmenting small amounts of real data with comparatively large amounts of synthetic data yielded minor performance improvements. Finally, we saw that GAN generated values were a good substitute for real data when imputing missing values in a time-series, but interpolation techniques in most cases can perform just as well except when the number of missing values is quite large.

While this is the first in-depth research on comparing the latest deep learning based generative models for a time series telecommunications data set, future extensions of the work may involve replicating it with a much larger data set, as well as with multivariate time-series data. Another direction could be to work with tabular data, which would entail using different GAN models as well as different evaluation metrics. Finally, the DoppelGANger as well as PAR are capable of reproducing time-series data with corresponding context/metadata information, so it may be worthwhile to see if their performance differs in that environment.

REFERENCES

[1] A. Imran, A. Zoha, and A. Abu-Dayya, "Challenges in 5G: How to empower SON with big data for enabling 5G," *IEEE Netw.*, vol. 28, no. 6, pp. 27–33, Nov./Dec. 2014.
[2] J. Moysen and L. Giupponi, "From 4G to 5G: Self-organized network management meets machine learning," *Comput. Commun.*, vol. 129, pp. 248–268, Sep. 2018.
[3] A. Ghasempour, "Internet of Things in smart grid: Architecture, applications, services, key technologies, and challenges," *Inventions*, vol. 4, no. 1, p. 22, Mar. 2019.
[4] G. J. Sutton, J. Zeng, R. P. Liu, W. Ni, D. N. Nguyen, B. A. Jayawickrama, X. Huang, M. Abolhasan, Z. Zhang, E. Dutkiewicz, and T. Lv, "Enabling technologies for ultra-reliable and low latency communications: From PHY and MAC layer perspectives," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2488–2524, Feb. 2019.
[5] U. S. Hashmi, A. Darbandi, and A. Imran, "Enabling proactive self-healing by data mining network failure logs," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Jan. 2017, pp. 511–517.
[6] U. S. Hashmi, A. Rudrapatna, Z. Zhao, M. Rozwadowski, J. Kang, R. Wuppalapati, and A. Imran, "Towards real-time user QoE assessment via machine learning on LTE network data," in *Proc. IEEE 90th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2019, pp. 1–7.
[7] M. Ibrahim, U. S. Hashmi, M. Nabeel, A. Imran, and S. Ekin, "Embracing complexity: Agent-based modeling for HetNets design and optimization via concurrent reinforcement learning algorithms," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 4, pp. 4042–4062, Dec. 2021.
[8] G. Barlacchi, M. De Nadai, R. Larcher, A. Casella, C. Chitic, G. Torrisi, F. Antonelli, A. Vespignani, A. Pentland, and B. Lepri, "A multi-source dataset of urban life in the city of Milan and the Province of Trentino," *Scientific Data*, vol. 2, no. 1, pp. 1–15, Dec. 2015.
[9] D. Gunduz, P. de Kerret, N. Sidiroupoulos, D. Gesbert, C. Murthy, and M. van der Schaar, "Machine learning in the air," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2184–2199, Oct. 2019.
[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2014.
[11] H. Ye, L. Liang, G. Y. Li, and B.-H. F. Juang, "Deep learning-based end-to-end wireless communication systems with conditional GANs as unknown channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3133–3143, May 2020.
[12] L. Sun, Y. Wang, A. L. Swindlehurst, and X. Tang, "Generative-adversarial-network enabled signal detection for communication systems with unknown channel models," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 47–60, Jan. 2021.
[13] B. Hughes, S. Bothe, H. Farooq, and A. Imran, "Generative adversarial learning for machine learning empowered self organizing 5G networks," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2019, pp. 282–286.
[14] M. N. Fekri, A. M. Ghosh, and K. Grolinger, "Generating energy data for machine learning with recurrent generative adversarial networks," *Energies*, vol. 13, no. 1, pp. 1–23, 2019.
[15] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional GANs," 2017, *arXiv:1706.02633*.
[16] O. Mogren, "C-RNN-GAN: A continuous recurrent neural network with adversarial training," in *Proc. Constructive Mach. Learn. Workshop (CML)*, 2016, p. 1.
[17] C. Han, K. Murao, S. Satoh, and H. Nakayama, "Learning more with less: GAN-based medical image augmentation," *Med. Imag. Technol.*, vol. 37, no. 3, pp. 137–142, 2019.
[18] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification," *Neurocomputing*, vol. 321, pp. 321–331, Dec. 2018.
[19] C. Han, L. Rundo, R. Araki, Y. Furukawa, G. Mauri, H. Nakayama, and H. Hayashi, "Infinite brain MR images: PGGAN-based data augmentation for tumor detection," *Smart Innov., Syst. Technol.*, vol. 151, pp. 291–303, 2020.
[20] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2242–2251.
[21] A. Borji, "Pros and cons of GAN evaluation measures," *Comput. Vis. Image Understand.*, vol. 179, pp. 41–65, Feb. 2018.
[22] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017, p. 30.
[23] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen, "Improved techniques for training GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2016, pp. 1–9.
[24] M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo, "Reliable fidelity and diversity metrics for generative models," in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, H. Daumé, III, and A. Singh, Eds., Jul. 2020, pp. 7176–7185.
[25] K. Shmelkov, C. Schmid, and K. Alahari, "How good is my GAN?" in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 213–229.
[26] *Introduction to Time Series Analysis*, document NIST/SEMATECH, e-Handbook of Statistical Methods, 2013.
[27] L. Kegel, M. Hahmann, and W. Lehner, "Generating what-if scenarios for time series data," in *Proc. 29th Int. Conf. Sci. Stat. Database Manage.*, New York, NY, USA, Jun. 2017, pp. 1–12.
[28] C. Bergmeir, R. J. Hyndman, and J. M. Benítez, "Bagging exponential smoothing methods using STL decomposition and box–cox transformation," *Int. J. Forecasting*, vol. 32, no. 2, pp. 303–312, Apr. 2016.
[29] A. Grover and S. Ermon, "Autoregressive models," Tech. Rep., Nov. 2019.
[30] H. Larochelle and I. Murray, "The neural autoregressive distribution estimator," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, vol. 15, G. Gordon, D. Dunson, and M. Dudík, Eds., Fort Lauderdale, FL, USA, Apr. 2011, pp. 29–37.
[31] N. Patki, R. Wedge, and K. Veeramachaneni, "The synthetic data vault," in *Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA)*, Oct. 2016, pp. 399–410.
[32] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu, "Time series data augmentation for deep learning: A survey," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, Z.-H. Zhou, Ed., Aug. 2021, pp. 4653–4660.
[33] J. F. Nash, Jr., "Equilibrium points in n-person games," *Proc. Nat. Acad. Sci. USA*, vol. 36, no. 1, pp. 48–49, 1950.
[34] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2019, p. 32.
[35] J. Yoon, D. Jarrett, and M. V. D. Schaar, "Time-series generative adversarial networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.
[36] Z. Lin, A. Jain, C. Wang, G. Fanti, and V. Sekar, "Using GANs for sharing networked time series data: Challenges, initial promise, and open questions," in *Proc. ACM Internet Meas. Conf.*, New York, NY, USA, Oct. 2020, pp. 464–483.
[37] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-26, no. 1, pp. 43–49, Feb. 1978.
[38] T. Giorgino, "Computing and visualizing dynamic time warping alignments inR: ThedtwPackage," *J. Stat. Softw.*, vol. 31, no. 7, pp. 1–24, 2009.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.
[40] A. Ahmadi, M. Nabipour, B. Mohammadi-Ivatloo, A. M. Amani, S. Rho, and M. J. Piran, "Long-term wind power forecasting using tree-based learning algorithms," *IEEE Access*, vol. 8, pp. 151511–151522, 2020.

MUHAMMAD HARIS NAVEED received the B.Sc. degree in electrical engineering from the School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Pakistan, in 2021. His research interests include the application of GANs to tabular and time-series networks data and audio keyword spotting using ASR-free approaches.

UMAIR SAJID HASHMI (Member, IEEE) received the B.S. degree in electronics engineering from the GIK Institute of Engineering Sciences and Technology, Pakistan, in 2008, the M.Sc. degree in advanced distributed systems from the University of Leicester, U.K., in 2010, and the Ph.D. degree in electrical and computer engineering from The University of Oklahoma, OK, USA, in 2019. During his Ph.D. degree, he worked as a Graduate Research Assistant with the AI4Networks Research Center. He also worked with AT&T, Atlanta, GA, USA; and Nokia Bell Labs, Murray Hill, NJ, USA, on multiple research internships and co-ops. Since Fall 2019, he has been serving as an Assistant Professor with the School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Pakistan, where he is working in the broad area of 5G wireless networks and application of artificial intelligence toward systems-level performance optimization of wireless networks, and health care applications. He is also affiliated with the University of Toronto as a Postdoctoral Research Fellow. He has published about 20 technical papers in high impact journals and proceedings of IEEE flagships conferences on communications. He has been involved in four NSF funded projects on 5G self organizing networks and is a Co-Pi on an Erasmus+ consortium with a combined award worth of over 4 million USD. Since 2020, he has been serving as a Review Editor for the IoT and sensor networks stream in the *Frontiers in Communications and Networks*.

NAYAB TAJVED received the B.S. degree in electrical engineering from the National University of Science and Technology, Pakistan, in 2021. Her research interest includes tackling data scarcity problems faced in future big-data empowered cellular networks using analytical and machine learning tools.

NEHA SULTAN received the Bachelor of Science (B.Sc.) degree in electrical engineering from the National University of Sciences and Technology (NUST), Pakistan, in 2021. Her research interest includes application of AI-driven systems in wireless networks.

ALI IMRAN (Senior Member, IEEE) received the B.Sc. degree in electrical engineering from the University of Engineering and Technology Lahore, Pakistan, in 2005, and the M.Sc. degree (Hons.) in mobile and satellite communications and the Ph.D. degree from the University of Surrey, Guildford, U.K., in 2007 and 2011, respectively. He is a Presidential Associate Professor of ECE and the Founding Director of the Artificial Intelligence (AI) for Networks Research Center and TurboRAN Testbed for 5G and Beyond, The University of Oklahoma. His research interests include AI and its applications in wireless networks and healthcare. His work on these topics has resulted in several patents and over 100 peer-reviewed articles, including some of the most influential papers in domain of wireless networks automation. On these topics, he has led numerous multinational projects, given invited talks/keynotes and tutorials at international forums and advised major public and private stakeholders and has co-founded multiple start-ups. He is an Associate Fellow of the Higher Education Academy, U.K. He is also a member of the Advisory Board to the Special Technical Community on Big Data and the IEEE Computer Society.