


# Towards a Framework to Evaluate Generative Time Series Models for Mobility Data Features

Iran F. Ribeiro   [ Universidade Federal do Espírito Santo | [iran.ribeiro@edu.ufes.br](mailto:iran.ribeiro@edu.ufes.br) ]

Giovanni Comarela  [ Universidade Federal do Espírito Santo | [gc@inf.ufes.br](mailto:gc@inf.ufes.br) ]

Antonio A.A. Rocha  [ Universidade Federal Fluminense | [arocha@ic.uff.br](mailto:arocha@ic.uff.br) ]

Vinícius F. S. Mota  [ Universidade Federal do Espírito Santo | [vinicius.mota@inf.ufes.br](mailto:vinicius.mota@inf.ufes.br) ]

 Computer Science Department, Universidade Federal do Espírito Santo, Av. Fernando Ferrari, 514, Goiabeiras, Vitória - ES - postal code: 29075-910. Network and Multimedia Lab, Room 31, CT-13 Building.

**Received:** 30 November 2024 • **Accepted:** 04 June 2024 • **Published:** 11 August 2024

**Abstract** Understanding human mobility has implications for several areas, such as immigration, disease control, mobile networks performance, and urban planning. However, gathering and disseminating mobility data face challenges such as data collection, handling of missing information, and privacy protection. An alternative to tackle these problems consists of modeling raw data to generate synthetic data, preserving its characteristics while maintaining its privacy. Thus, we propose MobDeep, a unified framework to compare and evaluate generative models of time series based on mobility data features, which considers statistical and deep learning-based modeling. To achieve its goal, MobDeep receives as input statistical or Generative Adversarial Network-based models (GANs) and the raw mobility data, and outputs synthetic data and the metrics comparing the synthetic with the original data. In such way, MobDeep allows evaluating synthetic datasets through qualitative and quantitative metrics. As a proof-of-concept, MobDeep implements one classical statistical model (ARIMA) and three GANs models. To demonstrate MobDeep on distinct mobility scenarios, we considered an open dataset containing information about bicycle rentals in US cities and a private dataset containing information about a Brazilian metropolis's urban traffic. MobDeep allows observing how each model performs in specific scenarios, depending on the characteristics of the mobility data. Therefore, by using MobDeep researchers can evaluate their resulting models, improving the fidelity of the synthetic data regarding the original dataset.

**Keywords:** Generative adversarial networks, time series, Mobility.

## 1 Introduction

Location-based technologies embedded in vehicles and mobile devices have improved the understanding of human mobility [Gonzalez *et al.*, 2008]. In this sense, information and communication technologies (ICT) and Internet of Things (IoT) devices, have become valuable allies for managers in urban centers [Zhang and Lu, 2020]. For instance, mobility data already allowed researchers to identify improvements in the city's infrastructure [Hillier *et al.*, 2009; Kitamura *et al.*, 2000], to assess how to mitigate natural disasters through crowd control [Helbing *et al.*, 2007; Johansson *et al.*, 2008], and to analyze the propagation of diseases such as malaria [Huang and Tatem, 2013], Ebola [Gomes *et al.*, 2014], and Covid-19 [Kraemer *et al.*, 2020].

These location embedded devices produce temporal-dependent data, also known as *traces* [Mota *et al.*, 2014], which may contain GPS-Position over time, trajectories, transitions between base stations of a wireless network [Zheng *et al.*, 2009; Malandrino *et al.*, 2018], transitions between technical conference sessions [Scott *et al.*, 2009; Ribeiro *et al.*, 2021], geo-localized social networks posts [Silva *et al.*, 2014], and traffic data applications. However, traces are prone to errors, missing data, and more important, may contain sensitive data, which limit their public sharing due to privacy concerns.

The implication of mobility in a given scenario can be assessed through synthetic mobility models or by real mobility datasets traces. Synthetic mobility models range from entirely random to models that aim to mimic social characteristics [Solmaz and Turgut, 2019]. However, although synthetic models allow studies on a larger scale with more repeatability, their realism is still limited. On the other hand, real mobility data allow understanding the dynamics of mobility effects with a greater degree of realism. There is an ongoing effort to make mobility data available in the public domain [Piorkowski *et al.*, 2009; Luca *et al.*, 2021].

Mobility data can be modeled as a time series where each observation is ordered in time, such as position or traffic in a road, in each instant. To overcome the aforementioned issues and scale mobility trace datasets, classical time series analysis, such as an AutoRegressive Integrated Moving Average (ARIMA) have been used to model and reproduce time-dependent and predictable data [Song *et al.*, 2010].

Recently, Generative Adversarial Networks (GANs) have been used to generate synthetic data that mimics the characteristics of the real dataset of images and videos [Goodfellow *et al.*, 2014]. GANs consist of simultaneously training a generative model and a discriminative model. The generative model captures the distribution of the data, while the discriminative model calculates the probability of whether an input comes from the generative model or the training base.

The main advantage of GANs relies on the ability to learn the main characteristics of a real dataset and generate other (synthetic) datasets that preserve such characteristics. For this reason, GANs have been used to generate realistic images and videos, text-to-videos, data imputation, and, generation of time series in the most diverse application domains [Gupta et al., 2018; Yoon et al., 2019; Song et al., 2019; Zhang et al., 2022; Rao et al., 2020; Jauhri et al., 2020; Qu et al., 2020; He et al., 2020].

However, each scenario may have its specificities, such as different mobility patterns, weekdays and weekends, and level of granularity of the movement. Such heterogeneity of scenarios hinders a unique solution for mobility time series modeling. Another challenge regards how to measure the model's suitability to generate synthetic mobility datasets. A challenge that persists concerning the use of GANs for time series is how to evaluate the fidelity and utility of the data generated by a GAN model.

By fidelity, we mean that the synthetic data must present similar statistical distributions to the real data. Furthermore, it keeps key correlations between data headers and time. For instance, if real data presents peaks or cycles, the synthetic data must also present similar behavior. On the other hand, by utility, we mean that a study based on synthetic data must produce similar results as real data. For instance, a Machine Learning-based model trained with synthetic data must deal with the real data.

We can summarize the challenges on generative time series modeling for mobility features as: *i)* to deal with and preserve the temporal and spatial characteristics of mobility; *ii)* to deal with sensitive data by providing a model instead of real data; *iii)* how to specialize the techniques used for modeling time series; and *iv)* how to measure and evaluate the resulting dataset based on proposed models.

To face these challenges, we propose the MobDeep, a deep-learning-based framework for generative modeling of time series of mobility data features, which allows evaluating and comparing the resulting models. Furthermore, MobDeep aims to respond which modeling technique fits better for a specific scenario. In such a way, MobDeep provides a higher level of generalization concerning data characteristics. To tackle the challenge of evaluating the fidelity and utility of a model, we propose a set of quantitative and qualitative metrics, which allow comparing the models. Given a dataset of mobility data features as input, MobDeep fits several models by using a set of algorithms. The generative model must generate datasets with acceptable variability while respecting the temporality characteristics. Each model generates a number of distinct synthetic datasets. MobDeep compares the average and standard deviation of residual error with the original dataset to measure this variability. The qualitative analysis allows visualizing the variability of the resulting datasets. Figure 1 depicts an overview of the proposed framework.

MobDeep implementation considers that mobility features data can be, in some cases, predictable and regular. Therefore, it implements the ARIMA model as a baseline for comparisons against deep learning-based models. In addition, it provides three time-series generative models based on GANs.

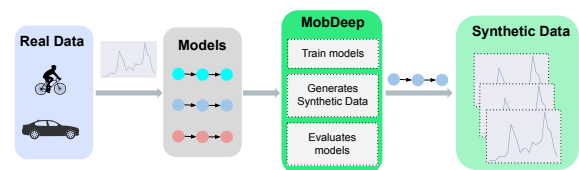
To assess MobDeep feasibility for data modeling and data

generation, we used MobDeep to model a public and a private dataset with different characteristics: Bikesharing, an open dataset with information on shared bicycle rentals in USA cities, which we aimed to model the bike renting around the stations; and VixCity, a dataset provided by the city hall of Vitória, Brazil, with real-time traffic information, in which we aimed to mimic the traffic jams flows. The utility of the synthetic datasets can be measured by training the prediction of bike rented in each station, for Bikesharing, or traffic flow for a given street, for VixCity, using the respective synthetic dataset and testing the model on the real data, that is, train on synthetic and test on real data (TSTR).

For Bikesharing, two out of the three GAN-based models were able to create synthetic datasets similar to the real one. Meanwhile, for the traffic flows generation, we observe that different GAN-based models perform better with a high traffic intensity in the streets and avenues of the city. With this approach, MobDeep allows determining which models produce data that resembles the original dataset while also incorporating sufficient variability.

The contributions of this work are summarized below:

- We present a unified framework to evaluate the performance of deep learning-based models to reproduce mobility features. We considered a classic stochastic model, ARIMA, as a baseline and distinct GANs models to generate synthetic time series with characteristics of the original dataset.
- We demonstrate that generative modeling techniques for time series can reproduce time series of mobility data features, capable of generating data similar to the original dataset.
- By adopting MobDeep, data holders of private datasets containing sensitive information could release only samples of synthetic dataset or their generative models.



**Figure 1.** High-level view of MobDeep Framework. Given a real dataset containing mobility features and a set of generative algorithms and its parameters, Mobdeep fits a generative model for each algorithm and generates a set of synthetic data with it. Finally, Mobdeep evaluates the synthetic data generated by each algorithm, comparing the fidelity and utility of each set of synthetic data. Therefore, researchers can compare which deep learning technique performs better for a specific scenario.

The rest of this article is organized as follows. Section 2 presents the background and motivation. The related work are presented in 3. Section 4 presents the design and overview of MobDeep. We describe the dataset, methodology, and time series GANs for generating time series data in Section 5. The evaluation of MobDeep in the scenarios proposed is presented in Section 6. Finally, Section 7 concludes the paper and discusses future works.

## 2 Background and Motivation

This section discusses the background to model mobility data. First, we considered mobility data a type of time series, which can be modeled by Generative Adversarial Network, a Deep Learning technique. Finally, we present the limitations of state-of-the-art concerning mobility data modeling, specifically time series data of mobility data features, using deep learning and how such limitations motivate this work.

### 2.1 Time Series

A time series  $X = [x_1, x_2, \dots, x_n]$  is a set of observations ordered in time, usually analyzed as a stochastic process [Brockwell and Davis, 2009]. A time series has four main characteristics:

- **Seasonality**: variations in data observations over a specific time or season.
- **Cycles**: variations that can occur at different times and time intervals.
- **Trend**: a change in the time series mean occurring over a long time. Time series average can increase (positive trend), decrease (negative trend), or stay null (horizontal trend).
- **Irregularities**: random changes in the data with no identifiable pattern.

Furthermore, a stochastic process can be stationary or not, which means whether the statistical properties of the process change over time or not. In practice, most of the time series are not stationary, which requires applying some technique, such as differencing, to remove variations in the statistical properties of the time series [Brockwell and Davis, 2009].

A common approach to model and predict values in time series relies on autoregressive models, which assumes that the present value of the series depends on past values together with a random error [Brockwell and Davis, 2009]. In this sense, an autoregressive integrated moving average (ARIMA) is a common approach to model and forecasting non-stationary time series.

### 2.2 Deep Learning Basics

Machine learning algorithms are techniques capable of extracting (or learning) relevant information about a dataset and, subsequently, making predictions based on the learned information. In this context, a Deep Learning algorithm can be defined as a Neural Network with numerous hidden layers in its architecture or, from a learning standpoint, a learning process done through several layers of data representation [Chollet et al., 2018].

Recurrent Neural Networks (RNN) are a type of deep learning architecture proposed to model sequential data in which past observations influence future observations (e.g., time series). The main difference between an RNN and a Neural Network is their recurrent connections, where the output signals of a layer are also inputs to the same layer. In this sense, the Long Short-Term Memory (LSTM) was introduced in 1997 by Hochreiter and Schmidhuber [1997], differing from a conventional RNN by the presence of memory

cells, input, output, and forgetting gates, which are the main components of an LSTM block.

The **Generative Adversarial Networks (GANs)** is a framework proposed by Goodfellow et al. [2014] to optimize the training of generative models. Through a min-max game, two competing neural networks are trained simultaneously and indefinitely: a Generator ( $G$ ), a multilayer perceptron (MLP) that generates false data based on random inputs, and a Discriminator ( $D$ ), another MLP that classifies the generated data, considering a real dataset. The objective of  $G$  is to generate samples whose distribution is so close to the real data distribution that  $D$  cannot distinguish one from another. The framework was initially evaluated with the Toronto Faces Dataset (TFD) dataset [Susskind et al., 2010], a dataset of gray scale images of peoples faces, and became very successful in the computer vision field [Karras et al., 2017; Brock et al., 2018].

Despite their performance, GANs are difficult to train since it is hard to infer the generated data's quality without visually analyzing them periodically. Consequently, during training,  $G$  can generate statistically indistinguishable data from real ones that make no sense to a human observer. Thus, training a GAN for more iterations does not imply better results.

### 2.3 Generative Time Series: Use cases and applications

Generative time series considers the temporal correlation of the original data, while must be able to generate data with some variability. Since GANs were originally proposed for images, in high dynamic range across samples, traditional GANs tend to output similar samples [Lin et al., 2020]. In the domain of images and videos, Inception Score (IS) and Fréchet Inception Distance (FID) are popular metrics to assess the quality of fake data produced by GANs [Borji, 2022]. However, metrics to assess fake images are limited for synthetic time series, due to the temporal dimension [García-Jara et al., 2022]. Therefore, the application and feasibility of Time Series GANs to provide models useful for data-driven research depends on the fidelity and utility of the synthetic data produced by the model.

The **fidelity** of a synthetic dataset can be measured by the distance or divergence of its probability distribution from the real dataset, such as Kullback-Leibler Divergence, Jensen-Shannon Distance, and Wasserstein Distance [Cunha et al., 2022]. However, even these metrics can fail to capture the temporal correlation of the data [Borji, 2022]. Therefore, we claim that qualitative analysis based on visual inspection of random synthetic samples and real data is mandatory.

The **utility** of a synthetic dataset has been commonly measured by a metric *Train on Synthetic - Test on Real* (TSTR), which means training a neural network model with the synthetic data and testing on real data. For instance, a classifier is trained using a synthetic dataset that is subsequently tested on real data. Assuming that the trained model achieves satisfactory results when compared with a model trained with the real data, it means that the GAN model learns the characteristics of the data, and synthetic data can be used for research as the real one.

Despite being a relatively recent technology, we present several uses cases of generative GANs for time series<sup>1</sup>.

**Music:** One of the earliest GANs designed for generating time series was the C-RNN-GAN [Mogren, 2016]. Its architecture resembles a standard GAN, but with LSTMs and fully connected layers serving as the Generator and Discriminator. This model was specifically trained to generate songs from classical composers in the MIDI format. During training, the songs were encoded into numerical representations, enabling the learning process for the Generator to generate fake songs and the Discriminator to classify them. To assess the model, the author compared music features, including polyphony, scale consistency, repetitions, and tone span, between real and synthetic datasets.

**Health:** Given that initial conditions can influence the data of a time series, [Esteban et al., 2017] proposed the RGAN (for real-valued data) and the RCGAN (conditional GAN). Unlike the C-RNN-GAN, the proposed models are composed of only LSTMs. The general objective was the generation of medical data. In addition to the conditional aspect mentioned above, the privacy provided to the data was an essential factor for the proposed model. To demonstrate the model capabilities in potential hospital scenarios, the authors used simulated datasets of sine waves and smooth signals, along with the ICU dataset that contains medical records. To evaluate the model’s performance, the authors used the Maximum Mean Discrepancy (MMD) and the proposed scores Train on Synthetic, Test on Real (TSTR), and Train on Real, Test on Synthetic (TRTS).

**Networking:** More recently, in the context of networks, Doppelganger [Lin et al., 2020] was proposed as an alternative for sharing network data, represented by multivariate time series. The authors compared Doppelganger with RGAN, TimeGAN, and Naive GAN, in terms of fidelity and utility, using TSTR in conjunction with several machine learning algorithms. In this sense, Doppelganger was used by NetShare [Yin et al., 2022], an end-to-end tool for sharing network datasets, such as flows and packet data. The efficiency of NetShare was demonstrated through several quantitative measurements and by testing the synthetic dataset against several classification and prediction algorithms.

**Mobility:** As mobility data is highly predictable and time-dependent, we can approach the modeling of mobility data features as a time series generation problem. However, in the literature, GAN-based techniques for mobility generation often utilize approaches with limited capabilities to handle the temporal dependencies of the data. One reason for this is the notable success of GANs in generating static data, such as images. However, this approach proves inadequate for addressing the dynamic nature inherent in mobility data, potentially resulting in models with restricted generalization across various scenarios. In this sense, recent work have showcased how modeling mobility as time series can improve the data generation. For example, the work by Feng et al. [2020] presents a GAN to model peoples’ mobility trajectories during the COVID-19 pandemic. Similarly, Yu et al. [2020] presents a GAN to simulate taxi rides in urban area in Bei-

jing, China. These advancements in time series modeling using GANs, indicate that diverse types of time series can be effectively modeled, thereby offering the potential to capture a broad range of mobility data features.

**General purpose:** From the point of view of time series data modeling, a general purpose model for times series generation would be domain independent (i.e., if the data is time-dependent, it should be possible to build a GAN model capable of learning its distributions). Although this issue was briefly discussed in C-RNN-GAN and RGAN, the first general purpose model, named TimeGAN, was proposed by Yoon et al. [2019]. TimeGAN has two additional components besides the data generator and discriminator: the Embedding and Recovery network so that the network learns the internal temporal dependencies of the data through its low-dimensional representations. The authors compare TimeGAN with the previous two models using four datasets with different characteristics (periodicity, discreteness, noise level, regularity of time steps, and correlation across time and features) to test TimeGAN’s performance. The evaluation includes three types: i) visualization (t-SNE and PCA) to compare the distributions in a 2-dimensional space; ii) discriminative score, where a post-hoc time series classification model is trained to distinguish between sequences from original and generated datasets; and iii) predictive score, where a post-hoc sequence-prediction model is trained on the generated dataset and tested on the original one. More recently, Jeon et al. [2022] proposed the GT-GAN model to deal with regular time series, in which each time interval has an observation, and irregular time series, where there are no observations in one or more time intervals. Among the models in the literature, this was one of the first to deal with the problem of irregularity in time series, which is common in several real-world problems. GT-GAN was compared with several state-of-the-art models for generating time series, considering visual (T-SNE) and quantitative (discriminative and predictive scores) evaluations.

We highlight the C-RNN-GAN was of the first generative time series GANs. Therefore, we considered C-RNN-GAN as a baseline to evaluate other generative models. Meanwhile, R-GAN have gained attention as one of the main generative time series model [Brophy et al., 2023]. Finally, TimeGAN represents a general purpose generative time series. Our proposed framework considers these three generative time series GAN adapting them to focus on the problem of generative mobility synthetic datasets.

## 2.4 Problem Formulation

Let  $D = \{d_1, d_2, \dots, d_n\}$  be a multivariate time series dataset, where each  $d_i$  represents a tuple  $d_i = \{t_i, \{f_1, f_2, \dots, f_m\}\}$  with a timestamp  $t_i$  for which  $t_i < t_{i+1}$  and a set of features  $f_m$ . Furthermore, consider  $M = \{M_1, M_2, \dots, M_n\}$  generative time series models, where each  $M_i$  receives  $D$  as input to fit a model able to produce synthetic datasets  $Z_{M_i}$ . The problem can be defined as:

Given a set of  $M$  generative models, how to compare and evaluate  $\mathbb{M}$  fitted models according to three main criteria: i) **fidelity**, as  $Z$ , a set of  $Z_{M_i}$  synthetic datasets generated by a specific trained model  $\mathbb{M}_i$ , must be similar to  $D$  consid-

<sup>1</sup>Interested readers can refer to generative time series surveys [Navidan et al., 2021; Zhang, 2003; Iglesias et al., 2023] for further details.

**Table 1.** Related work. Note that our method use different GANs models and each one of them have its architecture

| Application   | Reference                     | Generator                                   | Discriminator                   | Data type       |
|---------------|-------------------------------|---|---------------------------------|-----------------|
| Trajectories  | [Song <i>et al.</i> , 2019]   | CNN   | CNN                             | Images          |
|               | [Zhang <i>et al.</i> , 2022]  | CNN   | CNN                             | Images          |
|               | [Feng <i>et al.</i> , 2020]   | Embedding, concat, self-attention, Linear   | Embedding, CNN                  | Time series     |
|               | [Rao <i>et al.</i> , 2020]    | Embedding, Feature Fusion, LSTM, Regression | Embedding, Feature Fusion, LSTM | Time series     |
| Rides         | [Yin <i>et al.</i> , 2018]    | Dense                                       | Dense                           | Matrices        |
|               | [Yu <i>et al.</i> , 2020]     | LSTM  | LSTM                            | Time series     |
|               | [Jauhri <i>et al.</i> , 2020] | Dense                                       | Dense                           | Images          |
| Contacts      | [Lei <i>et al.</i> , 2019]    | GCN, LSTM, FC                               | GCN, LSTM, FC                   | Temporal graphs |
|               | [Zhang, 2019]                 | Dense, LSTM                                 | LSTM                            | Temporal graphs |
|               | [Qu <i>et al.</i> , 2020]     | MLP   | MLP                             | Temporal graphs |
| Urban traffic | [He <i>et al.</i> , 2020]     | MLP   | MLP                             | Matrices        |
|               | Our method                    | Agnostic                                    | Agnostic                        | Time series     |

ering its distributions and temporal characteristics; ii) **variability**, as is desired that each dataset  $Z_{M_i} \in Z$  produced by  $\mathbb{M}_i$  should be distinct among themselves and; iii) **utility**, machine learning algorithms, such as those for predictions, classifications, and clustering, trained with the synthetic dataset must present good results when applied to the real dataset.

Although the above general problem definition fits for any generative time series model, this work focuses on mobility features, since mobility datasets usually contain sensitive data or the features that are time-dependent among them. For instance, a city traffic dataset containing a tuple  $d_1 = \{1, \{10, 20, 30, \dots\}\}$  representing that in time  $t = 1$  there are 10, 20, and 30 cars in streets 1, 2, and 3, respectively, as features. In such scenario, the traffic jam on a street reflects on other streets.

### 3 Related work

We classify the related work on generative gan-based models of mobility dataset into four categories: trajectories generation, transport data generation, contacts generation, and urban traffic generation. In this sense, Luca *et al.* [2021] present an in-depth study on the generation of trajectories and urban traffic, using different deep learning models.

**Trajectories generation:** this category aims at generating trajectories made by people or vehicles from geolocation datasets. Early works employed similar methodologies to approach the problem: the region from the actual trajectories was modeled as an image, allowing the data to be viewed as a  $N \times N$  matrix. Thus, this approach allows the application of already known GAN models for image generation. For example, Song *et al.* [2019] use locations of people in 5 cities of South Korea, and Zhang *et al.* [2022] use data of taxi trajectories. Using a distinct approach, Feng *et al.* [2020] proposes a specific GAN architecture for trajectory generation. Concerning privacy issues, Rao *et al.* [2020] proposes the LSTM-TrajGAN, a model for generating trajectories based on GPS locations.

**Rides generation:** this category addresses the generation of geolocation data produced by taxis and riding-sharing services (i.e., origin and destination points of a ride). In Yin *et al.* [2018] the authors focus the study on privacy preservation of the generated data using a vanilla GAN with a loss function based on Wasserstein distance [Vallender, 1974].

In Yu *et al.* [2020] the authors simulate taxi demand data, considering the origin and destination locations of users, using GANs composed by Conditional GANs (CGANs) and LSTMs. For the ride-sharing problems, Jauhri *et al.* [2020] developed a model of ride request generation using datasets from 4 cities in the USA. The data is modeled as a sequence of images, and GANs models for image generation are used.

**Contact generation:** works in this category investigate the simulation of contacts among nodes (people, devices, or vehicles) in a network. For example, Lei *et al.* [2019] model mobility data as a dynamic graph and combine a GAN with a Graph Convolutional Network (GCN) to simulate the contacts made within the network. On the other hand, considering a dynamic network structure, Zhang [2019] models the contacts between passengers of the Washington Metro system using LSTM-based GANs. Considering the data privacy in a person-to-person contact dataset, Qu *et al.* [2020] proposes a GAN with differential privacy during the mobility data generation, ensuring a higher level of privacy compared to other anonymization methods.

**Urban traffic generation:** to the best of our knowledge, only He *et al.* [2020] refers to the generation of data regarding the traffic dynamics of a city. The authors use GANs to retrieve general traffic information based on specific intersections between streets in the studied region. The data used are static and modeled as matrices. Since our approach seeks to generate mobility data referring to urban traffic data (number of bicycles rented in a time interval and number of cars on the streets in a time interval), the present falls in the category of urban traffic generation. Our approach, however, considers the potential of using GAN models to generate time series data in this context, with efficiency and a sufficient level of generalization for such type of mobility data.

Table 1 presents a non-exhaustive list of related works, providing a comparison of their main characteristics. Each row shows the main paper’s application, the reference, the GAN’s main components on the generator and discriminator, and the data type expected by the model. Note that, although we do not propose a new architecture, we define our method as agnostic, since we can use any time series GAN model for the data generation. In summary, we provide a method of training and evaluating different time series GAN models as a way of easing the training and generation process, as well as providing evaluation techniques more consistent for time-dependent data.



## 4 MobDeep Design

In this section, we describe the main components of MobDeep, the models used to generate the synthetic data and present an overview of the metrics we use to evaluate the trained models.

### 4.1 Overview

We propose a framework to train and evaluate deep learning models for modeling time series of mobility data features (MobDeep). To achieve its goal, MobDeep receives a time series of mobility data features as input, attempts to fit the dataset model with an autoregressive (baseline) and a set of deep learning models (C-RNN-GAN, RGAN and TimeGAN, described previously), and generates  $n_z$  synthetic datasets for each model. As a result, it allows identifying which technique fits better for a specific dataset. Figure 2 depicts MobDeep's components:

- **Model Generator:** generates a fitted model, taking as input a dataset  $D$  of dimensions  $n \times t \times m$  ( $n$  days of observations,  $t$  observations per day, and  $m$  features per observation) and a model  $M_i \in M = \{M_1, M_2, \dots, M_n\}$ , where  $M_i$  is the  $i$ th model used. Furthermore, given that each  $M_i$  may require different parameters' setup, a setup file ( $P_M$ ) is also given as a parameter to the Model Generator.  $P_M$  has, besides the model's hyperparameters, specific information to ease the training and evaluation of a  $M_i$ , for example, the directory where the generated data and plots will be saved. The Model Generator Component returns a generative model  $M_i$ .
- **Data Generator:** given a generative model  $M_i$  and the number of desired datasets ( $n_z$ ), it outputs a set  $Z = \{Z_{M_i}, \dots, Z_{M_n}\}$  of synthetic datasets, where  $Z_{M_i}$  will have  $n_z$  samples.
- **Model Evaluation:** evaluates the residual error of datasets generated by a specific  $M_i$  model or by a set of  $M_i$  trained models ( $\mathbb{M}_i$ ).

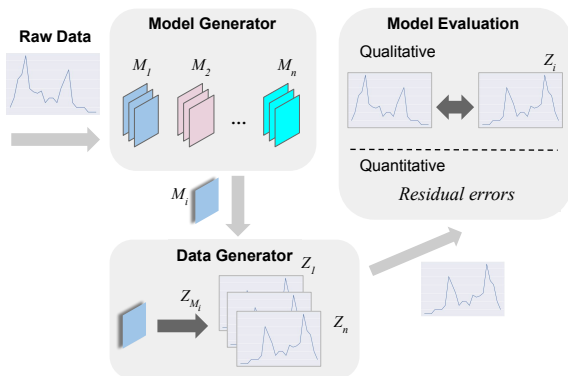


Figure 2. The overview of MobDeep framework.

Details of each component implementation are shown in Algorithm 1. Due to the number of parameters in the models, each component works based on a setup file ( $P_M$ ), which defines a list of model parameters and other information to ease the training and evaluation, for example, a specific folder to

---

### Algorithm 1: data generation and evaluation.

---

```

Input: Dataset  $D$ , Model  $M_i$ , Model setup  $P_M$ 
1 function ModelGenerator( $D, M_i, P_M, n_p$ ):
2    $P_M \leftarrow$  read parameters list from  $Setup$  file;
3   for  $j := 0$  to  $n_p$  do
4      $M_i[j] \leftarrow$  train model  $M_i$  using  $P_M^j$  and  $D$ ;
5   return  $\mathbb{M}_i$ ;
6 end function
7 function DataGenerator( $n_z, M_i, \mathbb{M}_i$ ):
8   if  $M_i$  is defined then
9     for  $j := 0$  to  $n_z$  do
10       $z \leftarrow$  generates a sample of  $D$  using  $M_i$ ;
11       $Z_{M_i}[j] \leftarrow z$ ;
12       $Z[M_i] \leftarrow Z_{M_i}$ ;
13   else
14     foreach  $M_i \in \mathbb{M}_i$  do
15       for  $j := 0$  to  $n_z$  do
16          $z \leftarrow$  generate a sample of  $D$  using  $M_i$ ;
17          $Z_{M_i}[j] \leftarrow z$ ;
18          $Z[M_i] \leftarrow Z_{M_i}$ ;
19   return  $Z$ ;
20 end function
21 function ModelEvaluation( $D, Z, n_z$ ):
22    $SI_D^W \leftarrow$  compute  $SI$  of  $D$  using Equation 4;
23   for  $Z_{M_i} \in Z$  do
24     for  $j := 0$  to  $n_z$  do
25        $z \leftarrow Z_{M_i}[j]$ ;
26        $Res_j \leftarrow$  residual error of  $z$  (Equation 1);
27       Plot residual error for  $Res_j$ ;
28        $SI_Z^W \leftarrow$  compute  $SI$  of  $z$ ;
29       Compare  $SI_Z^W$  against  $SI_D^W$ ;
30 end function

```

---

save the model, its synthetic data and eventual plots. This information allows MobDeep to generate models ( $M_i$ ) with distinct parameters' setup ( $P_M$ ). Besides, it allows *Model Evaluation* component to know which set of synthetic data to analyze. In our implementation, this setup is done through predefined JSON files and, for the sake of simplicity, we show the setup file only in the *ModelGenerator* component. Further details of each component are presented below.

### 4.2 Model Generator

The *Model Generator* component receives a dataset  $D$ , the model ( $M_i$ ), and experiment setup ( $P_M$ ). As previously stated,  $D$  has the dimension of  $n \times t \times m$ .  $M_i$  indicates the model that attempts to fit the original dataset ( $D$ ). For each parameter set of a model,  $P_M^j \in P_M$ , the *ModelGenerator* component trains a generative model and saves it in the disk (i.e., we train the same model with  $k$  different parameters sets). For the sake of simplicity we omitted, in the Algorithm, the filtering and the pre-processing of the dataset before passing it as an argument.

### 4.3 Data Generator

The *Data Generator* component receives a set of trained models ( $\mathbb{M}_i$ ), in the general case or, optionally, a specific trained model ( $M_i$ ). It receives, also, the number ( $n_z$ ) of synthetic data the user aims to generate. In this way, for the general case, the *Data Generator* will generate  $n_z$  synthetic datasets using each  $M_i$  model in  $\mathbb{M}_i$  or  $n_z$  synthetic datasets using the specified  $M_i$  model. Furthermore, MobDeep can call this component directly after training a model, as shown in the downward arrow of Figure 2.

A good generative model should create synthetic datasets with variability while preserving the statistical properties of the original dataset. Each generative model must create at least one set of synthetic data. Thereby, the *ModelEvaluation* component can evaluate the performance of a generative model.

### 4.4 Model Evaluation

The *ModelEvaluation* component generates quantitative and qualitative analysis based on the synthetics datasets  $Z$ .

#### 4.4.1 Quantitative Analysis

The quantitative analysis considers the residual errors of the trained models, computing its mean and residual standard deviation. A residual error ( $Res$ ) of a model can be defined as the difference between the expected output of a model ( $D$ ) and what the model actually produces ( $z$ ):

$$Res = |D - z| \quad (1)$$

Thus, *AnalysisGenerator* computes the residual mean of the models ( $Mean_{Res}$ ) considering each of the generated synthetic data as shown in Equation 2.

$$Mean_{Res} = \frac{\sum_{i=1}^{n_z} Res_i}{n_z} \quad (2)$$

where  $n_z$  is the number of datasets to be evaluated, and  $Res_i$  is the residual error ( $Res$ ) of the  $i$ -th synthetic dataset. Similarly, the residual standard deviation is calculated with Equation 3:

$$Std_{Res} = \sqrt{\frac{\sum_{i=1}^{n_z} (Res_i - Mean_{Res})^2}{n_z}} \quad (3)$$

For the data generation problem, we consider a good model the one that can generate synthetic data similar to the real ones and that have variability (*i.e.*,  $Mean_{Res}$  and  $Std_{Res}$  are  $\approx 0$ ). In this way, MobDeep can define thresholds to  $Mean_{Res}$  and  $Std_{Res}$ , which allows it to indicate the best model. We observed, as discussed in Section 5, models with a lower  $Mean_{Res}$  tend to produce better synthetic data.

#### 4.4.2 Qualitative Analysis

In the qualitative analysis, MobDeep produces comparative visualizations between synthetic and real data. To visualize whether the synthetic dataset follows the pattern of the original data, we propose a metric that allows visualizing of the

dataset each day of the week, separately, called *Sum by Intervals (SI)*, given as below.

**Sum by Interval (SI) Definition:** Consider a matrix  $D$  of dimensions  $k \times l$  where each  $k$  row contains all events of a day, from 0am to 11h59pm. In  $D$ ,  $d_{i,j} = [timestamp_i, value]$  and  $timestamp_i < timestamp_{i+1}$ .

$$D = \begin{bmatrix} d_{0,0} & \dots & d_{0,l} \\ \vdots & \ddots & \vdots \\ d_{k,0} & \dots & d_{k,l} \end{bmatrix}$$

Since we aim to visualize the behavior of the time series features for each day of the week, we group the rows that correspond to a specific day of the week. Let  $D^w$  (Fig. 3a) be a matrix with a dimension  $k \times l'$ , where  $w$  is the day of the week,  $k$  is the number of days in  $D^w$  with  $l'$  values in each day. Each  $D^w$  contains all data of a day of the week  $w$ , where  $|w| = 7$  (Monday, Tuesday, ..., Sunday).

For each day of the week  $w$ , the Sum by Interval (*SI*) vector is given by the sum of all values in each column of the respective  $D^w$  matrix, that is

$$SI[j] = \sum_{j=0}^k D_{j,i}^w \quad (4)$$

as depicted in Figure 3b.

$$D^w = \begin{bmatrix} d_{0,0} & d_{0,1} & \dots & d_{0,l'} \\ d_{1,0} & d_{1,1} & \dots & d_{1,l'} \\ \vdots & \vdots & \ddots & \vdots \\ d_{k,0} & d_{k,1} & \dots & d_{k,l'} \end{bmatrix}$$

(a) Each matrix row  $D^w$  is a day of the week  $W$ .

$$SI^w = \begin{bmatrix} \sum_{j=1}^k d_{j,0} & \sum_{j=1}^k d_{j,1} & \dots & \sum_{j=1}^k d_{j,l'} \end{bmatrix}$$

(b) Matrix for computing the SI for a specific  $W$ .

**Figure 3.** Illustration of the matrices used to compute SI.

The visualizations generated from *SI* allow the identification of properties and patterns in the data, such as peaks and differences between each day of the week. Another advantage of *SI* is the use of the entire dataset to perform the comparisons. It is expected that a good model will be able to generate synthetic data with similar properties, for example, peaks during working days and possible differences between working days and weekends. Table 2 shows a summary of parameters used in this paper.

**Table 2.** Summary of Parameters and Metrics

| Variable       | Description  |
|----------------|--|
| $D$            | A matrix where each row is a day of observation    |
| $D^w$          | Dataset that correspond to the day of the week $w$ |
| $n$            | Number of days of observation in $D$               |
| $t$            | Number of observations per day                     |
| $m$            | Number of features in each observation             |
| $z$            | A synthetic dataset                                |
| $M$            | Set of trainable models                            |
| $M_i$          | A specific model                                   |
| $\mathbb{M}_i$ | Set of trained $M_i$ models                        |
| $P_M$          | Set of parameters for a model $M$                  |
| $n_p$          | Number of parameters sets                          |
| $n_z$          | Number of synthetic dataset to be generated        |
| $Z_{M_i}$      | Set of $n_z$ synthetic datasets from a $M_i$ model |
| $Z$            | Set of $Z_{M_i}$ datasets                          |
| $SI_D^W$       | Sum by interval for every day for real data        |
| $SI_D^Z$       | Sum by interval for every day for synthetic data   |
| $Res$          | Residual error                                     |
| $Mean_{Res}$   | Mean residual errors                               |
| $Std_{Res}$    | Standard deviation residual errors                 |

## 5 Experiments

### 5.1 Datasets

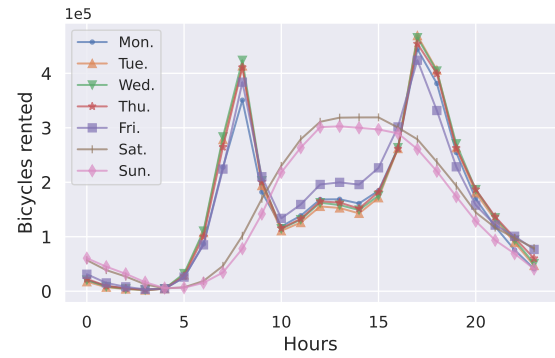
We used two datasets to carry out the experiments to evaluate MobDeep. The first one contains information about rentals of bicycles in 7 cities in the United States (Washington, Arlington, Alexandria, Montgomery, Prince George’s County, Fairfax County, and Falls Church) between January 2011 and February 2020, through the Bikesharing service<sup>2</sup>. The data from January 2011 to December 2012 were obtained from Fanaee-T and Gama [2014], and from January 2013 to December 2019, from the Capital Bikeshare website. The attributes of this dataset and their respective descriptions can be seen in Table 3. Our objective was to model the number of bicycles rented every hour of a day in the seven cities. We split this dataset into two parts. Bikesharing (from 2012 to 2019) is used for model training and Bikesharing<sub>t</sub> (January 01 to February 28, 2020), which will be used to validate the use cases proposed in the Section.

**Table 3.** Description of Bikesharing dataset.

| Attribute            | Description                                |
|----------------------|--|
| <b>Duration</b>      | Trip duration                              |
| <b>Start Date</b>    | Date and time of the start of the rental   |
| <b>End Date</b>      | Date and time of the end of the rental     |
| <b>Start Station</b> | Name and number of the origin station      |
| <b>End Station</b>   | Name and number of the destination station |
| <b>Bike Number</b>   | Identifier of the bicycle used on the trip |
| <b>Member Type</b>   | Membership type (registered or casual)     |

Figure 4 shows the SI of the base Bikesharing from Monday to Sunday. We can see that the weekdays have very similar number of rentals, with more rented bicycles around 8am, and 5pm, with peaks ranging between 350k and 450k rentals at 8am and 420k and 480k at 3pm. In addition, there is a clear difference between working days and weekends and a higher bicycle rental around 1pm in the latter case, with peaks of, approximately, 300k rentals.

<sup>2</sup><https://www.capitalbikeshare.com>

**Figure 4.** Sum by Interval (SI) for the Bikesharing dataset.

The second dataset, named VixCity, consists of a dataset composed of information about traffic in Vitória, Espírito Santo, Brazil. The dataset results from a partnership between the city of Vitória and the transit app for mobile devices, Waze.

In the VixCity dataset, users report atypical events on the roads (e.g., accidents, flooding, and traffic jams). The dataset also contains congestion levels by counting the approximate number of cars on congested roads. As a crowd-sourced service, the application’s efficiency depends on the number of connected users generating traffic information [Lenkei, 2018]. Thus, Waze provides three types of datasets through an API: one with alerts, one with congestions, and one with irregularities.

**Table 4.** Description of alerts dataset

| Attribute                | Description  |
|--------------------------|--|
| <b>uuid</b>              | Unique alert identifier  |
| <b>street, city</b>      | Street and city where the event was reported                                 |
| <b>confidence</b>        | Confidence of the event based on feedback from users. Ranges between 0 – 10. |
| <b>reliability</b>       | Reliability of the event based on feedback from users. Ranges between 0 – 5  |
| <b>type, subType</b>     | Event type and subtype   |
| <b>roadType</b>          | Type of street (street, avenue).   |
| <b>location</b>          | Coordinates of the event   |
| <b>eventDate</b>         | Event timestamp  |
| <b>speed</b>             | Current average speed in the jammed section of the road.                     |
| <b>reportDescription</b> | Short event description  |

We use the *alerts* dataset, whose description can be seen in Table 4. In this case, our objective was to model the number of cars in 30-minute time intervals in each street. For that, we use the column *uuid* to count the number of cars in each street, the column *street* to specify the streets to use, and the column *eventDate* to select a specific time range and group the data by day of the week. Due to the number of streets present in the dataset, we selected those with a total number of reported events greater than 25000. The selected streets are depicted in Figure 5.

Figure 6 shows the SI of the most representative streets in the VixCity dataset. The patterns seen in St. I (Figure 6a) are similar to the ones in St. VI and St. VIII, where there are few cars for most of the day and peaks starting around the interval 40. Also, St. III (Figure 6c) and St. VII are similar and resemble the Bikesharing dataset, with peaks in the mornings and afternoons. The streets St. II (Figure 6b) and St. V on the other hand, do not present clear patterns.



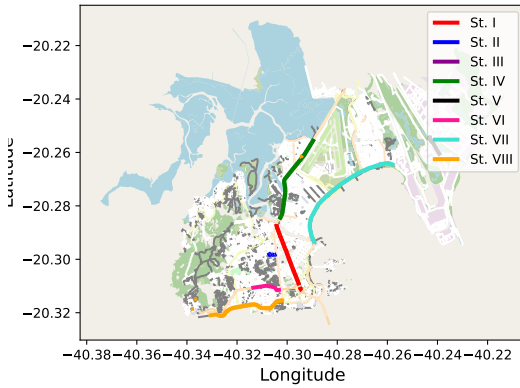


Figure 5. Vitória city map highlighting the eight streets used.

As with the Bikesharing dataset, working days and weekends are quite distinct.

## 5.2 Pre-processing

The first processing applied to the data was the division of the VixCity dataset into two categories. The first one is a univariate dataset consisting of the streets from I to III (we named these datasets as St. I, II, and III, respectively). The second one is a multivariate dataset, formed by streets from IV to VIII, where each street represents a feature in the dataset (named as VixStreets). We use this approach to evaluate how the models would perform on similar datasets, (of traffic in streets) but in different scenario (univariate and multivariate). For the purpose of evaluating the use cases, we selected 5 more streets from the VixCity dataset, which are not used for training the GANs models. We call this dataset VixStreet<sub>t</sub>.

Next, we deal with the missing data. In this case, for both the Bikesharing and VixCity dataset, we entered 0 where there was no data. As the purpose of generating models is to capture the main characteristics of a dataset, they must be able to reproduce the difference of observations in a given period if this is relevant in the actual data (e.g., the missing data).

Specifically for ARIMA, we verified if the univariate datasets, where it would be applied, were stationary. The tests showed that no differentiation was necessary before using this model. However, we applied the log function to stabilize the high variance present in the Bikesharing dataset.

For the GAN models, the main pre-processing performed was the data normalization. In this sense, we use min-max normalization to ensure that the data vary within the range  $[0, 1]$ . Furthermore, as previously mentioned, the datasets must have the dimensions  $n \times t \times m$ . Thus, the dimensions for the Bikesharing, St. I, St. II, St. III, and VixStreets are, respectively,  $(3287 \times 24 \times 1)$ ,  $(250 \times 48 \times 1)$ ,  $(230 \times 48 \times 1)$ ,  $(257 \times 48 \times 1)$ , and  $(250 \times 48 \times 5)$ .

## 5.3 Hyperparameters

We sought to identify which parameters presented the best results for the evaluated models in the experiments. For ARIMA, as all datasets used were stationary, it was defined

that  $d = 0$ . Finally, we use the *pmdarima* [Smith et al., 17], a library that automatically identifies the combination of parameters that best fit the data. The best parameters for each dataset are shown in Table 5.

Table 5. ARIMA parameters  $p$ ,  $d$ , and  $q$  for each dataset.

| Parameters | Bikesharing | St. I | St. II | St. III |
|------------|-------------|-------|--------|---------|
| $p$        | 2           | 2     | 3      | 2       |
| $d$        | 0           | 0     | 0      | 0       |
| $q$        | 3           | 1     | 3      | 3       |

For the GANs, we first verified the parameters that most influenced the training and defined different values for each one of them. Then we train the models on the different parameters and evaluate the results. A preliminary analysis helped define the parameters.

Table 6. GAN parameters with the batch size, learning rate, hidden dimensions, and epochs.

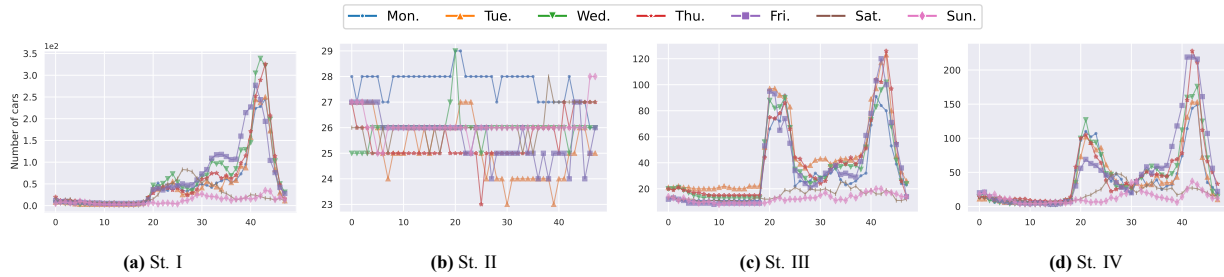
| Dataset           | Model     | bs | lr     | hd  | ep   |
|-------------------|-----------|----|--------|-----|------|
| Bikesharing       | C-RNN-GAN | 28 | .0001  | 100 | 50   |
|                   | RGAN      | 28 | 0.1    | 25  | 3000 |
|                   | TimeGAN   | 28 | 0.0005 | 24  | 5000 |
| St. I, II and III | C-RNN-GAN | 50 | .0001  | 100 | 300  |
|                   | RGAN      | 50 | 0.1    | 32  | 3000 |
|                   | TimeGAN   | 50 | 0.0005 | 72  | 4000 |
| VixStreets        | C-RNN-GAN | 50 | 0.0001 | 28  | 200  |
|                   | RGAN      | 50 | 0.1    | 64  | 3000 |
|                   | TimeGAN   | 50 | 0.0005 | 72  | 3000 |

Table 6 shows the best performance parameters for the Bikesharing, St. I to III, and VixStreets datasets. As each model has many parameters, we only display those that have the most influence on the training performance. It is noteworthy that, given the random behavior of the GANs, the *ep* parameter indicates the approximate number of iterations in which it was possible to obtain good results from the model.

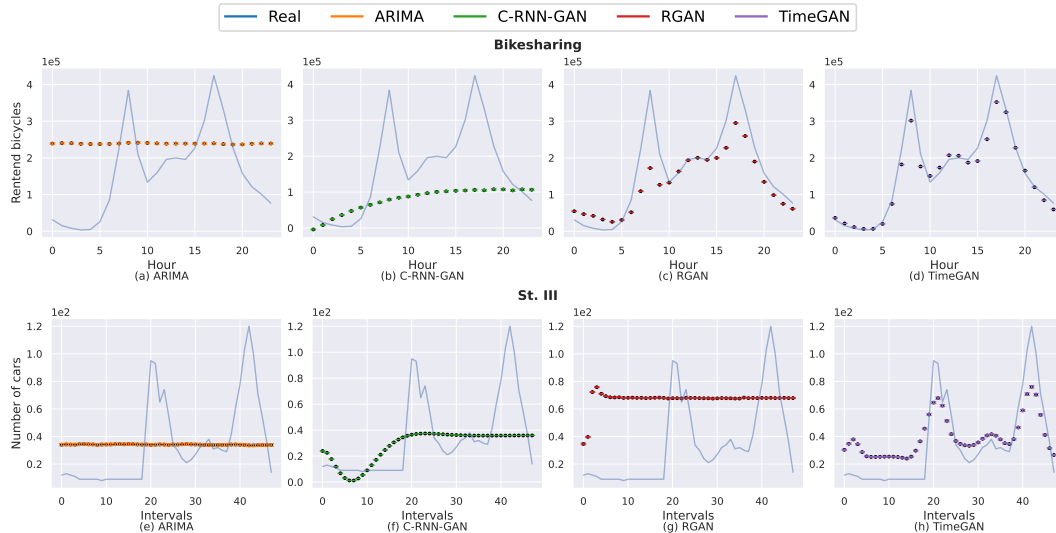
In this sense, the behavior of the GANs, regarding the different parameters, will be similar to the behavior of a Neural Network. For example, data with very complex relationships may require models with more ability to learn, such as higher values for the *hd* parameter. Also, decreasing the value of the *lr* parameter makes the models take longer to update their weights and, consequently, take longer to generate good synthetic data. On the other hand, a high *lr* can cause overfitting, decreasing the variability of the generated data. Similarly, a large *bs* reduces the model's generalization [Keskar et al., 2016].

## 5.4 Use cases

As already mentioned, the main motivations regarding the evaluation of the generation of synthetic datasets is how and if they really have the potential to be used in real situations. In this context, we propose the following use cases: i) with the synthetic data of rented bicycles, is it possible to train a model to predict the number of rented bicycles in the coming months? (for example, for resource allocation and bicycle maintenance) ii) with the synthetic data of cars on the streets, is it possible to train a model to predict the number of cars



**Figure 6.** Sum by intervals for the St. I (a), St. II (b) and St. III (c) datasets and one of the streets in VixCity dataset (St. IV) (d). The x-axis represents the intervals (every 30 minutes of a day).



**Figure 7.** Confidence interval for the mean of SI from 1000 synthetic data generated by ARIMA (a), C-RNN-GAN (b), RGAN (c), and TimeGAN (d) for the Fridays of Bikesharing dataset and; Confidence interval for the mean of SI from 1000 synthetic data generated by ARIMA (e), C-RNN-GAN (f), RGAN (g), and TimeGAN (h) for the Fridays of St. III dataset.

on the streets in the coming months? (for example, for traffic control and assisting in urban planning).

For this, we train an RNN on each synthetic dataset generated by each of the GAN models (C-RNN-GAN, RGAN, TimeGAN) for the respective datasets. In this case, we chose to use only the data referring to the Bikesharing and VixStreets datasets. The first, because it is a dataset with a longer time interval and the second because it already represents the street scenarios (St. I, St. II and St. III) and is a multivariate dataset. This way, we train the RNN to predict, from one day of data as input, the next hour (in the case of Bikesharing<sub>t</sub>) or the next 30 minutes (in the case of VixStreets<sub>t</sub>). Finally, each trained model is validated using the respective test datasets, applying the Mean Absolute Error (MAE). As a baseline, we trained an RNN on the original training data (Bikesharing and VixStreets datasets).

Thus, if the models have, indeed, learned the main characteristics of the original data, we expect to observe, by using the synthetic datasets, consistent results in the quantitative and qualitative evaluations and the scenarios proposed as use cases. In other words, if a model performs better in quantitative and qualitative analysis, this should also be observable in the use case analysis.

## 6 Evaluation

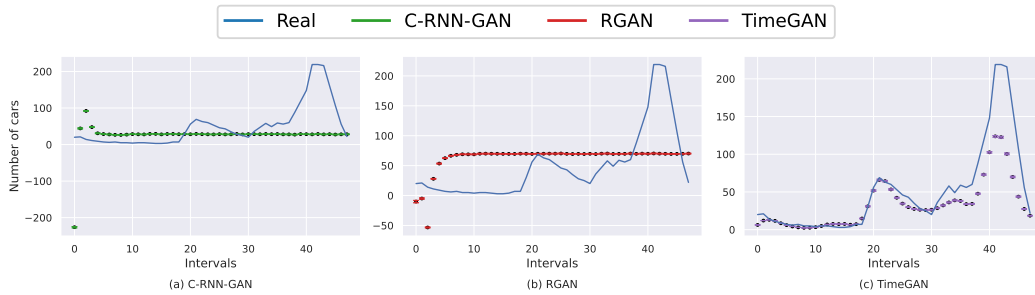
First, this section presents the results of the qualitative evaluation, considering  $SI$  of synthetic data generated by each

model. Next, we discuss the quantitative evaluation, with the models' residual mean ( $Mean_{Res}$ ) and standard deviation ( $Std_{Res}$ ).

### 6.1 Qualitative Analysis

To evaluate the performance of the models in generating synthetic data with the same characteristics as the real ones, we performed the  $SI$  of 1000 synthetic datasets of Fridays, that is, the  $SI^{w=Friday}$  of only days that are Fridays in the dataset. For each model, we calculated the mean and standard deviation of the  $SI$  for the synthetic data. Initially, we verified the performance of the models in the Bikesharing dataset, which has the highest variance, with evident characteristics, and the longest duration. Next, we present the performance for the St. III dataset, which represents a scenario with a minor variance but with identifiable characteristics. Finally, we discuss the most complex scenario in the VixStreets dataset, in which the models must learn the characteristics of more than one street simultaneously.

Thus, Figure 7 shows the confidence interval of  $SI$  from 1000 synthetic Bikesharing (Figures from a to d) and St. III datasets (Figures from e to g). For the Bikesharing dataset, the figure shows that only RGAN (Figure 7c) and TimeGAN models (Figure 7d) generate synthetic data similar to the real data, with peaks around 8am and 6pm. On the other hand, both ARIMA (Figure 7a) and C-RNN-GAN (Figure 7b) could not reproduce the characteristics of the real data. In this case, ARIMA is generally more efficient in stationary



**Figure 8.** Confidence interval for the mean of SI from 1000 synthetic data generated by C-RNN-GAN (a), RGAN (b), and TimeGAN (c) for the Fridays of the St. IV dataset.

**Table 7.**  $Mean_{Res}$  (left) and  $Std_{Res}$  (in parentheses). The best results are in bold.

| Model              | ARIMA               | C-RNN-GAN            | RGAN                 | TimeGAN              |
|--------------------|---------------------|----------------------|----------------------|----------------------|
| <b>Bikesharing</b> | 0.423 (0.01)        | 0.136 (0.115)        | <b>0.103 (0.091)</b> | 0.104 (0.084)        |
| <b>St. I</b>       | 1.225 (0.03)        | 0.033 (0.015)        | <b>0.027 (0.005)</b> | 0.048 (0.028)        |
| <b>St. II</b>      | <b>0.08 (0.004)</b> | 0.300 (0.010)        | 0.316 (0.011)        | 0.315 (0.10)         |
| <b>St. III</b>     | 0.597 (0.018)       | <b>0.015 (0.011)</b> | 0.068 (0.004)        | 0.068 (0.019)        |
| <b>VixStreets</b>  | -                   | 0.061 (0.017)        | 0.041 (0.018)        | <b>0.024 (0.020)</b> |

and linear datasets and datasets with short-term dependencies [Ho *et al.*, 2002]. Similarly, the C-RNN-GAN model is basically an RNN version of a vanilla GAN, which is known to be susceptible to problems such as the vanishing gradient problem and model collapse [Cao *et al.*, 2018], which may be preventing the model to capturing long-term dependencies.

For the St. III datasets, Figure 7g shows the TimeGAN has the best performance, reproducing the peaks in the morning (up to around interval 20) and in the late afternoon, around interval 40.

Finally, in Figure 8 we present the confidence interval for the mean  $SI$  of 1000 synthetic VixStreets datasets and, specifically, for the sake of visualization, we display only the results of the St. IV. Among the 3 models, TimeGAN was the one that managed to generate synthetic data closer to the real data (Figure 8c), with the lower peak between intervals 20 and 30 and the higher one around interval 40. The other models did not reproduce the expected trend for the data, and generated data with negative values. The generation of negative numbers can be justified by the implementation of the models, which were originally designed to work with datasets allowing negative values. However, during training, these models should learn to generate values that are greater than or equal to 0. Although it would have been possible to correct this during implementation, we emphasize that it was not within the scope of this work. Furthermore, we used the original implementations on the other models, which could also generate negative numbers. It is important to emphasize that, in this case, the models were trained to learn the characteristics of each street in the dataset simultaneously. Thus, although the other two GANs were unable to reproduce the data characteristics, TimeGAN’s performance on this problem exceeded our expectations.

Qualitative evaluations show that TimeGAN tends to be the best model used in MobDeep in the different proposed scenarios. In this sense, the models’ architecture is the most likely justification for the results obtained. As mentioned earlier, TimeGAN is the most complex model. The C-RNN-

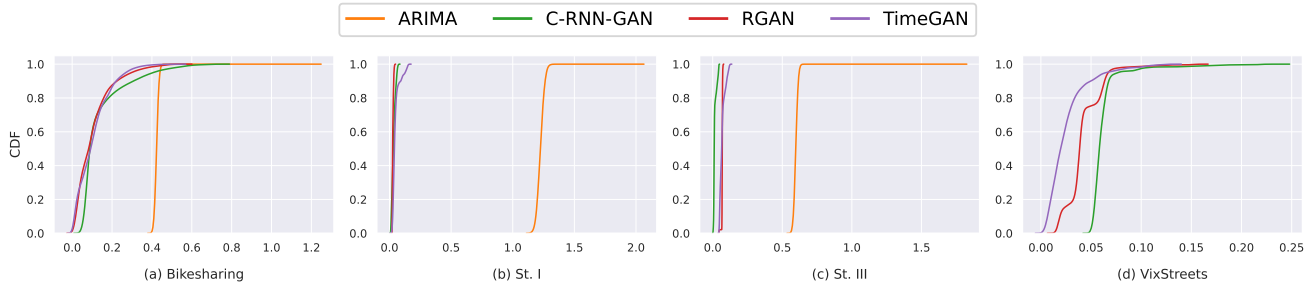
GAN and RGAN codes’ follows the original implementation of their respective authors, and the modifications made to integrate them into MobDeep do not influence their training.

## 6.2 Quantitative Analysis

Table 7 shows the  $Mean_{Res}$  and  $Std_{Res}$  of each model. For these metrics, we assume that the models have learned the main characteristics of real data, such trends and seasonality. This way, a predictive model trained on real data should have, for example, good accuracy when predicting future fake data. A good generative model, evaluated on these metrics, will have a residual with mean and standard deviation close, but not equal, to 0. Therefore, the best model is the one with the lower  $Mean_{Res}$ .

Table 7 demonstrates that, in general, GANs had best results than ARIMA, with RGAN model performing better on Bikesharing and St. II datasets, C-RNN-GAN and TimeGAN models performing better on St. III and VixStreets datasets, respectively, and ARIMA on St. I. It is worth mentioning that, as previously shown in the qualitative analysis, TimeGAN tends to generate fake data that are more similar to the real ones. We can see, for the example, that TimeGAN model had  $Mean_{Res}$  and  $Std_{Res}$  are very close to the RGAN model for Bikesharing dataset.

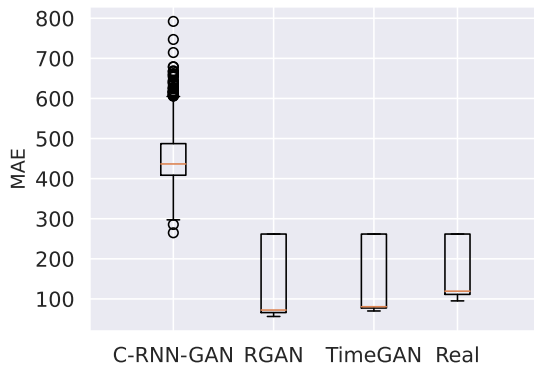
For the St. I and III datasets, the table shows that the RGAN and the C-RNN-GAN were the models that best captured the properties of the real data, respectively. However, in the qualitative evaluations, we showed that, in general, TimeGAN generates better synthetic datasets. In this sense, the performance of the models in both datasets is justified by one main factor: both C-RNN-GAN and RGAN tend to generate data with less variability than TimeGAN. Consequently, the residual errors in these datasets will have lower standard deviations. Finally, the table shows that, except for St. II, the  $Mean_{Res}$  of ARIMA for each dataset are the highest compared to the GAN models. This result corroborates the ones shown in the qualitative evaluations, in which the ARIMA model could not generate any synthetic data with



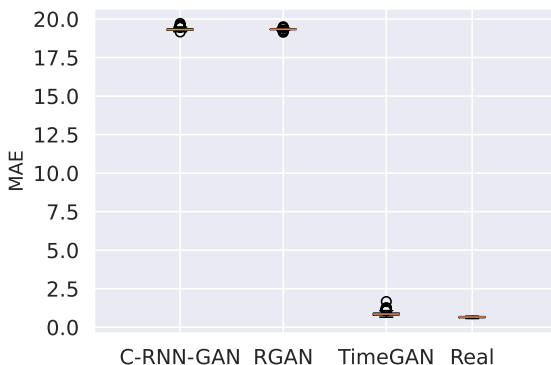
**Figure 9.** Cumulative Distribution Functions of residuals distributions for Bikesharing (a), St. I (b), St. III (c), and (d) VixStreets datasets.

similar to the real data.

Although the previous result summarizes the performance of the models concerning the residual means and standard deviations, the visualization of the cumulative distribution functions of residuals in Figure 9 can provide valuable insights about each model. For example, we can see that for the Bikesharing (Figure 9a), St. I (Figure 9b), and St. III (Figure 9c) datasets, the CDFs of all GANs models shows that  $\approx 90\%$  of residuals are in the range of  $[\approx 0.1, \approx 0.2]$ . However, for the same datasets, the ARIMA residuals have a much higher distribution, with  $\approx 90\%$  less than or equal to  $\approx 0.45$ ,  $\approx 1.3$ , and  $\approx 0.51$ , respectively. On the other hand, Figure 9d shows that for the VixStreets dataset, the GANs models had CDFs with  $\approx 90\%$  of residuals less than or equal to  $\approx 0.07$  (C-RNN-GAN),  $\approx 0.06$  (R-GAN), and  $\approx 0.05$  (TimeGAN).



**Figure 10.** Boxplot of MAE for C-RNN-GAN, RGAN, TimeGAN e the real data for the Bikesharing<sub>t</sub> dataset.



**Figure 11.** Boxplot of MAE for C-RNN-GAN, RGAN, TimeGAN e the real data for the VixStreet<sub>t</sub> dataset.

### 6.3 Use Case Analysis

Considering the case described in Section 5.4, we aim to assess whether synthetic datasets can perform as well as or better than the real dataset in a task that involves predicting future values based on a certain number of past values. Specifically, we use a full day of data to predict the number of bikes rented in the next hour (in the case of Bikesharing<sub>t</sub>) and the total number of cars on the streets in the next 30 minutes (in the case of VixStreets<sub>t</sub>).

Regarding the first case, Figure 10 illustrates the boxplot of MAE for the three models used, as well as the real dataset (baseline). The figure indicates that C-RNN-GAN has the highest MAE, with a median of  $\approx 409$  and an interquartile range of 78. In contrast, the MAE for the other two models is significantly smaller. Specifically, RGAN has an MAE of  $\approx 66$  with an interquartile range of 196, while TimeGAN has an MAE of  $\approx 77$  with an interquartile range of 185. Note that both RGAN and TimeGAN exhibit MAE values slightly lower than the baseline (111). Identifying the exact cause of this is beyond the scope of this article. However, we believe that the primary factor contributing to this result is the potential overfitting of the model trained with the real dataset, despite we applied measures to mitigate it. Furthermore, in our previous results, particularly the qualitative ones shown in Figure 7, we observed that TimeGAN outperformed RGAN in the Bikesharing dataset although RGAN's MAE still was lower. One possible explanation for this is that, as demonstrated earlier, TimeGAN effectively captures weekday patterns but struggles in representing the weekends. On the other hand, RGAN, while not capturing weekdays as effectively, tends to represent weekends slightly better, resulting in fewer errors in the model's predictions.

For the second case, Figure 11 presents the box plot of MAE for the three models concerning the VixStreets<sub>t</sub> dataset. We can see from the figure that both C-RNN-GAN and RGAN have the highest MAE. Specifically, C-RNN-GAN has an MAE of  $\approx 19.28$  with an interquartile range of 0.078, and RGAN has an MAE of  $\approx 19.31$  with an interquartile range of 0.036. Similar to the first case, TimeGAN's MAE is very close to the baseline ( $\approx 0.914$ ). It's important to note that in this second case, only TimeGAN performed satisfactorily. This outcome was expected, as qualitative evaluations had already indicated that the other two models struggled to learn the characteristics of the VixStreet dataset. Furthermore, it's worth noting that TimeGAN's performance exceeded our expectations in this use case. VixStreet itself is a complex dataset, and we anticipated that the MAE calculated

on new data points from the same streets as in the dataset would be higher than the baseline. Thus, when using data from five entirely different streets, we expected the MAE value to be considerably higher

## 7 Conclusions

In this paper, we present MobDeep, a framework for generating and evaluating time series of mobility data features models based on deep learning. MobDeep achieves a reasonable model generalization on different time-series from the datasets. We evaluated MobDeep using a classic and three deep-learning-based models trained using two datasets with distinct characteristics: an open one with information about bicycle sharing and a private one with traffic of cars in the streets of a city.

Our results show that MobDeep can train models to generate synthetic datasets, producing essential evaluations to identify the best model. We also show that deep learning models can capture the main characteristics of the datasets. However, in reproducing datasets with congestion flows, their performance depends on the intensity of cars on the streets. Furthermore, the use cases demonstrated that, in general, synthetic data generated by GANs can be effectively used in real scenarios, often exhibiting performance comparable to or even surpassing that of real datasets.

Considering the current state of MobDeep, it is important to point out some of its limitations. The first one concerns the type of mobility data expected by the framework. Currently, the carried-out experiments consider that the mobility time series correspond to some type of data summarization (e.g., the number of cars on a street in a given time interval). Thus, the framework needs to be evaluated on data containing geographic locations, for example.

Another limitation refers to the models' evaluation, mainly the qualitative analyses, which need a manual evaluation to assess the efficiency of the models. Thus, datasets with many variables can make qualitative assessment very costly. Finally, as we use models that can be suitable to any types of time series, it is important to understand how to optimize the models for generating mobility time series properly.

As additional future research, we can mention the addition of new models in MobDeep, such as ARIMAX and Variational Autoencoders; the evaluation of the framework in datasets with different characteristics, for example, with geolocation information; evaluation of mobile network data generation as well as the performance of mobile network algorithms on the generated data and; study of optimizations to the models used for the specific problem of mobility.

## Declarations

### Funding

This work was financed by CAPES (Finance Code 001), CNPq, FAPES (#2022/ZQX6F, #2021/GL60J, and #2022/NGKM5), and FAPESP/ MCTI/ CGI.br (grant #2020/ 05182-3 and grant #2023/00148-0). Additionally, the authors also would like to acknowledge the data support through the technical cooperation agree-

ment 004/2018, between the Municipal Public Security Secretary of Vitória-Espírito Santo and UFES.

## Authors' Contributions

IFR contributed to the conception, development, and deployment, and writing of this study. GC and AAAR contributed to the analysis of the results and writing. VFMS contributed to the conception of the paper, analysis of the results, and writing of this manuscript. All authors read and approved the final manuscript.

## Competing interests

The authors declare that they have no competing interests.

## Availability of data and materials

A repository containing the framework code, models, and synthetic datasets generated by our framework is available at: <https://github.com/lprm-ufes/MobDeep>.

## References

- Borji, A. (2022). Pros and cons of gan evaluation measures: New developments. *Computer Vision and Image Understanding*, 215:103329. DOI: 10.1016/j.cviu.2021.103329.
- Brock, A., Donahue, J., and Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis. DOI: 10.48550/arXiv.1809.11096.
- Brockwell, P. J. and Davis, R. A. (2009). *Time series: theory and methods*. Springer Science & Business Media. Book.
- Brophy, E., Wang, Z., She, Q., and Ward, T. (2023). Generative adversarial networks in time series: A systematic literature review. *ACM Computing Surveys*, 55(10):1–31. DOI: 10.1145/3559540.
- Cao, Y.-J., Jia, L.-L., Chen, Y.-X., Lin, N., Yang, C., Zhang, B., Liu, Z., Li, X.-X., and Dai, H.-H. (2018). Recent advances of generative adversarial networks in computer vision. *IEEE Access*, 7:14985–15006. DOI: 10.1109/ACCESS.2018.2886814.
- Chollet, F. et al. (2018). *Deep learning with Python*, volume 361. Manning New York. Book.
- Cunha, V. C., Zavala, A. Z., Magoni, D., Inácio, P. R. M., and Freire, M. M. (2022). A complete review on the application of statistical methods for evaluating internet traffic usage. *IEEE Access*, 10:128433–128455. DOI: 10.1109/ACCESS.2022.3227073.
- Esteban, C., Hyland, S. L., and Rättsch, G. (2017). Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*. DOI: 10.48550/arXiv.1706.02633.
- Fanaee-T, H. and Gama, J. (2014). Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2(2):113–127. DOI: 10.1007/s13748-013-0040-3.
- Feng, J., Yang, Z., Xu, F., Yu, H., Wang, M., and Li, Y. (2020). Learning to simulate human mobility. In *Proceedings of the 26th ACM SIGKDD International Conference*



- on Knowledge Discovery 'I&' Data Mining, page 3426–3433, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3394486.3412862.
- García-Jara, G., Protopapas, P., and Estévez, P. A. (2022). Improving astronomical time-series classification via data augmentation with generative adversarial networks. *The Astrophysical Journal*, 935(1):23. DOI: 10.3847/1538-4357/ac6f5a.
- Gomes, M. F., y Piontti, A. P., Rossi, L., Chao, D., Longini, I., Halloran, M. E., and Vespignani, A. (2014). Assessing the international spreading risk associated with the 2014 west african ebola outbreak. *PLoS currents*, 6. DOI: 10.1371/currents.outbreaks.cd818f63d40e24aef769dda7df9e0da5.
- Gonzalez, M. C., Hidalgo, C. A., and Barabasi, A.-L. (2008). Understanding individual human mobility patterns. *nature*, 453(7196):779–782. DOI: 10.1038/nature06958.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc. Book.
- Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., and Alahi, A. (2018). Social gan: Socially acceptable trajectories with generative adversarial networks. DOI: 10.48550/arXiv.1803.10892.
- He, M., Luo, X., Wang, Z., Yang, F., Qian, H., and Hua, C. (2020). Global traffic state recovery via local observations with generative adversarial networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3767–3771. IEEE. DOI: 10.1109/ICASSP40776.2020.9054656.
- Helbing, D., Johansson, A., and Al-Abideen, H. Z. (2007). Dynamics of crowd disasters: An empirical study. *Physical review E*, 75(4):046109. DOI: 10.1103/PhysRevE.75.046109.
- Hillier, B., Turner, A., Yang, T., and Park, H.-T. (2009). Metric and topo-geometric properties of urban street networks: some convergences, divergences and new results. *Journal of Space Syntax Studies*. Available at: <https://discovery.ucl.ac.uk/id/eprint/18583>.
- Ho, S., Xie, M., and Goh, T. (2002). A comparative study of neural network and box-jenkins arima modeling in time series prediction. *Computers Industrial Engineering*, 42(2):371–375. DOI: 10.1016/S0360-8352(02)00036-0.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780. DOI: 10.1162/neco.1997.9.8.1735.
- Huang, Z. and Tatem, A. J. (2013). Global malaria connectivity through air travel. *Malaria journal*, 12(1):1–11. DOI: 10.1186/1475-2875-12-269.
- Iglesias, G., Talavera, E., and Díaz-Álvarez, A. (2023). A survey on gans for computer vision: Recent research, analysis and taxonomy. *Computer Science Review*, 48:100553. DOI: 10.1016/j.cosrev.2023.100553.
- Jauhri, A., Stocks, B., Li, J. H., Yamada, K., and Shen, J. P. (2020). Generating realistic ride-hailing datasets using gans. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 6(3):1–14. DOI: 10.1145/3380968.
- Jeon, J., Kim, J., Song, H., Cho, S., and Park, N. (2022). Gt-gan: General purpose time series synthesis with generative adversarial networks. *Advances in Neural Information Processing Systems*, 35:36999–37010. DOI: 10.48550/arXiv.2210.02040.
- Johansson, A., Helbing, D., Al-Abideen, H. Z., and Al-Bosta, S. (2008). From crowd dynamics to crowd safety: a video-based analysis. *Advances in Complex Systems*, 11(04):497–527. DOI: 10.1142/S0219525908001854.
- Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. DOI: 10.48550/arXiv.1710.10196.
- Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. DOI: 10.48550/arXiv.1609.04836.
- Kitamura, R., Chen, C., Pendyala, R. M., and Narayanan, R. (2000). Micro-simulation of daily activity-travel patterns for travel demand forecasting. *Transportation*, 27(1):25–51. DOI: 10.1023/A:1005259324588.
- Kraemer, M. U., Yang, C.-H., Gutierrez, B., Wu, C.-H., Klein, B., Pigott, D. M., Du Plessis, L., Faria, N. R., Li, R., Hanage, W. P., et al. (2020). The effect of human mobility and control measures on the covid-19 epidemic in china. *Science*, 368(6490):493–497. DOI: 10.1126/science.abb4218.
- Lei, K., Qin, M., Bai, B., Zhang, G., and Yang, M. (2019). Gcn-gan: A non-linear temporal link prediction model for weighted dynamic networks. In *IEEE Conference on Computer Communications*, pages 388–396. IEEE. DOI: 10.1109/INFOCOM.2019.8737631.
- Lenkei, Z. (2018). Crowdsourced traffic information in traffic management: Evaluation of traffic information from waze. Available at: <http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1266883&dswid=-607>.
- Lin, Z., Jain, A., Wang, C., Fanti, G., and Sekar, V. (2020). Using gans for sharing networked time series data: Challenges, initial promise, and open questions. In *Proceedings of the ACM Internet Measurement Conference*, pages 464–483. DOI: 10.1145/3419394.3423643.
- Luca, M., Barlacchi, G., Lepri, B., and Pappalardo, L. (2021). A survey on deep learning for human mobility. *ACM Computing Surveys (CSUR)*, 55(1):1–44. DOI: 10.1145/3485125.
- Malandrino, F., Chiasserini, C., and Kirkpatrick, S. (2018). Cellular network traces towards 5g: Usage, analysis and generation. *IEEE Transactions on Mobile Computing*, 17(3):529–542. DOI: 10.1109/TMC.2017.2737011.
- Mogren, O. (2016). C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*. DOI: 10.48550/arXiv.1611.09904.
- Mota, V. F., Cunha, F. D., Macedo, D. F., Nogueira, J. M., and Loureiro, A. A. (2014). Protocols, mobility models and tools in opportunistic networks: A survey. *Computer Communications*, 48:5–19. Opportunistic networks. DOI: 10.1016/j.comcom.2014.03.019.
- Navidan, H., Moshiri, P. F., Nabati, M., Shahbazian, R., Gho-

- rashi, S. A., Shah-Mansouri, V., and Windridge, D. (2021). Generative adversarial networks (gans) in networking: A comprehensive survey & evaluation. *Computer Networks*, 194:108149. DOI: 10.1016/j.comnet.2021.108149.
- Piorkowski, M., Sarafijanovic-Djukic, N., and Grossglauser, M. (2009). CRAWDAD dataset epfl/mobility (v. 2009-02-24). DOI: 10.15783/C7J010.
- Qu, Y., Yu, S., Zhou, W., and Tian, Y. (2020). Gan-driven personalized spatial-temporal private data sharing in cyber-physical social systems. *IEEE Transactions on Network Science and Engineering*, 7(4):2576–2586. DOI: 10.1109/TNSE.2020.3001061.
- Rao, J., Gao, S., Kang, Y., and Huang, Q. (2020). Lstm-trajgan: A deep learning approach to trajectory privacy protection. DOI: 10.48550/arXiv.2006.10521.
- Ribeiro, I., Castanheira, L., Schaeffer-Filho, A., Cordeiro, W., and Mota, V. (2021). Mobility and community detection based on topics of interest. In *2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*, pages 1–6. IEEE. DOI: 10.1109/CCNC49032.2021.9369462.
- Scott, J., Gass, R., Crowcroft, J., Hui, P., Diot, C., and Chaintréau, A. (2009). CRAWDAD dataset cambridge/haggle (v. 2009-05-29). DOI: 10.15783/C70011.
- Silva, T. H., De Melo, P. O. V., Almeida, J. M., and Loureiro, A. A. (2014). Large-scale study of city dynamics and urban social behavior using participatory sensing. *IEEE Wireless Communications*, 21(1):42–51. DOI: 10.1109/MWC.2014.6757896.
- Smith, T. G. et al. (2017–). pmdarima: Arima estimators for Python. Available at: <http://alkaline-ml.com/pmdarima/>.
- Solmaz, G. and Turgut, D. (2019). A survey of human mobility models. *IEEE Access*, 7:125711–125731. DOI: 10.1109/ACCESS.2019.2939203.
- Song, C., Qu, Z., Blumm, N., and Barabási, A.-L. (2010). Limits of predictability in human mobility. *Science*, 327(5968):1018–1021. DOI: 10.1126/science.1177170.
- Song, H. Y., Baek, M. S., and Sung, M. (2019). Generating human mobility route based on generative adversarial network. In *2019 Federated Conference on Computer Science and Information Systems*, pages 91–99. IEEE. DOI: 10.15439/2019F320.
- Susskind, J., Anderson, A., and Hinton, G. E. (2010). The toronto face dataset. Technical report, Technical Report UTML TR 2010-001, U. Toronto. Non Public Dataset.
- Vallender, S. (1974). Calculation of the wasserstein distance between probability distributions on the line. *Theory of Probability & Its Applications*, 18(4):784–786. DOI: 10.1137/1118101.
- Yin, D., Yang, Q., and Ma, L. (2018). Gans based density distribution privacy-preservation on mobility data. *Sec. and Commun. Netw.*, 2018. DOI: 10.1155/2018/9203076.
- Yin, Y., Lin, Z., Jin, M., Fanti, G., and Sekar, V. (2022). Practical gan-based synthetic ip header trace generation using netshare. In *ACM SIGCOMM*, pages 458–472. DOI: 10.1145/3544216.3544251.
- Yoon, J., Jarrett, D., and van der Schaar, M. (2019). Time-series generative adversarial networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc. Book.
- Yu, H., Li, Z., Zhang, G., Liu, P., and Wang, J. (2020). Extracting and predicting taxi hotspots in spatiotemporal dimensions using conditional generative adversarial neural networks. *IEEE Transactions on Vehicular Technology*, 69(4):3680–3692. DOI: 10.1109/TVT.2020.2978450.
- Zhang, G. P. (2003). Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175. DOI: 10.1016/S0925-2312(01)00702-0.
- Zhang, H. and Lu, X. (2020). Vehicle communication network in intelligent transportation system based on internet of things. *Computer Communications*, 160:799–806. DOI: 10.1016/j.comcom.2020.03.041.
- Zhang, H., Wu, Y., Tan, H., Dong, H., Ding, F., and Ran, B. (2022). Understanding and modeling urban mobility dynamics via disentangled representation learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):2010–2020. DOI: 10.1109/TITS.2020.3030259.
- Zhang, L. (2019). StgGAN: Spatial-temporal graph generation. page 608–609, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3347146.3363462.
- Zheng, Y., Zhang, L., Xie, X., and Ma, W.-Y. (2009). Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of the 18th International Conference on World Wide Web*, page 791–800, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/1526709.1526816.