Universidad Politécnica de Madrid

Escuela Técnica Superior
de Ingenieros de Telecomunicación

Máster Universitario en Ingeniería
de Telecomunicación

Trabajo Fin de Máster

Design and Implementation of a Computing Scenario Forecasting System Based on Generative Adversarial Networks

Jaime Pérez Sánchez

2021

Máster Universitario en Ingeniería de Telecomunicación

Trabajo Fin de Máster

Título: Design and Implementation of a Computing Scenario Forecasting System Based on Generative Adversarial Networks

Autor: D. Jaime Pérez Sánchez

Tutor: Dña. Patricia Arroba García

Departamento: Departamento de Ingeniería Electrónica

Miembros del Tribunal

Presidente: D.

Vocal: D.

Secretario: D.

Suplente: D.

Los miembros del tribunal arriba nombrados acuerdan otorgar la calificación de: .....

Madrid, a de de 2021

Universidad Politécnica de Madrid

Escuela Técnica Superior
de Ingenieros de Telecomunicación

Máster Universitario en Ingeniería
de Telecomunicación

Trabajo Fin de Máster

Design and Implementation of a Computing Scenario Forecasting System Based on Generative Adversarial Networks

Jaime Pérez Sánchez

2021

Resumen

Nuestra sociedad se dirige hacia una transformación digital sin precedentes. La creciente demanda de aplicaciones innovadoras en la Internet de las Cosas y las Ciudades Inteligentes, combinada con la consolidación y expansión de las soluciones existentes basadas en el paradigma de Computación en la Nube, desencadenará una cantidad de tráfico y una demanda de recursos computacionales insólitas. Este contexto, junto con el desarrollo de nuevas redes de comunicación en 5G, se traducirá en unos requisitos de rendimiento excepcionales para los Centros de Datos (DCs).

De acuerdo con un estudio publicado en la revista *Science* [1], entre 2010 y 2018, el número de instancias de computación de los DCs aumentó en un 550%, resultando en un consumo global de energía de alrededor de 205 TWh en 2018, lo que representa alrededor del 1% de la energía consumida en todo el mundo. Teniendo en cuenta que la energía de refrigeración en los DCs representan, en promedio, hasta el 40% del total de la factura [2], esto inevitablemente apunta a una necesidad urgente de mejorar la eficiencia energética en estas instalaciones.

Los métodos de optimización basados en datos están revolucionando el modelado, la predicción, el control y la optimización de sistemas complejos. En particular, el estado del arte muestra un claro dominio de los algoritmos basados en técnicas de Aprendizaje Automático (ML) y Aprendizaje Profundo (DL). Por ejemplo, aplicando dichos algoritmos, los DCs de Google han reducido su consumo total de energía en un 40% [3].

Uno de los principales inconvenientes del uso de técnicas de optimización basadas en ML y DL es que se necesita una enorme cantidad de datos para que los algoritmos converjan a soluciones satisfactorias. En el ámbito de los DCs es difícil, costoso y en ocasiones peligroso recopilar con gran variabilidad, ya que la integridad física del equipo puede ponerse en peligro. Por lo tanto, la generación de datos sintéticos realistas en entornos de computación es crucial para impulsar estas tecnologías de optimización. Este proyecto propone resolver estos retos mediante la generación de datos sintéticos, prediciendo posibles escenarios realistas en un entorno real de Cloud DC.

Para este propósito, utilizaremos una técnica que ha obtenido resultados extraordinarios en los últimos años, las Redes Generativas Adversarias (GAN). Estos algoritmos generativos basados en redes neuronales artificiales se han utilizado comúnmente para generar datos multimedia sintéticos (principalmente imágenes y vídeos). Algunas propuestas en el estado del arte utilizan las GAN para generar datos de series temporales. Sin embargo, estas investigaciones tienen limitaciones significativas, tales como la imposibilidad de manejar datos multivariables y categóricos en la generación de escenarios. En este trabajo, abordamos este problema y presentamos una arquitectura de propósito general para datos de series temporales. Además, evaluaremos el control sobre los datos generados utilizando las GAN para producir escenarios anómalos bajo demanda. De esta manera, se obtendrán datos con una variabilidad más significativa para mejorar los algoritmos de modelado y optimización basados en datos.

Palabras Clave

Redes Generativas Adversarias, Datos Sintéticos, Predicción de Escenarios, Centro de Datos, Datos de Sensores, Optimización Energética, Inteligencia Artificial Aplicada

Summary

Our society is heading towards an unprecedented digital transformation. The growing demand for novel applications on the Internet of Things (IoT) and Smart Cities, combined with the consolidation and expansion of existing solutions based on the Cloud Computing (CC) paradigm, will trigger an abnormal amount of data traffic and computing resources demand. This context, coupled with the development of brand-new communication networks in 5G, will translate into unique performance requirements for Data Center (DC)s.

According to a study published in *Science* [1], between 2010 and 2018, the number of DCs compute instances increased by 550%, resulting in global energy use of about 205 TWh in 2018, which represents around 1% of global electricity consumption. Considering that cooling energy expenses in DCs represent, on average, up to 40% of the total bill [2], this inevitably points to an urgent need to improve energy efficiency in these facilities.

Data-driven optimization methods are revolutionizing the modeling, prediction, control, and optimization of complex systems. In particular, the state of the art shows a clear dominance of algorithms based on Machine Learning (ML) and Deep Learning (DL) techniques. For instance, by applying these algorithms, *Google*'s DCs have reduced its total energy consumption by 40% [3].

One of the main drawbacks in using ML and DL based optimization techniques is that a massive amount of historical data is needed for the algorithms to converge to quasi-optimal solutions. In the DCs' scope, it is difficult, expensive, and sometimes even dangerous to gather data from all kinds of real situations, as the equipment's physical integrity may be at risk. Thus, the generation of realistic synthetic data in computing environments is crucial in enabling these optimization technologies. This project proposes to solve these challenges by generating synthetic data, forecasting possible realistic scenarios in a real Cloud DC environment.

For this purpose, we will use a technique that has given extraordinary results in recent years, the Generative Adversarial Networks (GAN). These generative algorithms based on artificial neural networks have been commonly used to create realistic synthetic multimedia data (mainly images and videos). Some proposals in the state of the art use GAN to generate synthetic time-series data. However, this research has significant limitations, such as the limitation of handling multi-variable and categorical data in the generation of scenarios. In this work, we address this problem and present a complete general-purpose architecture for complex time-series datasets. Moreover, this research will evaluate the control over the generated data samples using GAN to produce on-demand anomalous but realistic scenarios. Thus, obtaining data with more significant variability to improve data-driven modeling and optimization algorithms.

Keywords

Generative Adversarial Networks, Synthetic Data, Scenario Forecasting, Data Center, Sensor Data, Energy Optimization, Applied Artificial Intelligence

"What I cannot create, I do not understand."

— RICHARD FEYNMAN

Contents

List of Acronyms

AI Artificial Intelligence.

AR Autoregressive.

ARIMA Autoregressive Integrated Moving Average.

BCE Binary Cross-Entropy.

BSTS Bayesian Structural Time Series.

CC Cloud Computing.

CNN Convolutional Neural Networks.

DC Data Center.

DL Deep Learning.

EMD Earth Mover's Distance.

GAN Generative Adversarial Networks.

GPU Graphics Processing Unit.

ICT Information and Communications Technology.

IoT Internet of Things.

IT Information Technology.

KL Kullback-Leibler.

LSTM Long Short-Term Memory.

MCMC Markov Chain Monte Carlo.

MH-GAN Metropolis-Hastings Generative Adversarial Network.

ML Machine Learning.

MSE Mean Squared Error.

NIPS Neural Information Processing Systems.

PUE Power Usage Effectiveness.

RMSE Root Mean Squared Error.

SGD Stochastic Gradient Descent.

TTUR Two Time-Scale Update Rule.

VAE Variational Auto-Encoders.

WGAN Wasserstein GAN.

WGAN-GP Wasserstein GAN with Gradient Penalty.

W-Loss Wasserstein Loss.

List of Figures

List of Tables

Chapter 1

Introduction and Objectives

1.1 Digital Economy & Society, Growth and Concerns

Our society is immersed in an unstoppable digitalization and interconnection process, a transformation so vast that none of the revolutions we have experienced throughout history can serve as a precedent. This digital transformation is driven by groundbreaking technologies such as the Internet of Things (IoT), Big Data, and Artificial Intelligence (AI) as part of the development of Smart Cities and brand-new 5G communication networks. According to *Cisco*'s Annual Internet Report [9] in 2018, 51% of the world's population used the Internet, and it is expected to reach 66% in 2023. The number of devices connected to IP networks was 1.6 times the world population in 2018 and will be 3.6 times in 2023. In Figure 1.1, we can observe the share of the population using the Internet in 2017.

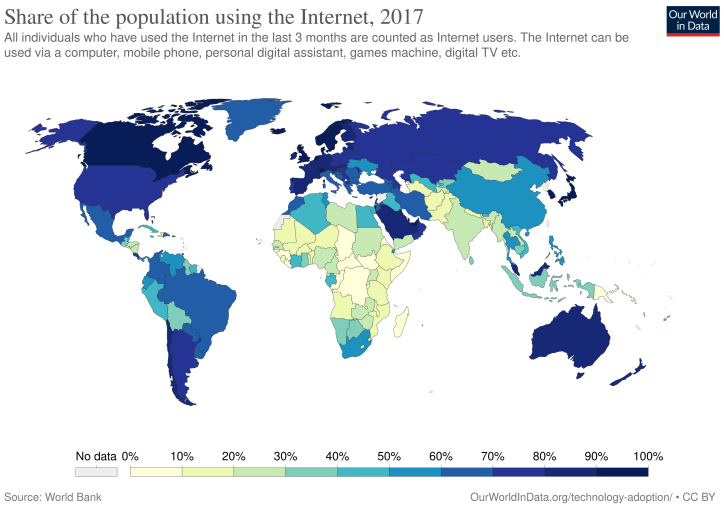


Figure 1.1: Share of the population using the Internet, 2017. All individuals who have used the Internet in the last three months are counted as internet users. Source: World Bank Data.

From an economic standpoint, the integration of Information and Communications Technology (ICT) brings additional innovations and organizational changes to businesses. According to the latest statistics published by the *European Commission* on the digital economy and society [10], 1 out of 5 EU companies made e-sales in 2018, and more than 1 in 3 used e-business integrations such as Enterprise Resource Planning and Customer Relationship Management.

Nevertheless, this digitalization trend is much more significant among households and individuals for two main reasons. Firstly, only 48% of European citizens between 16 and 74 used the Internet daily in 2014, although it reached 73% in 2019 [11]. Secondly, according to data from *Sandvine* [12], more than 75% of global Internet traffic is generated by video streaming, gaming, and social media applications. Figure 1.2 shows the evolution of the hours spent daily on digital media in the United States from 2008 to 2018.

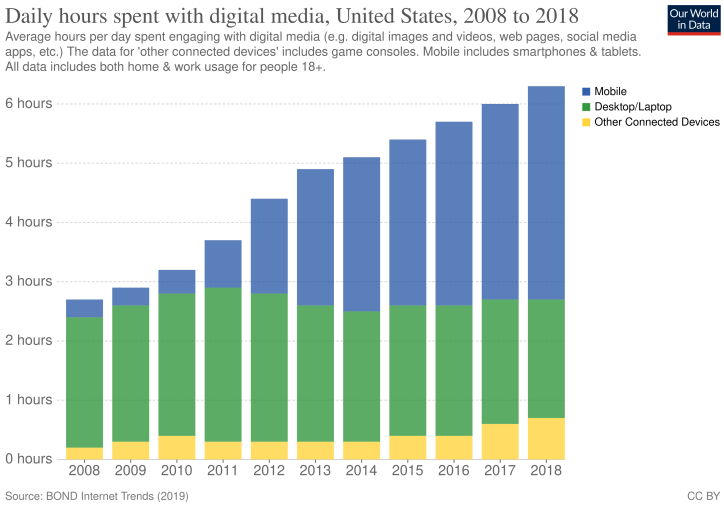


Figure 1.2: Daily hours spent with digital media, United States, 2008 to 2018. Source: World Bank Data.

The crucial key we must understand from the above statements is that the vast majority of these technologies base their operations on the Cloud Computing (CC) paradigm, particularly the more complex and data-intensive ones (e.g., video streaming and social media). From 2014 to 2018, the use of Cloud solutions in business increased significantly, particularly in the largest companies (+21%) [13]. The ratio of traffic used by social networks and video streaming will continue to increase, both in terms of the number of users and the offered content's resolution. Figure 1.3 shows the evolution of the number of users of social media platforms.

This context, together with the demand for novel applications in Internet of Things (IoT), autonomous vehicles, and Smart Cities, will lead to an unprecedented amount of data traffic and requests for computing resources in the Cloud, making much more apparent and worrying the inefficiencies Data Center (DC)s have. E. Masanet *et al.* [1] point out that between 2010 and 2018, the number of DCs compute instances increased by 550% and IP traffic by 1,000%. Resulting in global energy

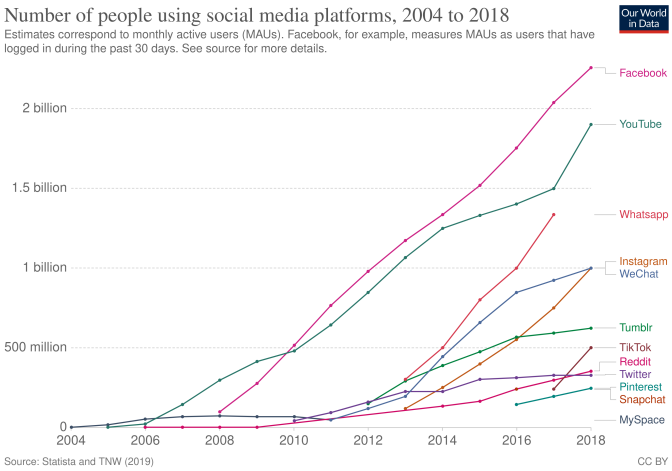


Figure 1.3: Number of people using social media platforms, 2004 to 2018. Source: World Bank Data.

use of about 205 TWh in 2018, which represents around 1% of global electricity consumption. An article published in *Nature* [4] suggests that by 2030, ICT's total electricity demand may reach 20.9% of the global market, being DCs the fastest growing sector (Figure 1.4).

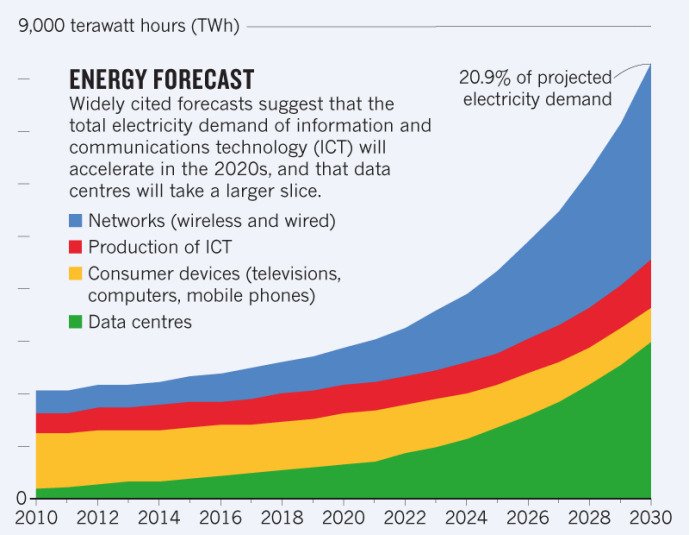


Figure 1.4: ICT energy consumption forecast. Source: Nature [4]

Despite this, evidence shows global progress in DC energy efficiency [1], led by industry leaders such as *Google* and *Facebook*. While compute instances had increased more than sixfold, overall electricity use per computation had dropped by a factor of four, resulting in only a 25% increase in overall energy use since 2010. The global average Power Usage Effectiveness (PUE) had decreased to 1.67 [14] (ideal is 1). Figure 1.5 shows the trends of global energy-use drivers in DCs between 2010 and 2018.

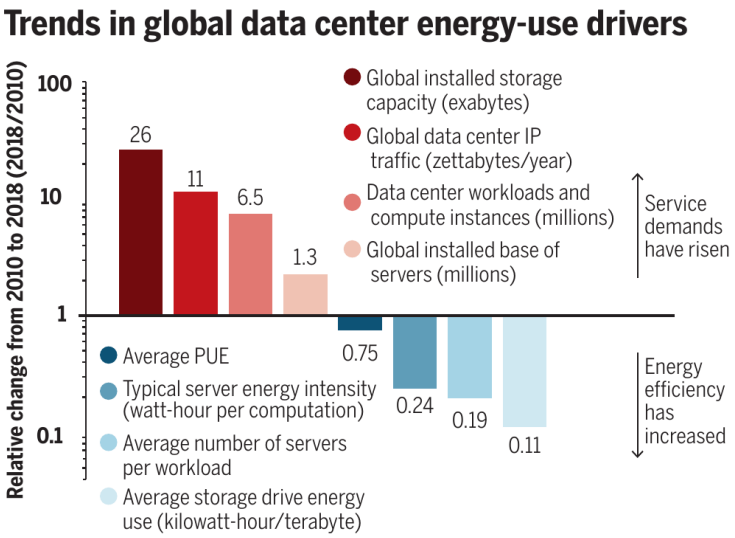


Figure 1.5: Trends in global data center energy-use drivers. Source: Science [1]

Yet, given the growing demand for DC services, How much longer can these current efficiency trends last? Predicting the long-term efficiency limits of Information Technology (IT) devices is notoriously tricky, and the next doubling of global DC compute instances may occur within the next 3 or 4 years [15]. There remains a lot more to do, given the world's increasing need for computing power. L. Castellazzi *et al.* [2] evidenced in 2017 that cooling energy expenses in DCs represent, on average, up to 40% of the total bill.

Growth in energy use has slowed, owing to the efficiency gains that smart policies can help maintain in the short term. However, much larger public data and modeling capacities are urgently required to understand and monitor DC energy use, and design effective optimization policies [1]. Data-driven optimization methods have proven to be the most accurate for modeling, predicting, controlling, and optimizing complex systems. We will address this topic in the next section.

1.2 Data-driven Operation and Optimization in Data Centers

The exponential growth of technological progress has brought exceptional advantages in recent decades, but at a high cost: systems and processes are becoming more complex. Diagnostic operation and optimization have become essential elements of complex systems and processes to ensure their productivity, efficiency, reliability, availability, and avoid expensive maintenance. Nowadays, any optimization strategy must begin with data. Data-driven science and engineering methods are revolutionizing the modeling, prediction, control, and optimization of complex systems. The other approach to data-based models is explicit or physical models. That in addition to being more expensive and time-consuming, the more sophisticated a system is, the less accurate it becomes.

In recent years, the discipline of data-based optimization methods has been dominated, both in terms of research progress and practical applications, by the techniques embedded in Machine Learning (ML). Besides being cheaper than other techniques, its use has allowed the implementation of more proactive policies that can be applied in real-time, giving a considerable competitive advantage to the systems that implement it. Even more recently, a discipline within ML has had exponential growth in its performance, Deep Learning (DL). Based on artificial neural networks, this type of algorithm has superseded state of the art in many complex problems, even outperforming humans. Three enabler reasons make this degree of accuracy possible: (i) Easy access to massive amounts of data; (ii) Increased computing power with Graphics Processing Unit (GPU); (iii) Transfer learning from pre-trained models built by experts and better mathematical understanding.

The modern DC is composed of a multitude of mechanical, electrical, and control systems communicating dynamically. It is incredibly challenging to understand and optimize energy efficiency due to the sheer number of potential operating configurations and nonlinear interdependencies. Furthermore, the most classic optimization alternatives cannot quickly adapt to internal or external changes (like weather), and each DC facility has a unique architecture and environment. Therefore, an intelligent general framework is needed to understand all nonlinear interactions. Many fronts are targeted for optimization within the DC, for instance:

- Anticipate changing needs from customers and users
- Risk analysis scenarios through simulations
- Improved uptime reliability
- Power Management
- Cooling Efficiency
- Computation Usage
- Security policies
- Workload distribution

Data-driven optimizations have proven to capture the better ever-changing and extremely volatile nature of these large infrastructures [16]. Modern ML, DL, and analytics tools can highlight anomalies and correlate to potential inefficiencies in real-time to predict failures. As reported by *DeepMind* [3], *Google*'s Artificial Intelligence (AI) research laboratory, the use of AI-driven solutions in its DC facilities have yielded cooling energy savings of 40%, which equates to a 15% reduction in overall PUE. It produced the lowest PUE ever seen in a *Google* facility. In any large scale energy-consuming environment, this would be a vast improvement. Given how sophisticated *Google*'s DCs are already, it is a phenomenal step forward. In Figure 1.6, we can observe the test of the ML recommendations on a live *Google*'s DC.

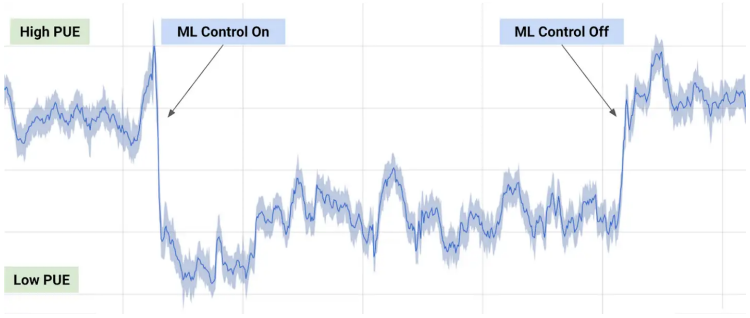


Figure 1.6: *Google DeepMind* testing ML recommendations in a real DC. Source: *DeepMind* [3]

The main challenge to apply this approach globally is that these extraordinary AI-driven results are only possible through massive amounts of data collected by thousands of sensors in each DC, and these data are not publicly available. Gathering these amounts of data has other associated problems. Most notably, they are very costly in terms of time, resources, and budget. In practice, the benefits of data-driven solutions are restricted to those who own the data and can collect them. Besides, to generate robust and reliable models, we need heterogeneous data that include risk situations, which may endanger the electronic equipment's integrity. Hence, this work's primary motivation is to develop a methodology to generate realistic synthetic data, thus increasing the amount and diverseness of public data, essential for the novel operation and optimization techniques in DCs.

1.3 Synthetic Data Generation

Interest in synthetic data has been increasing over the last few years, driven by two simultaneous trends. The first is the demand for large amounts of data to build and train AI and ML models. In certain domains, this data may be challenging to find, expensive, or involve privacy issues for users (e.g., medical imaging). The second involves recent work that has demonstrated effective methods for generating high-quality useful synthetic data. Companies like *NVIDIA* [17], *IBM* [18] [19], *Google* [20], and agencies such as the *US Census Bureau* [21] have adopted different data synthesis methodologies to support model building, application development, and data dissemination. In Figure 1.7, we can appreciate remarkable examples of *NVIDIA*'s results in the augmentation of autonomous vehicle datasets, using a technique known as style transfer.

At a conceptual level, synthetic data are generated from real samples and have similar statistical properties. The degree to which a synthetic dataset is an accurate approximation for the real one is a utility and realism measure. Real data are complex and messy, and data synthesis needs to work within that context. In some cases, open data may lack sufficient heterogeneity for robust training models, not capturing rare cases well enough.

Figure 1.7: *NVIDIA* Street image translation. For each pair, the left image is the input, and the right is the translated image. Source: *NVIDIA* [5]

We can highlight two essential benefits of data synthesis. First, providing more efficient access to data is essential for developing more reliable data-driven models. The *Government Accountability Office* [22] and the *McKinsey Global Institute* [23] note that accessing data to build and test AI models is a challenge for their adoption more broadly. An analysis from *Deloitte* [24] concludes that data-access issues are ranked in the top three companies' top challenges when implementing AI. Data synthesis can provide realistic data to work with, efficiently, and at a scale. Second, enabling better analytics when access to real data is too costly, dangerous, or unethical. For instance, in autonomous driving vehicles, medical data, or the generation of extreme case scenarios in DC's electronic equipment, which would be a use case in this work. Another valuable scenario is when actual data are not accessible yet. Hence, synthetic data are used to train an initial model, significantly accelerating the convergence and potentially increasing the final model's robustness.

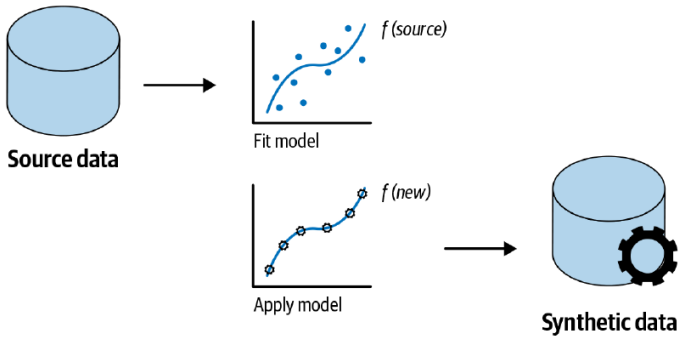


Figure 1.8: Conceptual process of data synthesis using a model. Source: Practical Synthetic Data Generation - *O'Reilly*

Statistical imputation methods (e.g., Synthetic Minority Over-sampling Technique (SMOTE), Adaptive Synthetic Sampling (ADASYN)) are widely used in unbalanced datasets. However, nowadays, the approach producing state-of-the-art results is to use a model that fits the available real data and use it to generate new samples (Figure 1.8). The challenge is that if the model is different from the eventual models that will be built with the synthetic data, then the imputed values may not be very reflective of the real ones, introducing errors. The risk of building the wrong model has led to historic caution in the application of data synthesis. Advances in statistical ML and DL have managed to achieve models capable of capturing the distributions and complex relationships between variables with great precision. Therefore, we are getting closer to the point where generative models create datasets that accurately reproduce real data.

Over the last few years, we have seen the adoption of synthetic data grow in various industries, such as manufacturing and distribution [17], healthcare [25], transportation [26] [5], and financial services [27]. Since data-access challenges are not likely to get easier anytime soon, it is expected that the applicability of this approach will grow, including more use cases. The benefits of synthetic data can be dramatic; it can make impossible projects doable, significantly accelerate AI initiatives, or result in a material improvement in the outcomes of AI projects. For all these reasons, this MSc Thesis will apply a generative algorithm that has given extraordinary results in recent years, the Generative Adversarial Networks (GAN), to forecast realistic computing scenarios in a DC facility.

1.4 Project Motivation and Objectives

Numerous researchers use GANs for data augmentation on image datasets, achieving outstanding results in recent years. The work presented in this MSc Thesis uses GANs to generate synthetic time-series data, which has been scarcely explored by academia and has extraordinary potential in numerous real-world scenarios.

Our research's final objective is to design and implement a general-purpose scenario forecasting system based on GANs that empowers a better generation of realistic synthetic time-series data. To demonstrate our work's usefulness, we will address a use case in the DC field for improved analysis and modeling of energy use in these complex systems. However, this approach can be employed in any similar time-series-like problems in other fields of application. To the best of our knowledge, no ongoing projects are using advanced synthetic data augmentation techniques in the field of DC optimization. In this work, we will use real data collected from an operating DC of the company *Adam Data Centers*, located in Navalcarnero (Madrid), during the summer of 2019. *Adam Data Centers* is a relevant supplier in the DC sector with 30 years of experience. The startup company *TycheTools* developed the sensors utilized to gather the data.

The main objectives of this MSc Thesis are the following:

- Provide a comprehensive study of GANs and their possible applications in DC optimization.
- Extend the state-of-the-art methodology to produce synthetic scenarios using GANs, enabling the use of multi-variable and categorical data.
- Generate on-demand anomalous synthetic situations in computation systems, coherent with available real data.
- Generate realistic synthetic data to enable better analytics and models for DC optimization.

This MSc Thesis is divided as follows:

- Chapter 2, analyzes the state of the art for synthetic data generation in DC and time-series data. Moreover, we will provide a short overview of the theory behind GANs.
- Chapter 3 presents a concise explanation of the case study, covering the methodology used.
- Chapter 4 provides a comprehensive analysis of the conducted experiments and the obtained results.
- Chapter 5 reflects the proposal's conclusions, explores its limitations, and recommends coming steps to stay ahead.

Chapter 2

Literature Review

This chapter analyses the state-of-the-art solutions found in the literature to generate synthetic time-series data. We will also make a brief review of the theory supporting GANs and discuss the possible advantages and disadvantages of their use in artificial data generation. To our knowledge, there are no ongoing projects involving synthetic data in the DCs' field. Thus, we will not cover it in this work.

2.1 Synthetic Time-Series Data Generation

Time-series data is a particular type of data in which there is a time dependency. It is ubiquitous in many businesses and applications (e.g., stock market, weather patterns, electricity demand), critical in the DC industry. There are several limitations in obtaining massive and highly representative amounts of this data type in real-world scenarios. For these reasons, the creation of synthetic temporal data has been an open challenge in industry and academia. If successfully achieved, it would have a dramatic potential to unlock cross-organizational and cross-industry collaboration to solve some of the biggest problems at a world scale.

Nevertheless, generating synthetic time-series data that preserve time dependency is far more challenging than producing other data types such as images or tabular data. The synthetic tabular data assumes that a single row stores all the information from a particular event. Sequential data, on the other hand, has time-sensitive information spread across many rows and columns. The length of these sequences is often variable, and many historical events potentially condition each data point across multiple variables. Therefore, small errors can be propagated through the entire sequence introducing considerable deviations.

Statistical methods for time-series data generation have been extensively studied. Unfortunately, many of these efforts have resulted in low quality and limited flexibility of the generated data. In many cases, the proposed models were designed for each specific problem, thus requiring specialized domain knowledge. We will now examine some of the fundamental approaches on which most time-series data generation models are based.

Naive Approaches

The most commonly used traditional methods of time-series data augmentation include Gaussian noise addition [28] [29], rotation [28] [30], scaling [28] [30], warping [28] [31] and permutation [28]. Brian Kenji *et al.* [32] recently conducted an empirical survey comparing the results of classical data augmentation methods for time series classification with neural networks. These methods are useful in many problems and can be used in conjunction with other more complex methods, but are very limited by the available data.

Autoregressive (AR) Models

AR models are dynamic stationary processes where each point in the time-series is represented as a function of the previous n steps. Non-linear AR variants such as Autoregressive Integrated Moving Average (ARIMA) are powerful and extensively used approaches in time-series forecasting [33] [34] [35] [36]. However, AR models have a fidelity problem. That is, they produce simplistic models that are unable to capture complex long-term temporal correlations.

Markov Models

Markov Models are a general approach for modeling dynamic stochastic categorical systems. It is assumed that future states depend only on the current state, not on the events that occurred before it. Variants such as Hidden Markov Models have been used to model time series distributions [37] [38] [39] [40] [41]. Like AR models, the major limitation of these methods is that they cannot capture complex long-term dependencies.

Bayesian Models

Bayesian statistics are widely used in time series modeling [42] [43] [44] [45]. They are ideal for noting an event that occurred and predicting the probability that any one of several known possible causes was the contributing factor. For example, it could represent the probabilistic relationships between diseases and symptoms. Dynamic Bayesian Networks are a particular variety of Bayesian networks that can perform sequential data modeling. Other thriving models include the Bayesian Structural Time Series (BSTS) and Bayesian Model Averaging. These approaches' main limitations are that they require manually defining the prior probabilities and cannot capture complex long-term relationships.

Dynamic Stationary Processes

These methods work by representing each point in the time series as the sum of deterministic processes and adding random noise. It is a popularly used approach

for modeling sequences with methods like bootstrapping [46] [47] [48] [49]. However, it is critical to incorporate expert knowledge of long time dependencies into the model, making it unsuitable for complex datasets with unknown correlations.

These more classical models we have examined are transparent and straightforward so that they can be understood clearly, and it is possible to derive their properties in a rigorous statistical way. Besides, they can be applied in small datasets and still get good results. The general weakness of these methods is that significant expertise knowledge is required for successful results, and these approaches do not easily spread across workloads and use cases. Furthermore, their simplicity results in not always improved outcomes when more data are available. By definition, they are built to handle linear dynamics. Thus, they will do a poor job when non-linear relationships are dominant.

In recent years methods based on Generative Adversarial Networks (GAN) have emerged as a popular technique for augmenting datasets, with particular prominent results in images and videos. Many papers have started to emerge from applying other types of data and proposals for generic time-series generation frameworks. We will now make a brief review of the use of GANs in the field of time series.

GANs have been proposed on multiple occasions for time-series forecasting since, due to their nature as a generative algorithm, they may understand the complex nature of sequential data. The models from the discriminative paradigm attempt to learn the conditional probability distribution of the labels Y given the observations x symbolically expressed as $P(Y|X = x)$. On the other hand, generative models such as GANs directly estimate the conditional probability distribution of the data X given the observations x, symbolically expressed as $P(X = x)$. Xingyu Zhou *et al.* [50] proposed a generic framework employing Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN) for adversarial training to forecast high-frequency stock market data. Kang Zhang *et al.* [51] also studied the application of GANs in stock market forecasting with excellent results. Unfortunately, these two works do not compare their results with each other, only with more classical state-of-the-art methods (e.g., ARIMA) and supervised neural networks. The application of GANs in prediction and modeling has also been successfully used in sensory time-series data [52] [53].

Concerning the data augmentation approach, the ground-breaking publication was presented by Cristobal Esteban *et al.* [54]. The authors demonstrated how GANs could be used to produce realistic-looking heterogeneous medical time-series data. Related solutions to generate synthetic data have been successfully applied in the energy sector [55] [56] [57], sensory data [58], and health applications [59] [60] [61]. The state-of-the-art paper so far in the augmentation of synthetic time-series data was *TimeGAN* [62]. Nevertheless, a promising recent proposal by Zinan Lin *et al.* [63] explores enabling a generic framework (*DoppelGANger*) to augment time-series datasets based on GANs, where minimal expert knowledge is needed. They address some general problems in the field, such as the abstraction of long-term correlations and non-linearities between categorical and continuous variables.

Research	Data Augmentation	Single-variable Scenario Generation					*Ours*
	[63]	[66]	[67]	[68]	[69]	[70]	
Realistic Data Generation	✓	✓	✓	✓	✓	✓	✓
Scenario Generation	✗	✓	✓	✓	✓	✓	✓
Generation from particular time instants	✗	✓	✗	✗	✗	✗	✓
Generation of on-demand anomalous situations	✗	✗	✗	✗	✗	✗	✓
Multi-variable Generation	✓	✗	✗	✗	✗	✗	✓
Introduce Categorical Variables	✓	✗	✗	✗	✓	✗	✓
Introduce Spatial Information	*Not Tested*	✓	✓	✓	✓	✓	✓
Disentangled Latent Space	✓	✓	✓	✓	✓	✓	*Future Work*

Table 2.1: State of the art comparison in synthetic time-series data generation using GANs

Moreover, they successfully demonstrate their solution in various datasets and real-world use cases. However, this approach has significant limitations, as it is only capable of generating data consistent with the real data. It cannot generate on-demand anomalous situations, making the generated data variability limited and biased by the available data. Besides, it cannot generate data from a given moment in time or a specific category.

Our work focuses on a field within time series generation that has been scarcely explored by academia: scenario forecasting. The main advantages of scenario generation over punctual forecasting and general data augmentation are that it shows the predictions' uncertainty and reflects the data's time dependency. The information provided by these generated scenarios is valuable for a host of decision-making and stochastic optimization problems, such as those encountered in a DC facility. Some publications address scenario generation through statistical methods [64] [65], although these approaches require extensive knowledge of the specific problem and have several limitations in the obtained results.

The emergence of GANs has provided a renewed impulse to this field. In 2018, Yize Chen *et al.* [66] proposed a novel data-driven scenario forecasting approach of energy systems (e.g., wind, solar, load). This model-free approach produces a set of realistic scenarios based only on historical data observations. The methodology is based on training a GAN that learns the data's intrinsic patterns. After that, given a real time-series, an optimization problem is performed to obtain the optimal noise inputs that produce realistic future scenarios. To the best of our knowledge, this approach has only been used in the energy sector [66] [67] [68] [69] [70]. Congmei Jiang *et al.* [67] introduce a crucial concept that we will use in our work, the spatial-temporal prediction of scenarios. We also find particularly exciting the work presented recently by Ji Qiao *et al.* [70] in which they use orthogonal regularization techniques in the latent space, making each dimension of this space represent differentiated characteristics in the scenario generation. We suggest this concept for future research on this work. Nonetheless, the works studied present essential limitations that need to be addressed.

Our main contribution to the state of the art is the conditional generation of multi-variable scenarios using GANs. Our approach allows on-demand generation of scenarios and anomalous situations, in particular time instants and categories. Table 2.1 summarizes the state-of-the-art limitations and the contributions of our proposal. To demonstrate the usefulness of the presented work, we will address a case study of sensors in a DC facility. The generation will be conditioned by two inputs: the specific sensor's ID (categorical variable) and the last hours' historical data in that same sensor (time-series variable). Our work is designed to handle complex multivariate time-series datasets with fixed discrete features and ever-changing continuous features. In the following section, we will examine the GANs in-depth, unravel their inner workings, and analyze the advantages and disadvantages of their use in synthetic data generation.

2.2 Review of Generative Models, and the promise of Generative Adversarial Networks

Machine Learning (ML) is a subfield of Artificial Intelligence (AI) that arises from the scientific study of algorithms and statistical models, which computer systems use to perform certain tasks without specific instructions. Instead, they use pattern recognition and inference learned from example data to accomplish these tasks. ML algorithms are widely used in several applications, such as image recognition, medical diagnosis, translation between languages, specialized advertising, and optimization of production processes. Deep Learning (DL) is likewise a subfield of ML, based on Artificial Neural Networks. These Deep Neural Networks are computer systems inspired, but not identical, to the biological networks of neurons that constitute animal brains and nervous systems. This approach aims to extract high-level features from raw data inputs, and in recent years have demonstrated cutting-edge results in numerous fields of application such as computer vision, speech recognition, or natural language processing.

There are two main approaches in statistical modeling and ML, known as generative and discriminative models (Figure 2.1). Let us suppose a model with input data features X and target classes Y to explain the difference. The discriminative models, most commonly used in ML, attempt to learn the conditional probability distribution of the target Y given an observation x, symbolically expressed as $P(Y|X = x)$. On the other hand, generative models estimate the conditional probability distribution of the observable data X, given a target y, symbolically expressed as $P(X|Y = y)$. For unlabeled data, the generative model estimates the probability distribution of the observations expressed as $P(X = x)$.

A significant advantage of generative models is that it can approximate the probabilistic distribution of both the features input data and the target data through the Bayes' theorem. We can create a model that behaves in the same way as a discriminative model making predictions and create new input features that infer the same probability distribution of the real data. Andrew Y. Ng *et al.* [71] empirically demonstrated that discriminative algorithms, which directly estimate

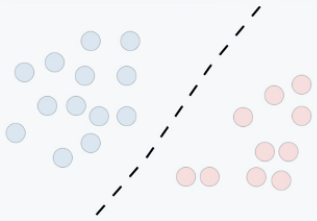
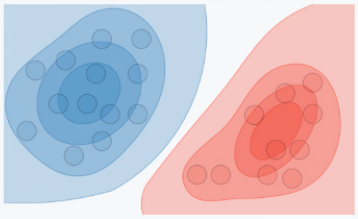
	Discriminative model	Generative model
Goal	Directly estimate $P(y\|x)$	Estimate $P(x\|y)$ to then deduce $P(y\|x)$
What's learned	Decision boundary	Probability distributions of the data
Illustration		
Examples	Regressions, SVMs	GDA, Naive Bayes

Figure 2.1: Comparison between Discriminative and Generative models in ML

$P(Y|X)$, significantly outperformed generative ones in classification tasks. However, generative models' most significant advantage is in creating synthetic data, which has many applications in the ML and real-world contexts. Some examples of popular generative models are Naive Bayes, Boltzmann Machines, Deep Belief, Networks, Hidden Markov Models, VAE, among others.

With some general simplifications, generative models operate by computing the maximum likelihood (Equation 2.1). That is, measuring the log-probability that the model's density function $p_{model}(x|\Theta)$ assigns to all the training data points and adjusts the parameters theta Θ to maximize its value. The way they achieve this goal of computing the maximum likelihood, is what differentiates some generative models from others. In Figure 2.2, we find a taxonomy of the main generative models.

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{x\sim p_{\text{data}}} \log p_{\text{model}}\left(\boldsymbol{x} \mid \boldsymbol{\theta}\right) \tag{2.1}$$

DL's field belonged almost exclusively to the discriminative group until 2014 when Ian Goodfellow *et al.* [72] proposed a new type of architecture, the Generative Adversarial Networks (GAN). Despite their short lifetime, GANs have demonstrated an incredible ability to generate data with very realistic features, and impressive new applications are being researched each year. Yann LeCun, director of AI research at *Facebook* and professor at New York University, famously said that *"Generative Adversarial Networks [...] and the variations that are now being proposed is the most interesting idea in the last ten years in Machine Learning"*. In Figure 2.3, we can find a famous image that illustrates the evolution of the results on face generation produced by the GANs in just four years.

In Figure 2.4, we observe the structure of the original GAN proposal [72]. It is composed of two Neural Networks, Generator and Discriminator, which confront each other to compete during the training phase in a "game" (in the sense of game theory), usually in the form of a zero-sum game. The Generator network $G(z)$ obtains input samples z from a random noise distribution $p_z(z)$ and attempts to generate realistic data samples to fool the Discriminator. The data generated by

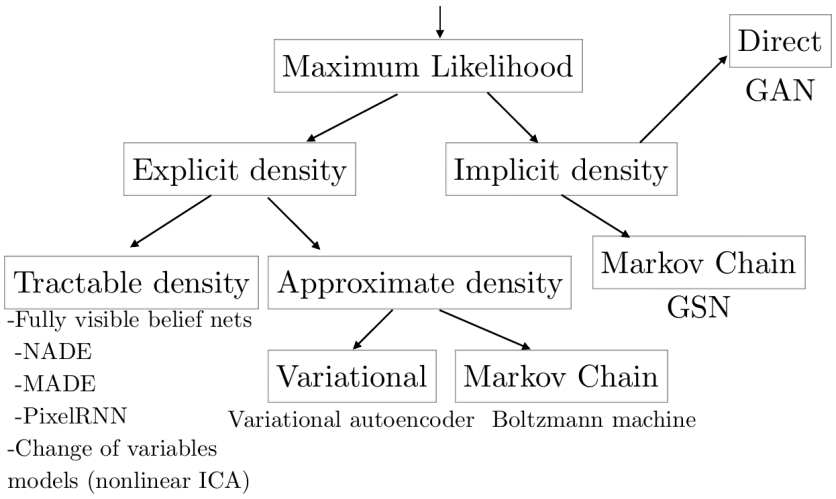


Figure 2.2: Taxonomy of generative models. Source: Ian Goodfellow, NIPS Tutorial, 2016

Figure 2.3: Progress on face generation by GANs in four years. Source: Ian Goodfellow *Twitter* account

the Generator, together with samples x from the real data distribution $p_{real}(x)$, is fed into the Discriminator network $D(x)$. The Discriminators' task is to correctly distinguish between the fake data generated by $G(z)$ and the real data in a binary classification problem. Like Variational Auto-Encoders (VAE) or Boltzmann Machines, GANs use a latent code where all possible synthetic samples are encoded. Usually, the dimension of z has to be larger than the dimension of X for successful decoding. GANs are also asymptotically consistent, unlike variational methods. If unlimited data is available, and the model achieves the balanced saddle point (*Nash equilibrium*), it will eventually recover the actual distribution that generates the data.

The GAN training phase is a two-player game, known as a minimax game. This function (Equation 2.2) expresses the Binary Cross-Entropy (BCE) between the Discriminator predictions and the correct labels in a binary classification task of discriminating real data from fake data. This cost function has two differentiated terms, one relevant to each class. The first term expresses the Discriminator's log-probability to accept as "Real" the data samples from the actual data distribution $p_{real}(x)$. The second term expresses the log-probability that the Discriminator

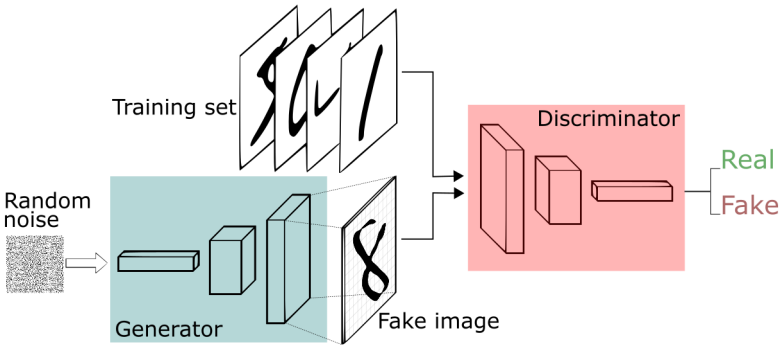


Figure 2.4: Generative Adversarial Network architecture. Source: *FreeCodeCamp*

correctly identifies as "False" the data generated by sampling from the random distribution $p_z(z)$. The Discriminator attempts to maximize the objective function making predictions close to 1 for real samples and close to 0 for fake ones. The Generator operates the other way around, minimizing the objective function such that the predictions for fake samples are close to 1. Therefore, we seek a balance in the game, a saddle point known as *Nash equilibrium*.

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log \left(1 - D_{\theta_d} \left(G_{\theta_g}(z) \right) \right) \right] \tag{2.2}$$

In practice, to avoid convergence and saturation problems, each of the players has its own independently parameterized cost function (Equations 2.3 and 2.4). Specifically, a negative term is added to ensure that functions are always greater than or equal to 0. Therefore, the Generator now only has to maximize the log-probability that the Discriminator will make a mistake when fed with fake data samples. In Figure 2.5, we can visualize the values that the Discriminator cost function takes depending on its predictions. The left figure shows the case where the label is 1 (i.e., a real sample). The right figure shows the opposite case where the label is 0 (i.e., a fake sample). If the predictions are close to the labels, the function takes values close to 0. However, it approaches infinity when the labels and the predictions differ.

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log(1 - D(G(\boldsymbol{z}))) \tag{2.3}$$

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log D(G(\boldsymbol{z})) \tag{2.4}$$

To train a GAN, we alternate training on the Discriminator and the Generator. When training the Discriminator (Figure 2.6), we feed it with real data X and Generated data $\hat{X}$. Next, the Discriminator makes predictions $\hat{Y}$ about each input's probabilities being real or fake. We then compute the cost and update the parameters using a Stochastic Gradient Descent (SGD) based optimizer, typically *Adam* [73]. To train the Generator, we also need to use the Discriminator in the

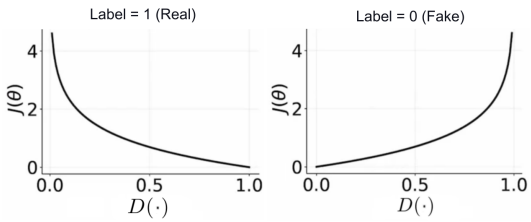


Figure 2.5: Values of the Discriminator cost function depending on its predictions. The left figure shows the case where the label is 1 (i.e., real sample), and the right figure shows where the label is 0 (i.e., fake sample). Source: *deeplearning.ai*

process (Figure 2.7). First, we produce fake data $\hat{X}$ and pass it to the Discriminator. We compute the cost with the obtained predictions by setting all the labels to 1 (i.e., Real). Once we have done this, we only have to update the Generator network parameters to maximize the Discriminator prediction using an SGD-based optimizer.

It is crucial to keep in mind that both networks must improve "simultaneously", and we should keep them at similar skill levels from the beginning of the training. The reasoning behind all this is that if we have an extraordinarily better Discriminator, it will be impossible for the Generator to know in which direction to improve. The same happens in the opposite case where the Generator is extraordinarily better. This problem is known as vanishing gradients. In practice, GAN training is a complex and costly task. Some alternative proposals and tricks make the training a much more stable process and produce better results. We will go deeper into this topic in the section 3.2.

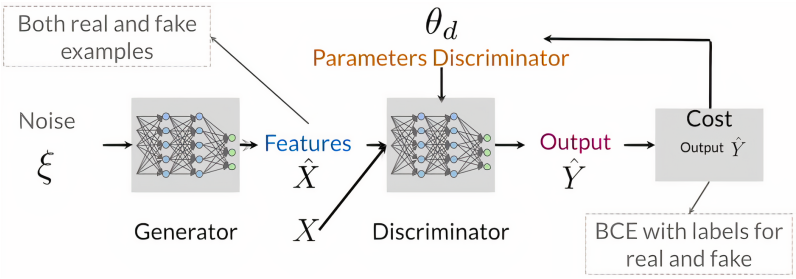


Figure 2.6: GAN training process: Training the Discriminator. Source: *deeplearning.ai*

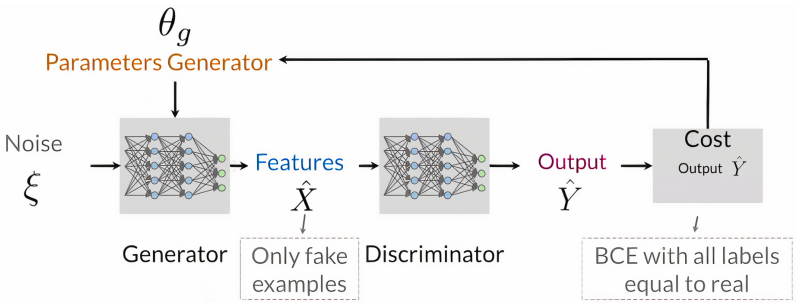


Figure 2.7: GAN training process: Training the Generator. Source: *deeplearning.ai*

The main advantages of using GANs over other generative algorithms are the impressive empirical results of the generated data fidelity and a fast inference to generate data much faster than other generative algorithms. In Figure 2.8, we can find a chart presented by Brett K. Beaulieu-Jones *et al.* [6], where they show the accuracy of different ML models trained with real data, and it is compared with their performance training exclusively with synthetic data produced by a GAN. We observe that training with GAN data approximates quite well the accuracy using real data. Furthermore, the use of GANs to generate synthetic data has several additional **advantages**:

- It usually produces better data than hand-crafted synthetic examples.
- It enables flexible tune generation, which is very useful in imbalanced datasets.
- The use of synthetic data in discriminative algorithms improves its generalization and performance.
- Protects the privacy of real patients' data (e.g., medical applications) and provides anonymity (e.g., stigmatized groups, assault victims, witnesses, activists).
- It can encourage data-sharing between institutions and organizations.
- Synthetic data is less expensive and more abundant than real data.
- This technique can be applied in conjunction with traditional data augmentation methods.

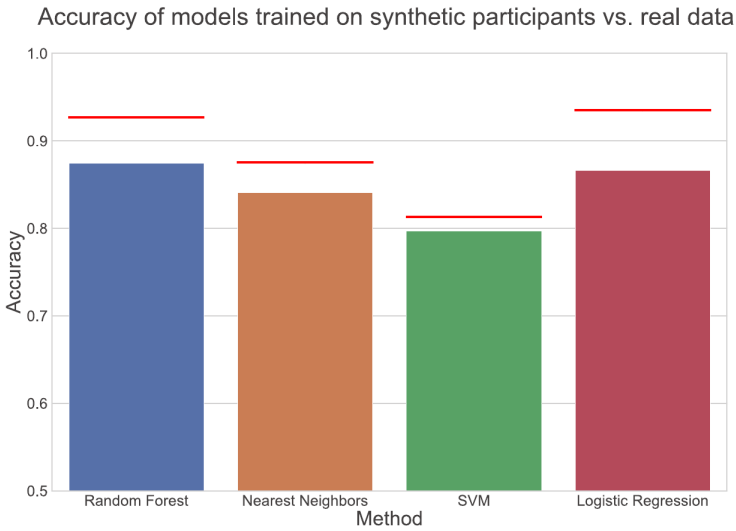


Figure 2.8: Accuracy of models trained on synthetic GAN data *vs.* real data. Source: [6]

It is also essential to consider the **drawbacks** of using GANs to address them with other methods and make the results more meaningful. One of the main ones is the lack of intrinsic evaluation metrics, which is critical for image generation, often requiring manual human evaluation. Other disadvantages are, for instance:

- Unstable training, although this has been mostly solved with new architectures and loss functions like Wasserstein Loss.
- The diversity of the data generated is limited by the training data available.
- It needs relatively large amounts of data for efficient training.
- Sometimes fake data overfits the real data, which will not be useful for its use in other models and does not preserve privacy (Post-processing may help avoid this data leakage).
- Unlike other generative models, it does not compute an explicit density estimation.
- Inverting the generation process is not straightforward and could be useful in some applications like image editing. However, there are proposals such as *CycleGAN* [74] that partially solve this problem.
- The generation could be biased from the data and the metrics used during the training process.
- It can be used in an unethical way (e.g., DeepFakes)

Chapter 3

Scenario Forecasting in Data Centers

This chapter describes the DC case study of scenario forecasting using sensor data. To this end, we will analyze the collected data and then describe the methodology used to develop the algorithms within this work. In the methodology section, we will also explain the improvements incorporated into the original proposal of GANs and the metrics we will use to compare the conducted experiments' outcomes.

3.1 Case Study: Sensor Data

In this work, we will use real sensor data gathered from an operating data center of the company *Adam Data Centers*[1] to demonstrate our contribution's usefulness. *Adam Data Centers* is a neutral data center provider with 30 years of experience, 18 of them managing critical infrastructure at their facilities in Madrid and Barcelona. The data was collected during the summer of 2019 at *Adam Data Centers* facility in Navalcarnero (Madrid). The sensors were designed and developed by the startup company *TycheTools*[2]. In Figure 3.1, we find photos of the sensors used. This company was born in 2019 from a department at the *Escuela Técnica Superior de Ingenieros de Telecomunicación* of the *Universidad Politécnica de Madrid*.

DC facilities are primarily based on air-based cooling solutions, following the Hot Aisle / Cold Aisle strategy [75]. It consists of lining up the racks (i.e., computing units) to face cold air intakes and hot air exhausts alternatively, and hence, forming cold and hot aisles (Figure 3.2). The air conditioning units cool down the heated air from the hot aisles and then reintroduce it to the cold aisles using the ventilation ducts located on the room's floor. The sensors used at *Adam Data Centers*'s facility were placed equally in the cold and hot aisles. To model a DC facility's behavior

[1] *Adam Data Centers* [`https://adam.es/data-center/`]
[2] *TytheTools* [`https://www.tychetools.com`]

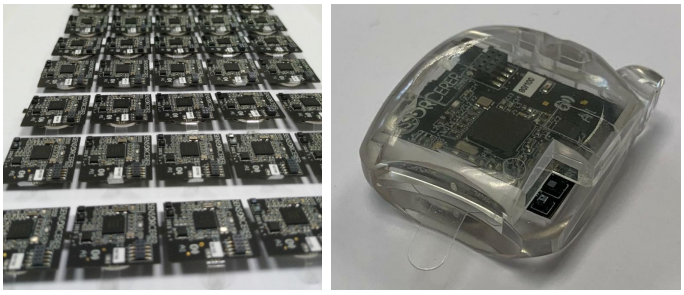


Figure 3.1: Photos of *Tychetools*' sensors.

is crucial to monitor as many system variables as possible. However, temperature and relative humidity are some of the most relevant and straightforward variables to capture. Besides studying the cooling system dynamics, there is a clear correlation between these variables and the total power consumed [76] [77]. Since the sensors were still under development when the data was collected, and considering them sufficiently relevant, only relative humidity and temperature were captured. These will be the variables to be used in this work. However, following our approach, the number of variables can be easily expanded in the future.

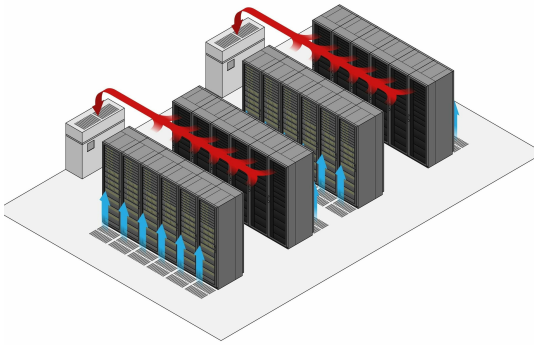


Figure 3.2: Data Center cooling system following the Hot Aisle / Cold Aisle strategy. Source: Illustration by Zern Liew

After preliminary data cleaning, we have a total of 35 sensors available. The measurements are taken every 10 minutes and sent to the gateway that inserts them into the database. In Figure 3.3, we can observe the data collected by one sensor chosen at random. By accumulating the data from all the sensors, we can display them in a histogram (Figure 3.4). We can recognize that the samples are spread mainly in three Gaussian-like distributions, with different proportions, means, and standard deviations. The maximum registered temperature is 46.94 ºC and the minimum 8.14 ºC. In Figure 3.4, we also observe the distribution of temperature data, showing with colors the contribution of each sensor. If we compute each sample's difference with its following (i.e., the derivative), we can observe the distribution in a histogram (Figure 3.5). Temperature seems to have high inertia, so most of these jumps between samples are concentrated around the value 0. The maximum increase value corresponds to a rise of 11.91 ºC, and the minimum corresponds to a fall of 9.79 ºC.

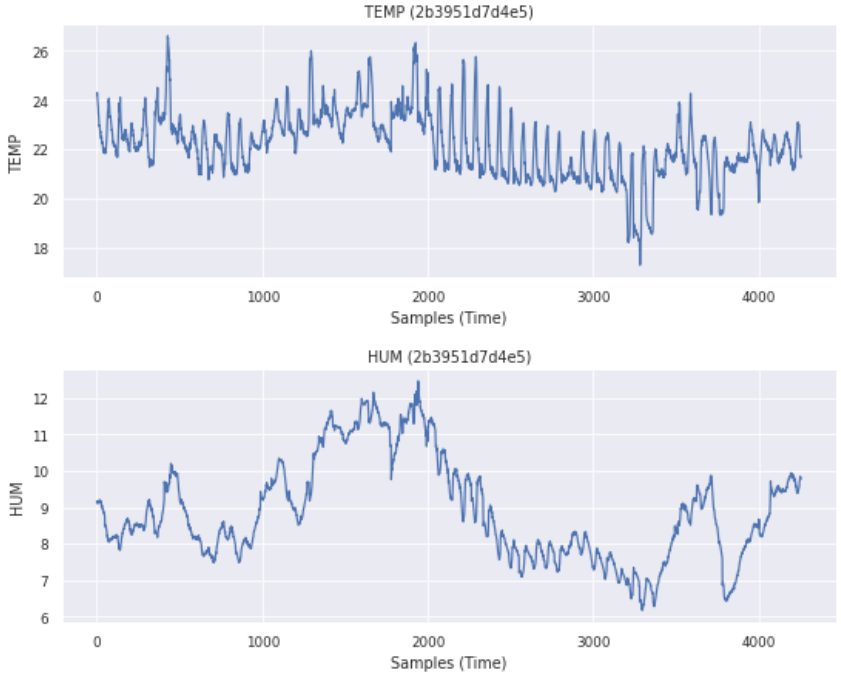


Figure 3.3: Plot of the data collected by one sensor. Samples are collected every 10 minutes.

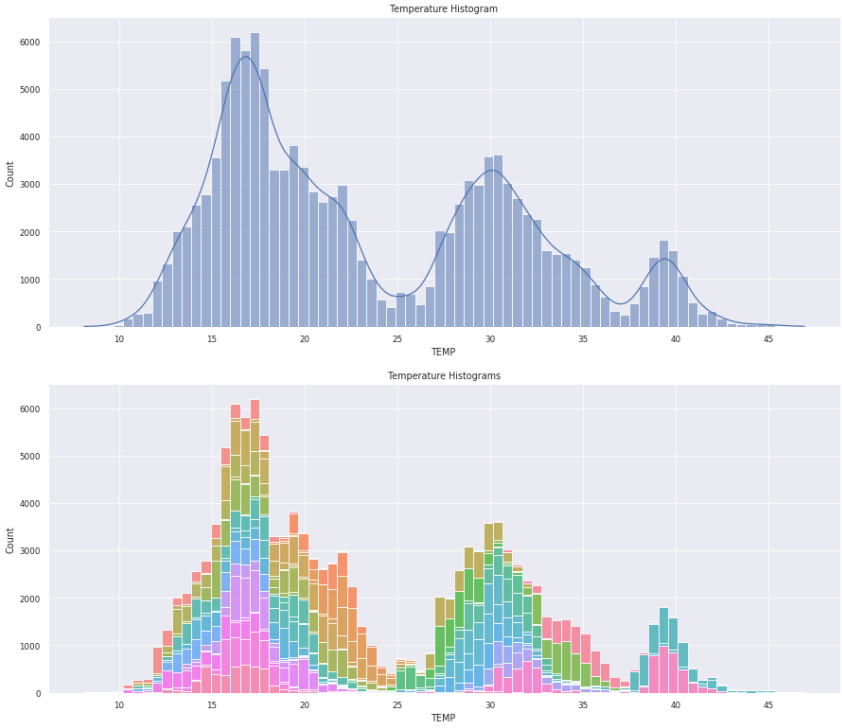


Figure 3.4: Temperature data histogram. The colors represent the contribution of each sensor.

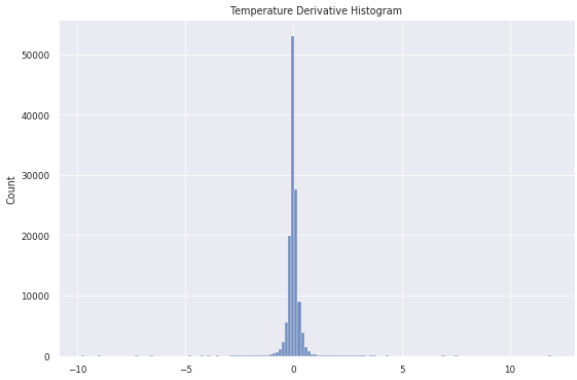


Figure 3.5: Derivative histogram of temperature data.

By performing the same observations to the relative humidity data (Figure 3.6), we can notice that the distribution is less clear than in the temperature, although specific points of greater concentration do exist. The maximum recorded relative humidity is 23%, and the minimum is 3%. Figure 3.6 also shows with colors the contribution of each sensor to the total distribution. By computing the difference between each sample with the following one (i.e., the derivative), we can ascertain that the relative humidity also has most of these jumps concentrated around the 0 value (Figure 3.7). The maximum jump corresponds to a rise of 2.64%, and the largest drop is 3.59%.

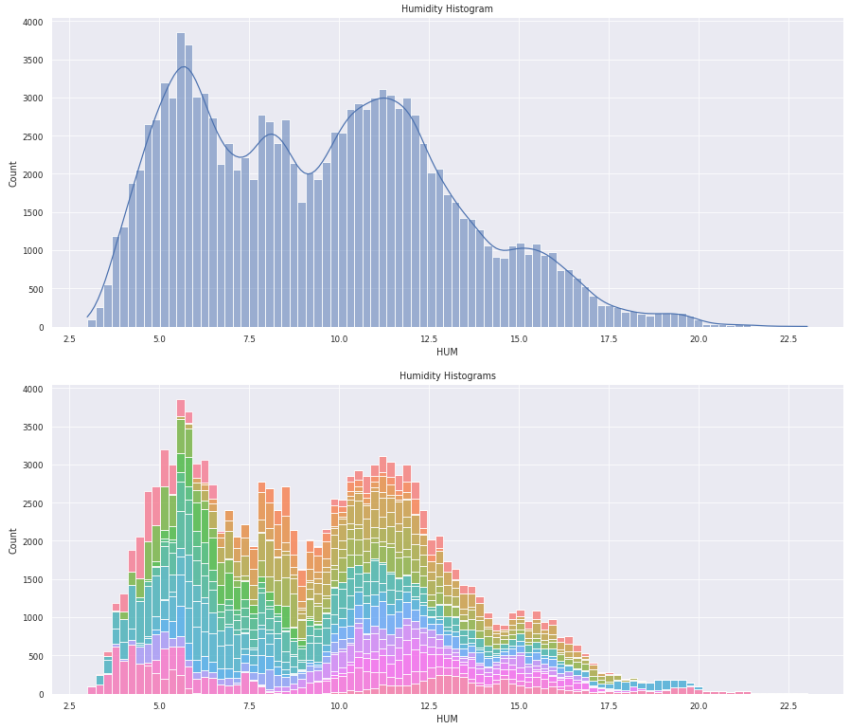


Figure 3.6: Relative humidity data histogram. The colors represent the contribution of each sensor.

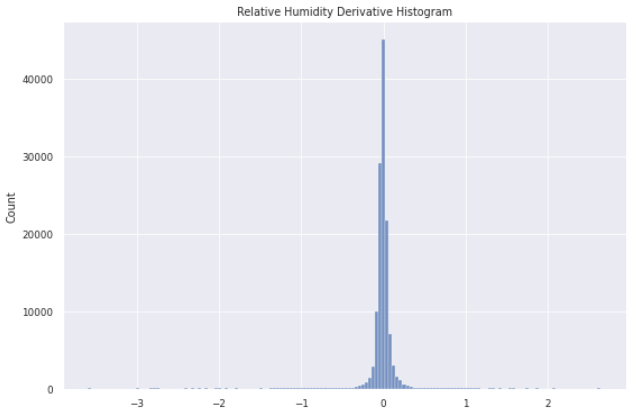


Figure 3.7: Derivative histogram of relative humidity data.

Figure 3.8 displays a scatter plot comparing the accumulated temperature and relative humidity data. We find that there is some correlation between these two variables. If the water vapor content stays the same and the temperature drops, the relative humidity increases. In the same situation, if the temperature rises, the relative humidity decreases. This phenomenon occurs because humidity has an impact on the thermal conductivity of the air, also affecting the power consumption of cooling systems [77]. The figure 3.8 also shows each variable's box plots, a method of representation through the quartiles information. Figure 3.9 shows the same data represented in Figure 3.8 but computing the Kernel Density Estimation, allowing us to identify the areas with the highest samples' concentration.

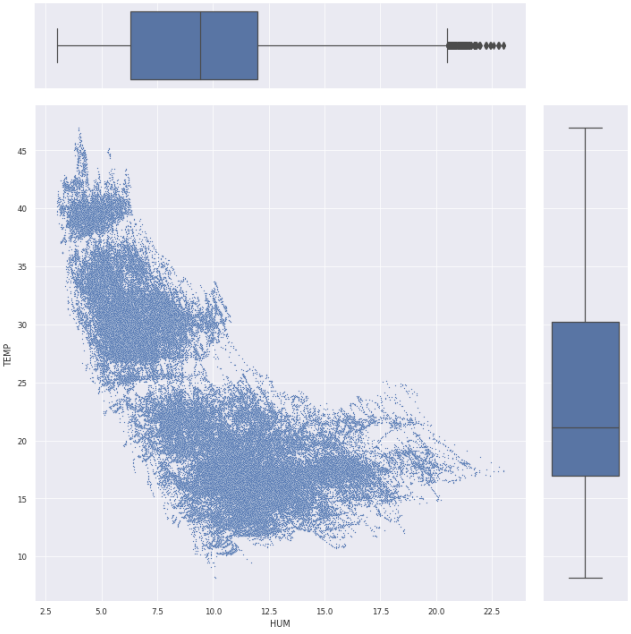


Figure 3.8: Plot confronting Relative Humidity and Temperature data. The box plots on the sides represent the quartile information.

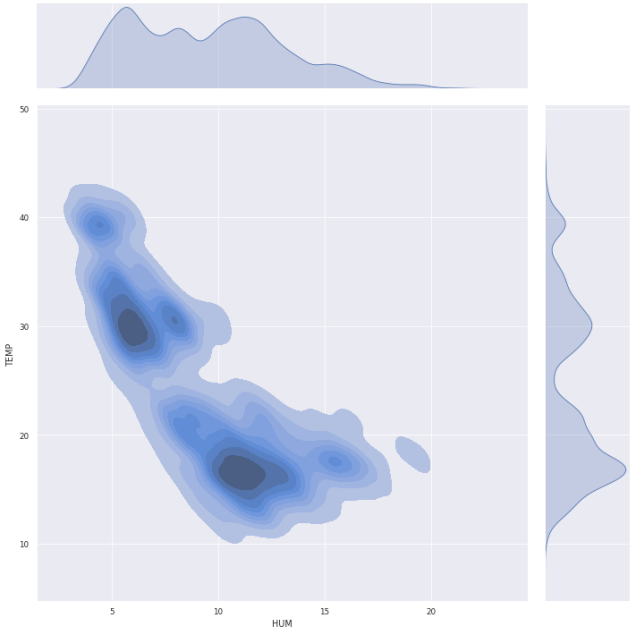


Figure 3.9: Kernel Density Estimation plot confronting Relative Humidity and Temperature data.

Figure 3.10 shows the differences between each sample and the following (i.e., the derivative) of both variables on a scatter plot. Again, some correlation can be noted, although it is very concentrated around the 0 value. To check if this correlation is predictive in any sense, Figure 3.11 compares the previous figure's same data but with a temporal lag in one of the variables. We observe that any possible correlation is then lost, suggesting a simultaneous correlation between temperature and relative humidity may exist. Neither of these variables serves as a robust temporal predictor of the other.

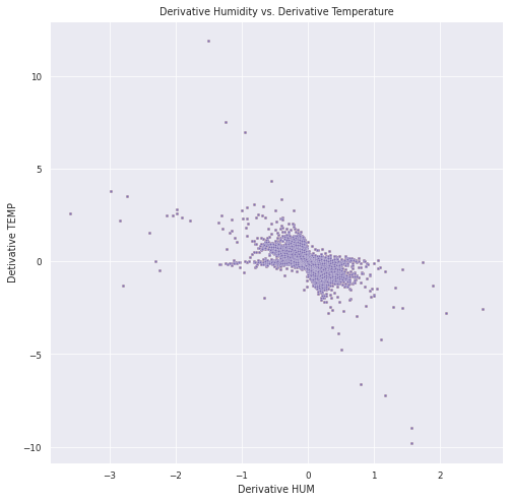


Figure 3.10: Derivatives scatter plot confronting Relative Humidity and Temperature data.

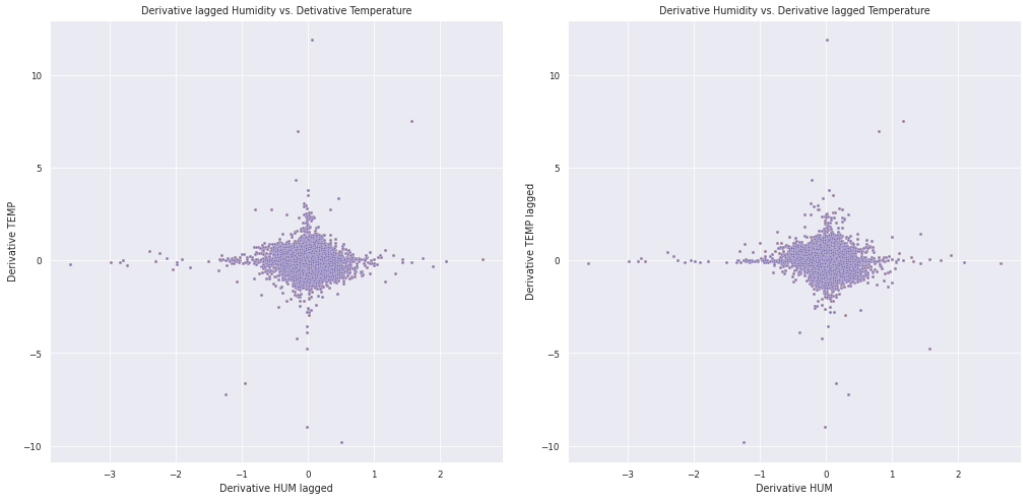


Figure 3.11: On the left: Plot confronting Relative Humidity and Temperature derivatives, with Relative Humidity data lagged one step. On the right: Plot confronting Relative Humidity and Temperature derivatives, with Temperature data lagged one step.

In a time series, autocorrelation indicates the linear relationship between its points as a function of their time difference. A time series's partial autocorrelation is the partial linear relationship with itself for a given lag, removing all the shorter lags' correlation values. The principal difference between these two statistical measures is that the partial correlation shows which data are genuinely informative and harmonic over shorter periods.

Figure 3.12 shows the autocorrelation and partial autocorrelation of the temperature measurements gathered by two different sensors. The two figures on top show a sensor where the temperature autocorrelation starts with high values and then gently decreases to a value lower than 0.4. The temperature partial autocorrelation shows us that the first two lags are much more informative than the rest. The two figures below show a sensor with a temperature autocorrelation that remains very high despite the increase in lag. Therefore, the partial correlation indicates that almost only the first lag is highly informative. The rest of the lags give redundant information. Figure 3.13 shows the autocorrelation and partial autocorrelation of the relative humidity measurements in two different sensors. Generally, it seems that relative humidity has a higher autocorrelation than temperature, which means it is less likely to change. In both sensors, the partial autocorrelations indicate that only the first two lags provide meaningful information.

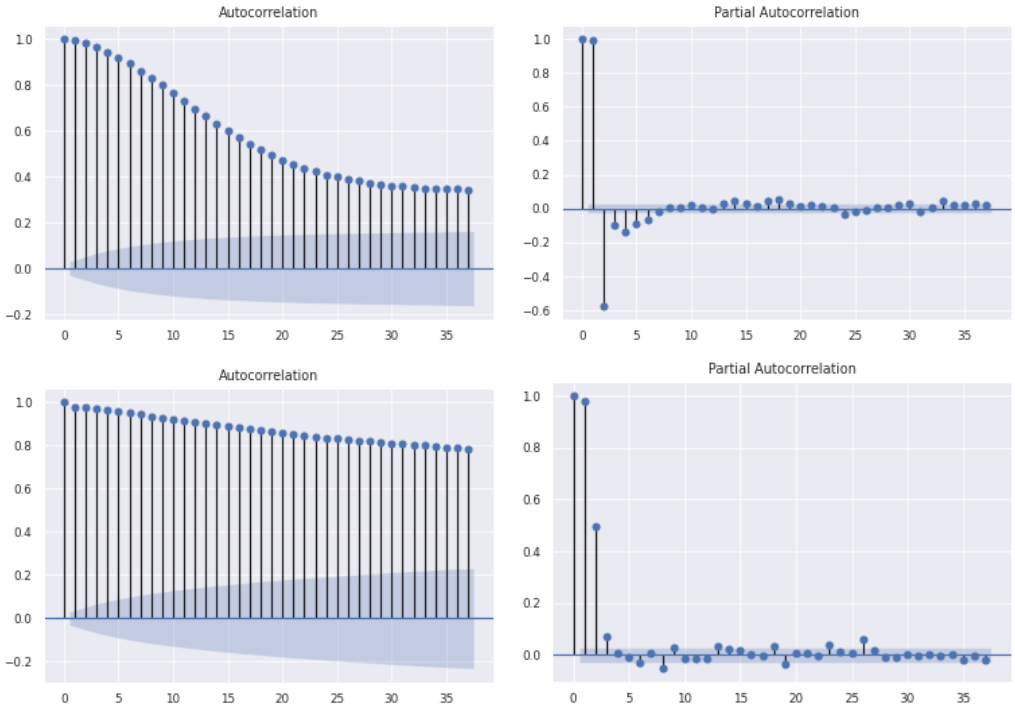


Figure 3.12: Autocorrelation and partial autocorrelation of temperature measurements from two different sensors.

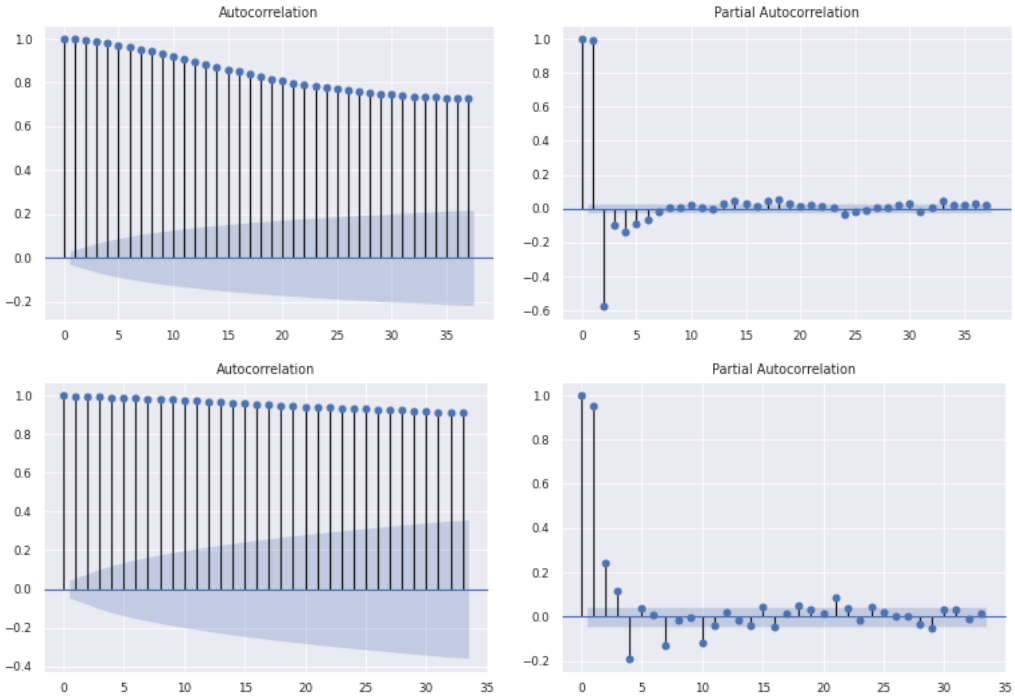


Figure 3.13: Autocorrelation and partial autocorrelation of relative humidity measurements from two different sensors.

3.2 Methodology

In this section, we will explain the improvements introduced to the original proposal of the GANs to stabilize training, the proposed final architecture, a method for improving the sampling of the latent space, and the evaluation metrics used to compare the outputs.

3.2.1 GAN Training Improvements

As explained in section 2.2, training GANs consists of a minimax game between the Generator and the Discriminator networks. This process can be translated into a more general objective of the whole GAN, to make the distribution of the generated data look similar to the real data distribution. In practice, training a GAN is a complex and unstable process. During the last few years, plenty of proposals have been made to solve some of its most common problems.

One of the most challenging GAN failure problems is the vanishing gradients, which consists of the following. The task to be performed by the Generator network is much more complicated, so in most cases, the Discriminator becomes better promptly. In the beginning, this is not a severe problem because the Discriminator can give useful feedback to the Generator. However, if the Discriminator becomes too good at doing its job, the Binary Cross-Entropy (BCE) cost function approaches flat regions (Figure 3.14). Thus the gradients are very close to 0, and the feedback is no longer useful to the Generator.

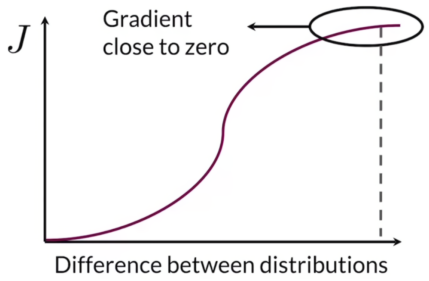


Figure 3.14: Vanishing gradients problem. Source: *deeplearning.ai*

Other main problems suffered by GANs are those related to mode collapse. A mode in a data distribution is an area with a high concentration of observations. Real-world datasets are usually multimodal. Mode collapse occurs when the Generator gets stuck around a single-mode and cannot generate samples from the other modes. Mode dropping happens when the Generator fails to represent all the modes from the real data, meaning that some of them are missing. Figure 3.15 shows an example of a mode collapse.

The ilustrations in Figure 3.15 represent the Generator's heat map along with several steps during the training, and the final column presents the real data distribution. We can notice how the Generator rotates through the different modes trying to fool the Discriminator without ever converging or producing samples of

different modes simultaneously. The use of the BCE cost function in GAN training leads to critical problems such as vanishing gradients and mode collapse. In the following section, we will review some of the most successful proposals to mitigate these issues.

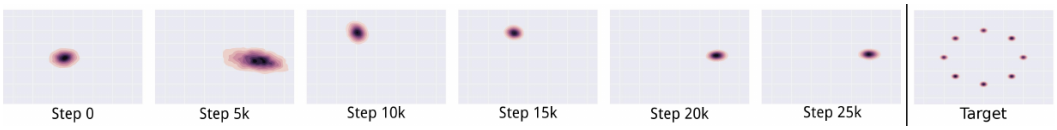


Figure 3.15: Mode Collapse in a GAN. The last column shows the real data distribution. Source: [7]

3.2.1.1 Wasserstein GAN with Gradient Penalty

In 2017 Martin Arjovsky *et al.* [78] presented the Wasserstein GAN (WGAN), a novel GAN architecture that aimed to solve some of the stability problems we have studied. Their work's most significant contribution is a new cost function, the Wasserstein Loss (W-Loss), that attempts to approximate Earth Mover's Distance (EMD). With the W-Loss, there is no longer a limit between 0 and 1 on the outputs of the Discriminator. Instead, it continues to grow regardless of how apart the distributions are, and it does not have flat regions. The W-Loss solves the vanishing gradients problem and reduces the likelihood of mode collapse. Figure 3.16 provides an illustrative comparison between the BCE and W-Loss cost functions.

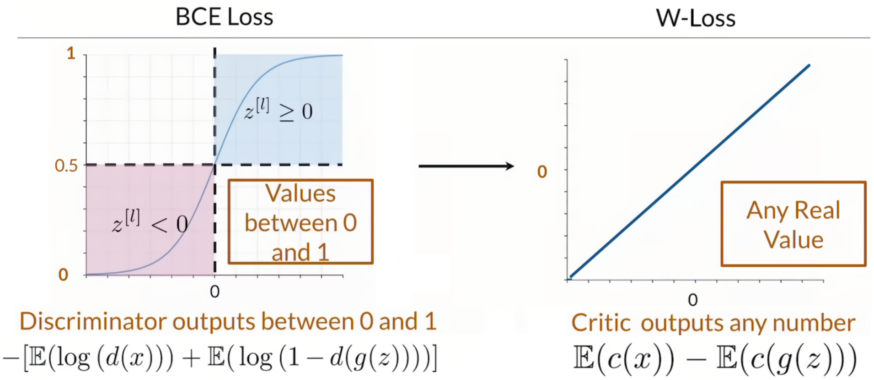


Figure 3.16: Comparison of BCE Loss and W-Loss. Source: *deeplearning.ai*

It is crucial to note that WGAN requires a particular condition on the Discriminator network for stable training: the W-Loss cost function must be 1-Lipschitz continuous. For a function to be 1-Lipschitz continuous, the gradient norm (i.e., slope) must be less than or equal to 1 at all points (Figure 3.17). This condition is critical because it ensures that the W-Loss function is continuous, differentiable, grows linearly, and correctly approximates the EMD. If this condition is met, the training will remain stable.

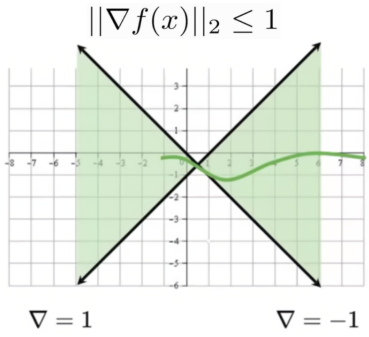


Figure 3.17: 1-Lipschitz continuity condition of a function. Source: *deeplearning.ai*

To ensure that 1-Lipschitz continuity is satisfied, the WGAN authors suggested the Weight Clipping method [78]. They propose forcing the Discriminator network's weights to a fixed interval after updating through gradient descent algorithm. It needs intensive hyperparameter tuning to perform correctly. Ishaan Gulrajani *et al.* [79] proposed a successful alternative: the Wasserstein GAN with Gradient Penalty (WGAN-GP). Consist in sampling some points by interpolating (using a random number epsilon) between real and fake samples, and it is on the critics' prediction of these interpolated images that you want the 1-Lipschitz continuity condition to be met. Empirically Gradient Penalty method tends to produce higher quality results more stably. For its implementation, only a regularization factor needs to be added to the W-Loss cost function (Equation 3.1).

$$L_d = \underbrace{\mathbb{E}[D(x)] - \mathbb{E}[D(G(z))]}_{\text{Original critic loss}} + \underbrace{\lambda \cdot \mathbb{E}_{\hat{x} \sim p_{\hat{x}}} \left[\left(\|\nabla_{\hat{x}} \left(D(\hat{x}) \|_2 - 1 \right)^2 \right] \right.}_{\text{Gradient penalty}} \\ \text{with } \hat{x} = \epsilon x + (1 - \epsilon) G(z) \\ \text{and } \epsilon \sim U[0, 1] \tag{3.1}$$

3.2.1.2 Additional Training Improvements

Takeru Miyato *et al.* [80] proposed Spectral Normalization, another prosperous method to improve GAN training's stability. This method can be used in conjunction with WGAN-GP. It consists of normalizing the Discriminator weight matrices by their corresponding spectral norm[3], which helps control the Lipschitz constant. Spectral Normalization helps improve stability and avoid vanishing gradient problems, such as mode collapse.

Martin Heusel *et al.* [81] proposed the Two Time-Scale Update Rule (TTUR). This method consists of using a higher learning rate in the Discriminator than in the Generator. The training is more stable and converges better towards Nash's equilibrium. The authors empirically demonstrate that this approach outperforms the original WGAN-GP proposal where they used more training steps in the Discriminator, but with the same learning rate in both networks.

[3] *Spectral Norm* [Calculus Wiki Link]

Soumith Chintala, one of the *DCGAN* [82] paper co-authors, made a presentation at Neural Information Processing Systems (NIPS) 2016 titled *"How to Train a GAN?"* [83] summarizing many tips for stable training of GANs. Some of these tips we will use in this work are: (i) normalize inputs between -1 and 1; (ii) sample noise from Gaussian distributions; (iii) build different batches for real and false data; (iv) use the *LeakyReLU* activation function; (v) use the *Adam* optimizer [73]; (vi) use Embedding layers for discrete variables; (vii) use regularization methods such as Dropout [84] and Batch Normalization [85].

3.2.2 GAN Data Generation Improvements

In the original GAN proposal, only the Generator network is used to produce new samples after the training phase. To this end, random noise vectors from the latent space are introduced into the Generator. If the training has produced a perfect Generator, its probability density function p_G should be the same as that of the actual data. Unfortunately, GANs do not converge to the same distribution of the actual data in practice [86]. Thus, some of the generated samples will not look similar to the original training data.

The inconsistency of p_G leads us to consider a different probability density function, the one implied by the Discriminator concerning the Generator p_D. Typically this distribution is closer to the actual data distribution because the Discriminator has learned information that can be useful in many cases to correct the Generator. Nevertheless, generating samples of this new distribution is not as straightforward as when using only the Generator. To address this challenge, Ryan Turner *et al.* [8] propose the Metropolis-Hastings Generative Adversarial Network (MH-GAN). A sampling method based on a Markov Chain Monte Carlo (MCMC) approach. In this work, we will implement this proposal.

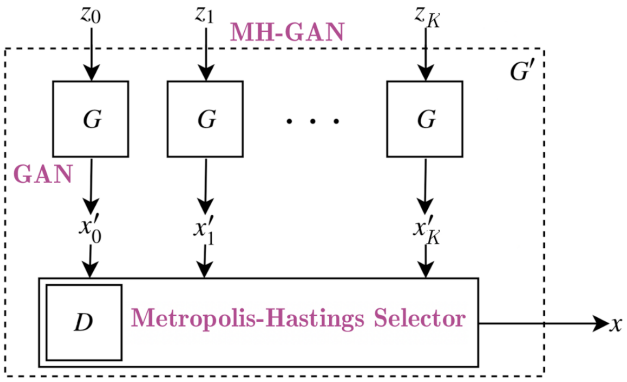


Figure 3.18: Metropolis-Hastings Generative Adversarial Network. Source: [8]

The Metropolis-Hastings approach works as follows. We take K samples from the proposed distribution (i.e., the Generator distribution). Then, we choose one of these samples and sequentially decide whether to accept the next sample or keep the previous one, based on an acceptance rule (Figure 3.18). The decisive feature of MH-GAN is that the acceptance probability can be computed through

the probability densities ratio $\frac{p_D}{p_G}$, and this is available at the Discriminator output (Equation 3.2). The number of samples K is a hyperparameter that can be adjusted based on trade-offs between speed and fidelity.

$$\frac{p_D}{p_G} = \frac{1}{D^{-1}-1} \Longrightarrow \alpha\left(\mathbf{x}', \mathbf{x}_k\right) = \min\left(1, \frac{D(\mathbf{x}_k)^{-1}-1}{D(\mathbf{x}')^{-1}-1}\right) \quad \text{where } D(\mathbf{x}) = \frac{p_D(\mathbf{x})}{p_D(\mathbf{x})+p_G(\mathbf{x})} \tag{3.2}$$

In practice, to avoid long burn-in periods, the authors recommend initializing the sampling process with a randomly chosen real data sample. It is also highly desirable to use a calibration method (e.g., isotonic regression model) for the Discriminator outputs since it is impossible to achieve a perfect Discriminator empirically. Through this simple post-training method, we can significantly improve the fidelity of the generated data samples.

3.2.2.1 Proposed Approach

Having studied the methods for developing our models satisfactorily (*a priori*), we will explain the proposed architecture and data management in depth. In Figure 3.19, we find the workflow that has been applied for the development of all models. It consists of three main phases: (i) Data Wrangling; (ii) Training; (iii) Scenario Generation. Between the second and third phase, there is a feedback loop focused on hyperparameter tuning.

Concerning the first phase of data wrangling, the dataset provided by *TycheTools* has been processed and cleaned for its use in the neural network models. The data has been normalized between values of -1 and 1, as advised by S. Chintala *et al.* [83]. Moreover, a comprehensive Exploratory Data Analysis has been performed, presented in section 3.1.

Figure 3.20 shows the complete GAN architecture, specifying each model's inputs and outputs, and data management during training. The Generator network has three inputs. The sensor ID is introduced in the first one, and an Embedding layer will treat it since it is a categorical variable. In the second one, the historical time-series data of the sensor being studied is introduced, which can be treated using LSTM or CNN layers. Finally, in the last entry, a random noise vector sampled from a multidimensional Gaussian distribution is introduced. The Generator network produces a multidimensional output, trying to generate the next step of the introduced time series correctly. Therefore, if we wish to obtain longer-term predictions, we must reintroduce the previously produced output to the input data and repeat it up to the desired point. The Discriminator network has two inputs. The first one also provides the sensor ID. An Embedding layer of the same dimension as in the Generator will treat this categorical variable. The second input corresponds to the time series data, alternating between real and fake data. Figure 3.21 provides an example of the time-series data fed into the neural models and the ground truth next step.

In the scenario generation phase, we will use two approaches. One of them is to explore the latent space randomly. Since most of the dataset does not contain any anomalies, this random generation will presumably be conservative in most cases and will occasionally produce less usual outputs. The second approach is based on MH-GAN for high-fidelity sampling, as explained in the previous section.

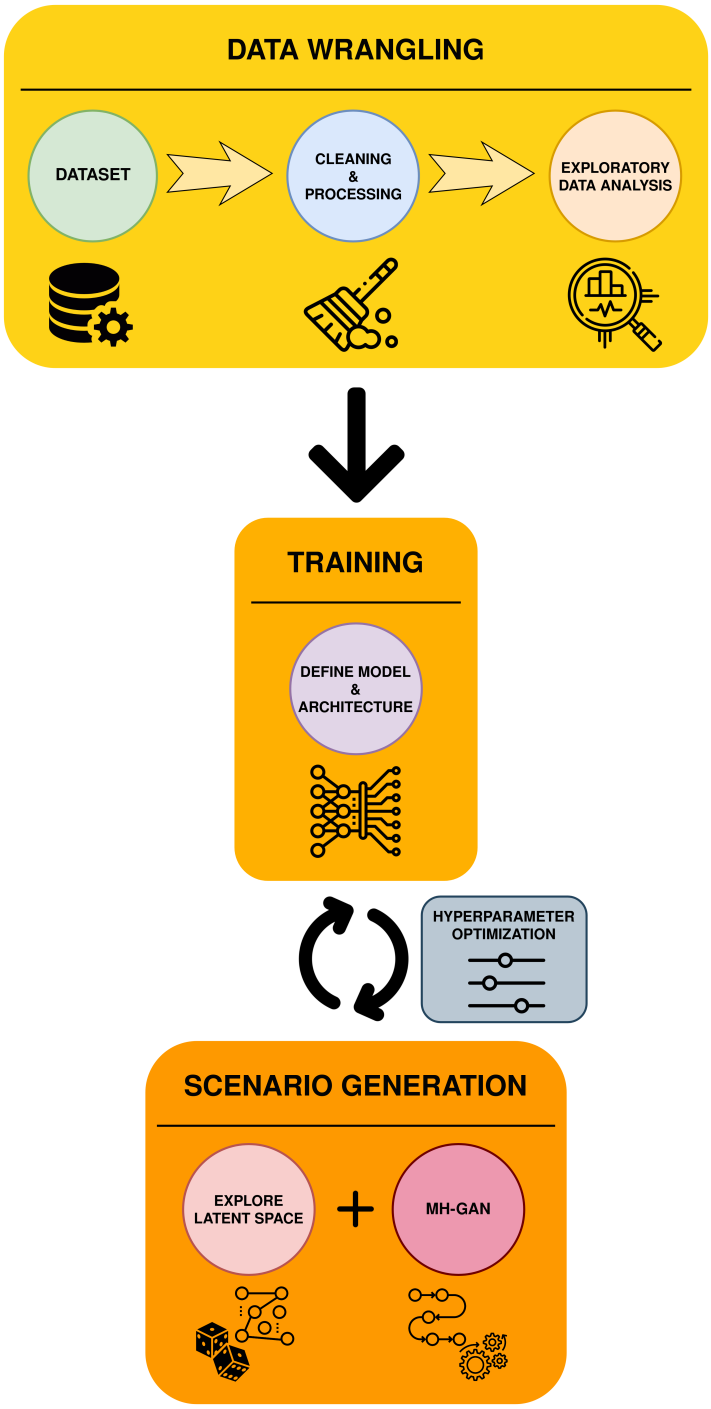


Figure 3.19: Model development workflow.

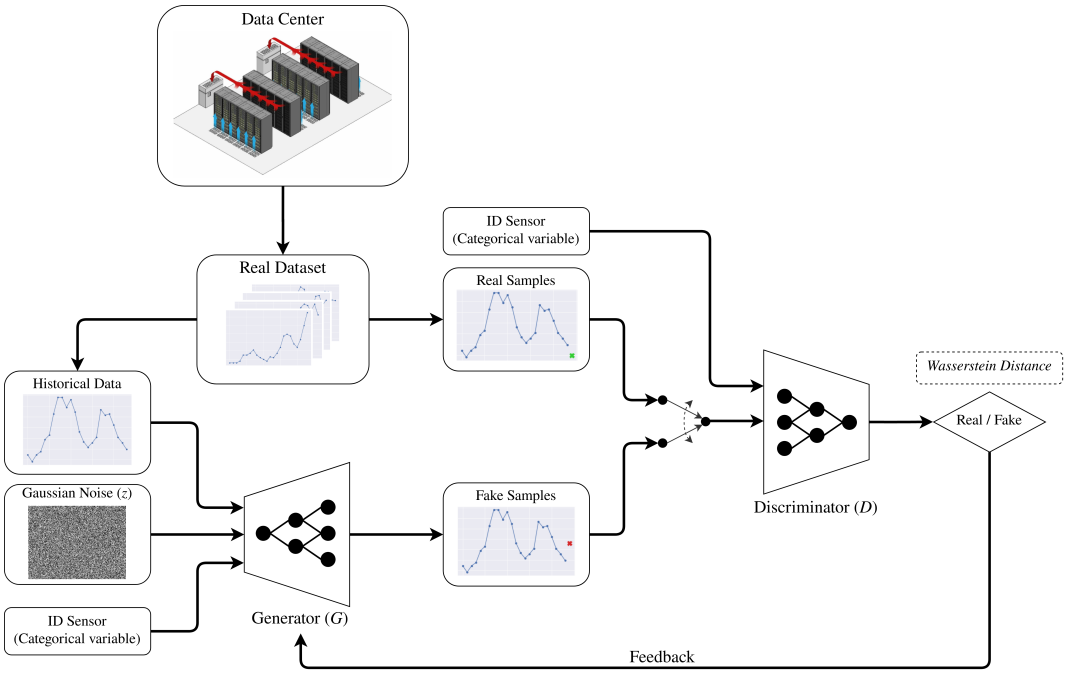


Figure 3.20: Complete GAN architecture

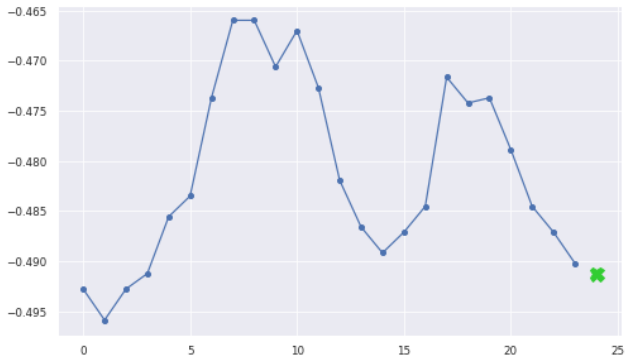


Figure 3.21: Time-series input example. The green cross represents the ground truth next step in the time series.

3.2.2.2 Evaluation Metrics

Validating the quality of the generated scenarios is more challenging than measuring the performance of point forecasts. On the one hand, scenarios must be realistic enough to reflect the structural and inter-variable interdependencies of predicting values at different prediction horizons. On the other hand, they must have enough variability to provide useful additional information. The most commonly used approach in state of the art is to compare the autocorrelation of the generated data with that of the real data and perform a visual examination to ensure that the scenarios "look realistic". This MSc Thesis proposes a novel approach in which we use three different metrics: Kullback-Leibler Divergence, Pinball Loss Function, and Mean Squared Error.

The **Kullback-Leibler (KL) Divergence**, also known as relative entropy, measures how two probability distributions differ (Equation 3.3). It is a distribution-wise asymmetric metric with several applications in statistics, machine learning, neuroscience, and fluid mechanics. In a simplified case, a KL divergence of 0 indicates that the two distributions are identical. In the context of coding theory, this metric can express the extra amount of bits needed to code samples of a distribution P using a code optimized for the compared distribution Q. In the Bayesian inference field, this divergence metric can express a measure of the information gained by revising one's beliefs from the prior probability distribution Q to the posterior probability distribution P. In the proposed use case, we want to verify that the probability distribution of the generated multi-variable scenarios is consistent with the probability distribution of the real data.

$$D_{\mathrm{KL}}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right) \tag{3.3}$$

In our use case, we want to verify that the probability distribution of the generated multi-variable scenarios is consistent with the real data's probability distribution. For this purpose, at the end of each epoch during the training, we will compute a multi-dimensional histogram of a randomly selected validation set and compare it with the forecasted scenarios' multi-dimensional distribution.

The **Pinball Loss Function** L_τ is an error metric for a quantile prediction (Equation 3.4). Unlike other measures, it assigns scores according to the differences between a time series's real values and the quantiles obtained from the forecast. The lower the score, the better the forecast. We will apply it to measure how similar the generated and actual time series are in our use case. Let τ be the target quantile, y the real value, and z the quantile forecast. Figure 3.22 shows an illustration of the Pinball Loss Function L_τ.

$$\begin{aligned} L_\tau(y,z) &= (y-z)\tau && \text{if } y \geq z \\ &= (z-y)(1-\tau) && \text{if } z > y \end{aligned} \tag{3.4}$$

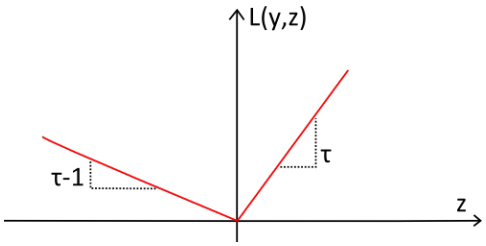


Figure 3.22: Pinball Loss Function illustration.

The Mean Squared Error (MSE) of an estimator measures the squares' average of the committed errors (Equation 3.5). Unlike the mean of the errors in absolute values, this metric gives greater weight to large errors.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{3.5}$$

At the time of conducting the experiments, we will perform a validation of the model at the end of each epoch, using the explained metrics on a randomly selected validation set. If any of the metrics achieve an improvement over the last best score, the model weights are saved for further analysis.

Chapter 4

Experiments and Results

This section describes the performed experiments and discusses the quality of the obtained results. First, we will explain the searching process to obtain the best hyperparameters for the proposed GAN architecture. Then, in section 4.1 we will find some examples of data generated using the random latent space exploration method. In section 4.2 we will use the high-fidelity sampling method MH-GAN and compare the results with those of section 4.1 Finally, in section 4.3 we will present the approach to generate on-demand anomalous situations and present some examples. Table 4.1 shows the primary software tools employed in the project.

The decision to develop the project using the *Google Colab* platform is primarily due to its free Graphics Processing Unit (GPU) executions. Besides, they encourage the projects' reproducibility and compliance with open science' best practices by offering a complete virtual programming environment. The data used and the code developed in this work are openly available on GitHub [94].

As we have explained in the previous sections, the GAN training is a complex and delicate process. Training stability depends on many hyperparameters that must be set by hand and have complex inter-relationships between them. In image generation, the research community has made a great effort to find the hyperparameters that produce better results and more stable training. Unfortunately, the use of GANs for time-series data has been scarcely explored, and no in-depth studies have explored the best hyperparameters for this field. Therefore in this work, we have employed many recommendations from the GANs' state of the

Programming Language	*Python* 3.6
IDE	*Google Colab* [87]
Deep Learning Framework	*Tensorflow* [88]
Python Libraries for Data Processing	*Pandas* [89], *Scikit-Learn* [90], and *Numpy* [91]
Python Libraries for Data Visualization	*Matplotlib* [92], and *Seaborn* [93]

Table 4.1: Software tools employed in the experiments

Hyperparameters	Values
Feature Scaling	Min-Max Scaler [-1, 1]
Training Loss Function	Wasserstein Loss with Gradient Penalty
Batch Normalization in Generator network	✓
Spectral Normalization	✓
Gaussian Noise Dimension	8
Embedding Layer Dimension	8
Networks Size Ratio (Discriminator / Generator)	~4.5
Batch Size	64

Table 4.2: Initial GAN hyperparameters

Hyperparameters					**Results Metrics**				
Optimizer	**Skip Connection**	**Output Activation**	**TTUR**	**Dropout**	**KL Divergence** [bits]	**Pinball Loss Function**		**MSE**	
						Temp.	**Humid.**	**Temp.** [$\check{z}C^2$]	**Humid.** [$\%^2$]
Adam	✗	*linear*	✗	✗	1.888	0.971	0.506	1.943	1.011
Adam	✗	*linear*	✗	✓	1.801	1.179	0.546	2.359	1.092
Adam	✗	*linear*	✓	✗	1.771	1.103	0.389	2.206	0.778
Adam	✗	*linear*	✓	✓	1.825	1.234	0.422	2.468	0.844
Adam	✗	*tanh*	✓	✗	2.431	1.102	0.493	2.203	0.987
Adam	✓	*linear*	✓	✗	2.358	1.122	0.361	2.244	0.721
AdaBelief	✗	*linear*	✗	✗	2.375	1.567	0.542	3.134	1.083
AdaBelief	✗	*linear*	✗	✓	2.013	0.736	0.331	1.473	0.662
AdaBelief	✗	*linear*	✓	✗	1.707	0.68	0.291	1.359	0.583
AdaBelief	✗	*linear*	✓	✓	**1.432**	**0.488**	**0.219**	**0.977**	**0.438**
AdaBelief	✗	*tanh*	✓	✓	1.872	0.656	0.209	1.317	0.419
AdaBelief	✓	*linear*	✓	✓	2.018	0.778	0.593	1.556	1.187

Table 4.3: Hyperparameter tuning process. Experiments result in validation data.

art, combined with an empirical heuristic search to set some reasonably good initial hyperparameters reflected in Table 4.2.

After the first hyperparameter selection process, and based on state-of-the art-examples and some empirical tests, we decided to set the use of Long Short-Term Memory (LSTM) neurons in the Generator and 1D Convolution neurons in the Discriminator. In Figures 4.1 and 4.2, we find the architectures of both networks. The number of trainable parameters in the Discriminator network is approximately 735,000, and in the Generator network, around 165,000.

Once the architectures are fixed, we adjust the following hyperparameters: (i) Optimizer: Adam [73] or AdaBelief [95]; (ii) Skip-Connection: A change in the Discriminator architecture that connects the last step of the time series with the Fully Connected block; (iii) Activation function at the generator output: *Linear* or *tanh*; (iv) TTUR [81]; (v) Dropout [84]. Table 4.3 illustrates the results of this hyperparameter adjustment. The metrics shown are made by comparing the scenario generation of 24 time-steps ahead from a ground truth randomly selected validation set.

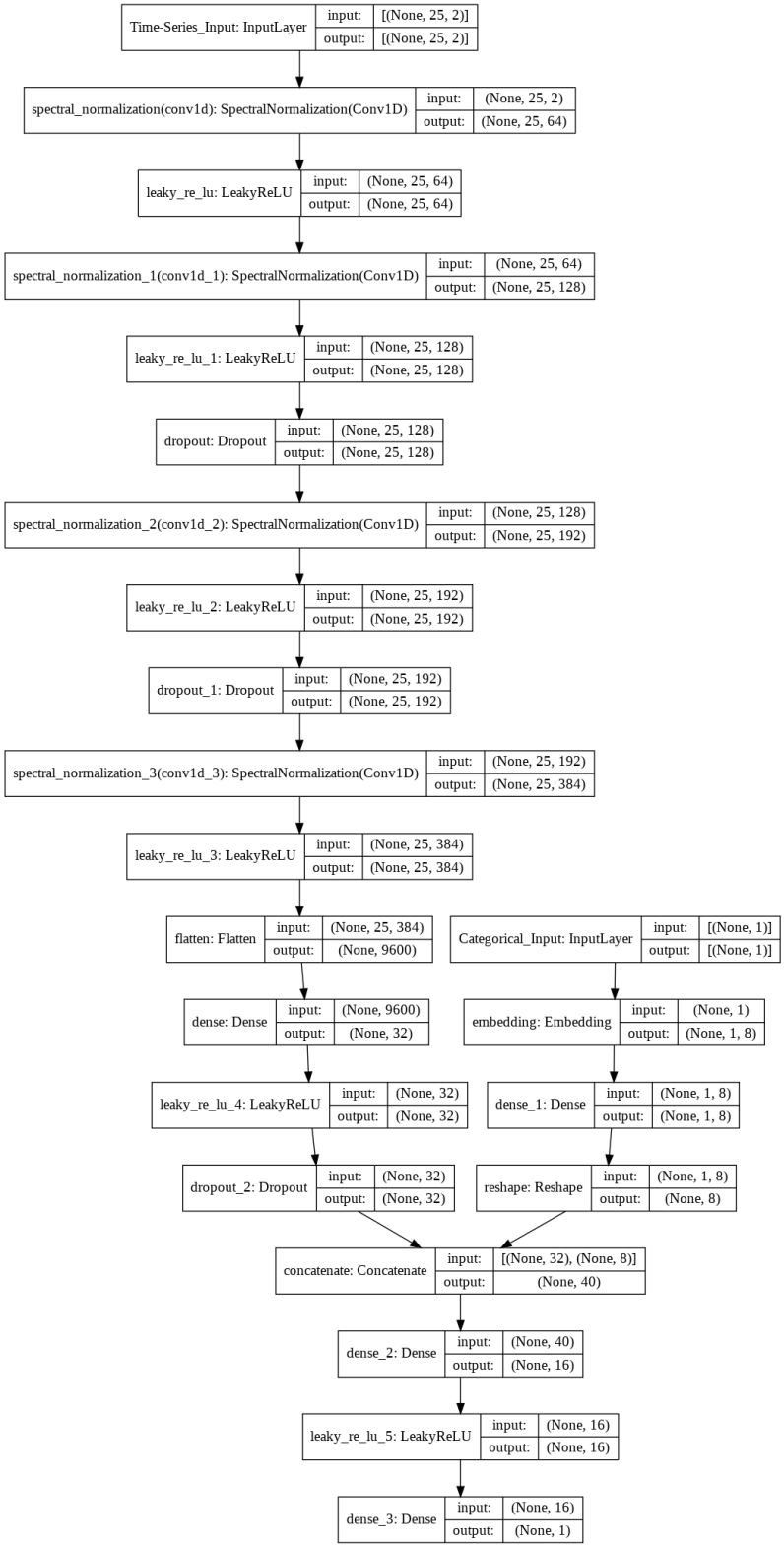


Figure 4.1: Discriminator network architecture. The question marks on the shapes indicate the batch size, which is undefined in the network architecture.

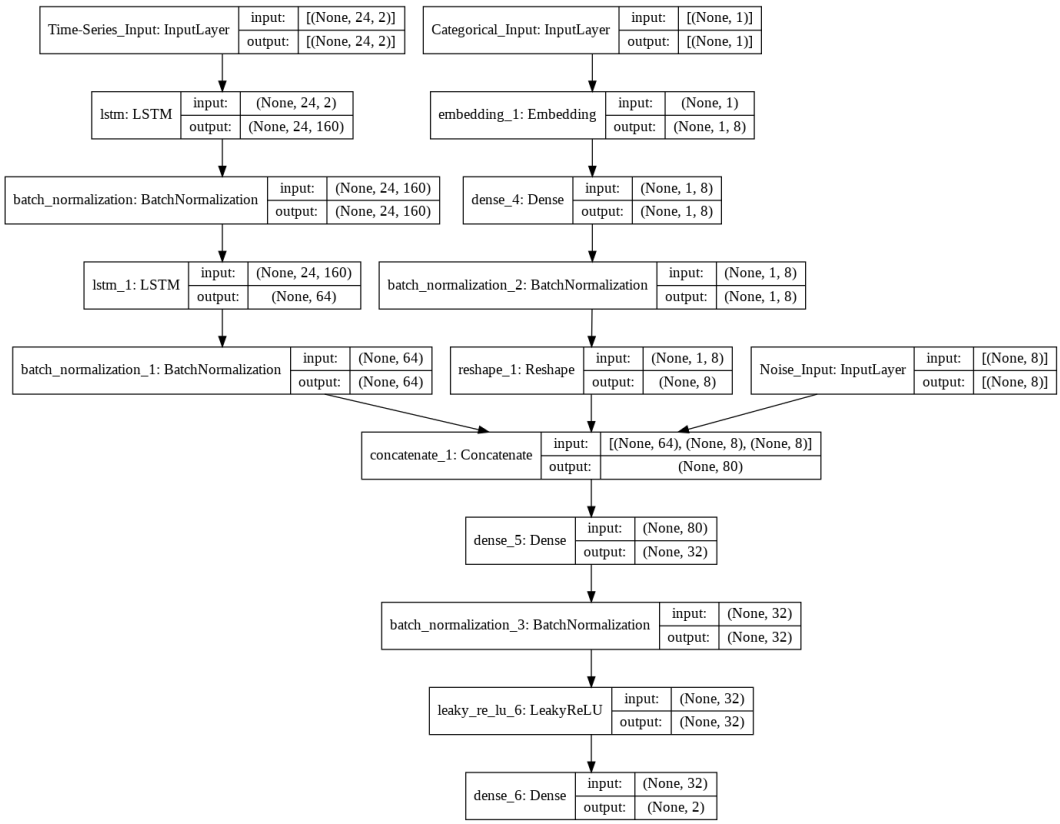


Figure 4.2: Generator network architecture. The question marks on the shapes indicate the batch size, which is undefined in the network architecture.

As shown by the experiment results reflected in Table 4.3, the hyperparameters that offer the best metrics are: (i) AdaBelief Optimizer [95]; (ii) Do not use the Skip-Connection architecture; (iii) *Linear* output activation function in the Generator; (iv) Use the TTUR training technique [81]; and (v) Use the Dropout regularization method [84]. Figure 4.3 illustrates the comparison of the distributions of the scenario predictions of 24 time-steps (4 hours) ahead made by the Generator and a ground truth randomly selected validation set. Hence, the generated distributions from the predictions are consistent with the available real data. In figures 4.4 and 4.5, we observe the comparison between the distributions of the real validation data and the generated data 24 time-steps (4 hours) ahead. Therefore, we confirm the hypothesis that we have managed to generate data that simulate real data characteristics, with a KL divergence of 1.432 bits and an Root Mean Squared Error (RMSE) accuracy error of 0.988ºC for temperature and 0.661% for humidity. In the following sections, we will explore the Generator network's latent space and find examples of realistic scenarios and on-demand anomalous situations.

Figure 4.3: Comparison between Real and Generated data distributions. The Generated data consist of random scenarios of 24 time-steps duration from a validation set of data.

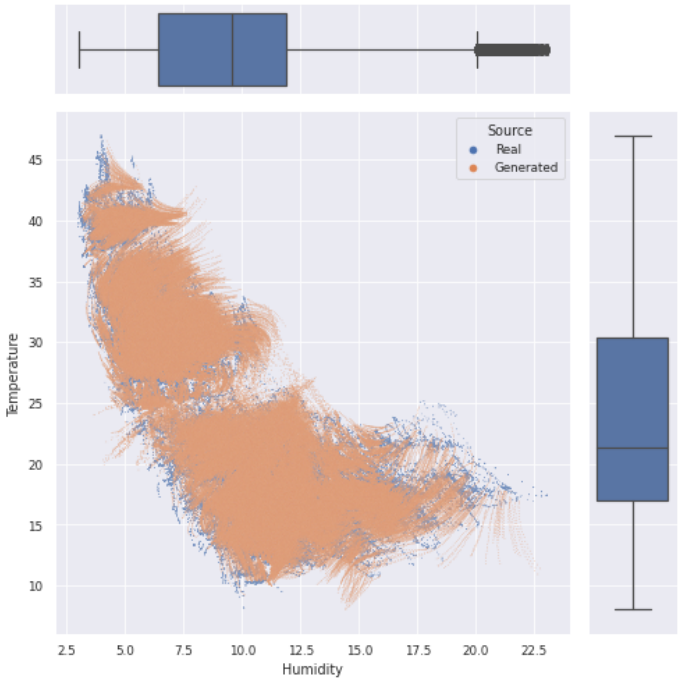


Figure 4.4: Plot confronting Relative Humidity and Temperature data, from Real and Generated distributions. The Generated data consist of random scenarios of 24 time-steps duration from a validation set of data.

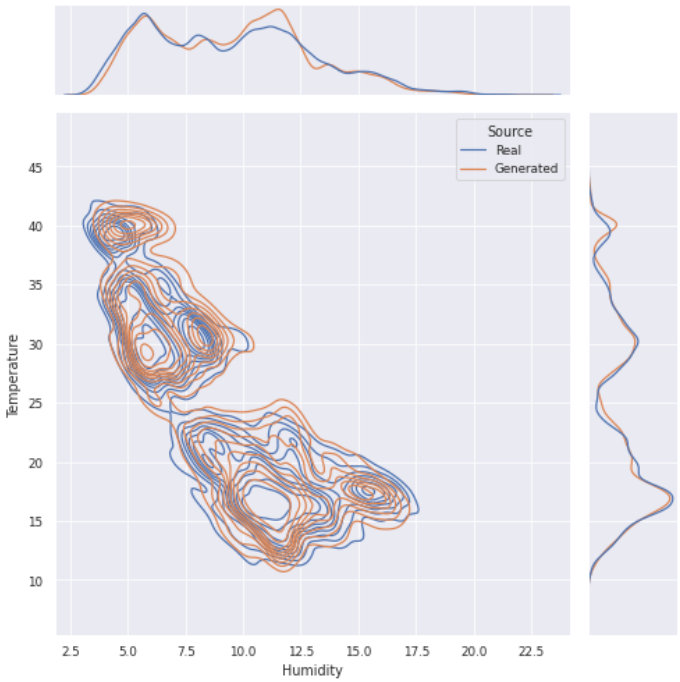


Figure 4.5: Kernel Density Estimation plot confronting Real and Generated data distributions. The Generated data consist of random scenarios of 24 time-steps duration from a validation set of data.

4.1 Scenario Generation: Random Exploration of the Latent Space

In this section, we will explore scenario generation through the method of random exploration of latent space. Since this method is the most straightforward to produce GAN outputs, during the training phase we have used it to obtain the result metrics, obtaining a KL divergence of 1.432 bits in the validation set, and an RMSE accuracy error of 0.988ºC for temperature and 0.661% for humidity.

At each GAN training step, we have been introducing noise vectors sampled from a multidimensional Gaussian space, known as latent space. This implies that when creating a new scenario, the Generator's output will change depending on the noise vector introduced at the input. Therefore, by randomly sampling vectors from latent space, and if the GAN has trained correctly, we get different realistic scenarios for the same given situation. If our Generator were perfect, all possible scenarios produced by the real data distribution would be encoded by the Generator itself in the latent space. However, in practice, the training's imperfections cause points in this latent space that do not correspond to any real situation. Therefore, a certain amount of human supervision is still needed afterward in the generation.

To demonstrate the consistency of the proposed architecture, we have carried out experiments in a test set, where we have obtained a KL divergence of 1.133 bits and an RMSE accuracy error of 1.033ºC for temperature and 0.673% for humidity. In the following, we find some examples of scenarios generated in some sensors, which helps us to visualize the quality of the obtained results. Specifically, six different scenarios have been generated in each example, with a duration of 24 time-steps concatenated predictions (which represents a prediction of 4 hours ahead). In Figure 4.6, we present a situation where the scenarios exhibit low uncertainty, indicating that the Generator has a higher degree of confidence in the results. In Figures 4.7 and 4.8, we show situations where the uncertainty is relatively low in early prediction steps, but it increases as we advance in the number of steps ahead. Figure 4.9 shows a situation where the predicted scenarios' uncertainty is considerably high even from the first steps. The uncertainty captured in the generation of random scenarios gives us useful information about the algorithm's confidence when making a prediction. In all the generated scenarios, we can observe how the correlation between temperature and humidity variables remains consistent. Furthermore, all the generated scenarios, despite those with high uncertainty, can be considered realistic and therefore useful to augment the dataset synthetically.

Figure 4.6: Scenario Generation example of 24 time-steps (4 hours) ahead using the random exploration of the latent space method. Low uncertainty. The scenarios were generated by concatenating each prediction at the input of the Generator.

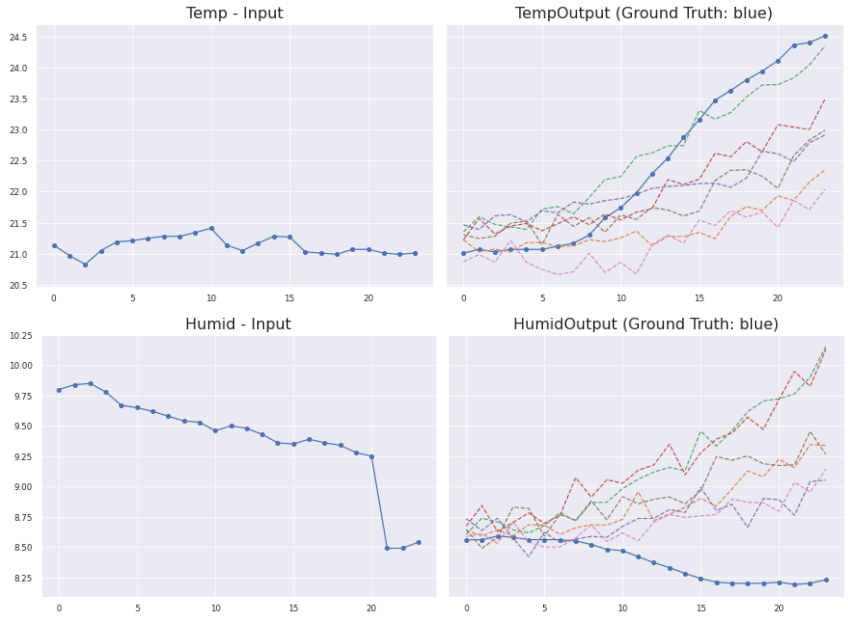


Figure 4.7: Scenario Generation example of 24 time-steps (4 hours) ahead using the random exploration of the latent space method. Increasing uncertainty. The scenarios were generated by concatenating each prediction at the input of the Generator.

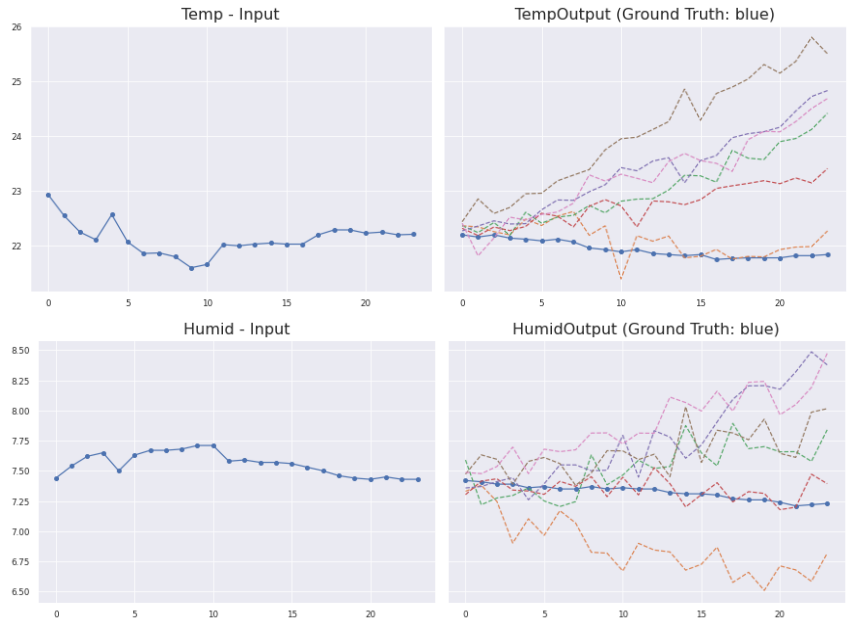


Figure 4.8: Scenario Generation example of 24 time-steps (4 hours) ahead using the random exploration of the latent space method. Increasing uncertainty. The scenarios were generated by concatenating each prediction at the input of the Generator.

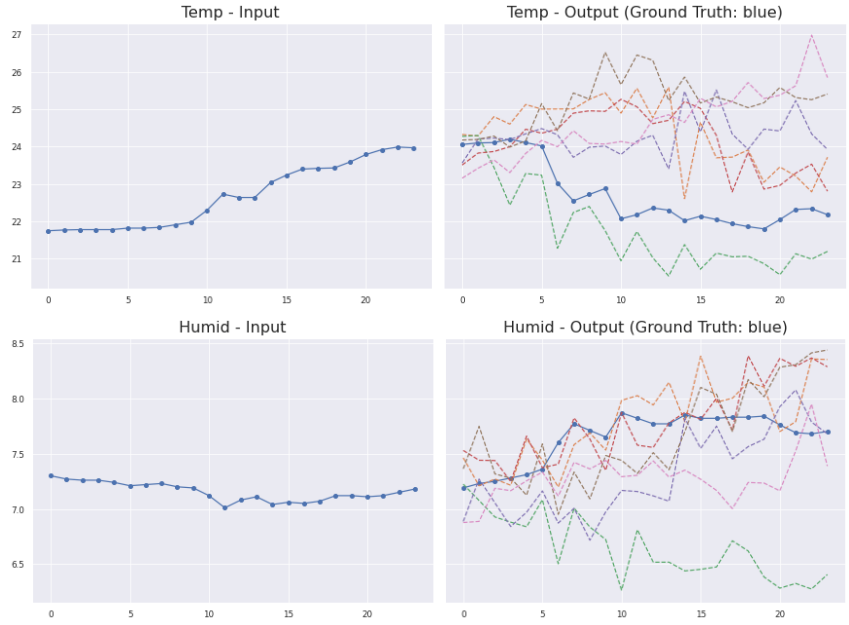


Figure 4.9: Scenario Generation example of 24 time-steps (4 hours) ahead using the random exploration of the latent space method. High uncertainty. The scenarios were generated by concatenating each prediction at the input of the Generator.

To provide an overview of the scenarios generated in all sensors, in Figures 4.10 and 4.11, we find heat maps of the temperature and humidity, where each cell represents a different sensor. Specifically, Figure 4.10 shows the heat maps for an example scenario where one prediction step has been made, comparing the real data with the fake data generated. Figure 4.11 shows the same comparison but after 24 time-step predictions (4 hours ahead). The differences between sensors are due to the distribution of temperatures and humidity in a DC is not homogeneous since there are regions with greater intensity of cooling (inlets) and other zones where the heat produced by many servers accumulates (outlets). Figures 4.10 and 4.11 show how this heterogeneity in the data is consistently maintained by the generated fake data, thus demonstrating that our approach allows the generation of realistic data coherent with the particularities of the scenario under study.

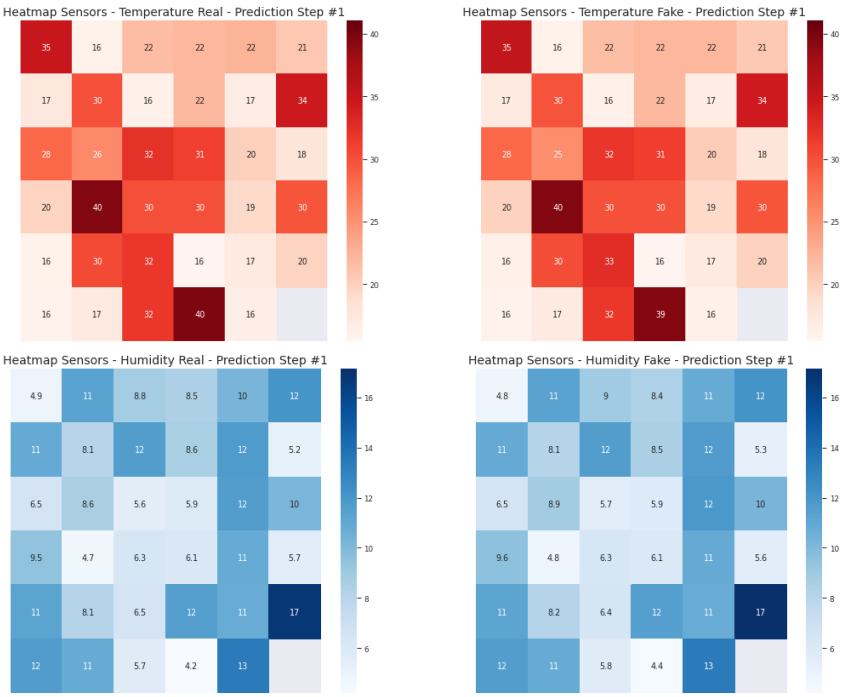


Figure 4.10: Heat maps of temperatures and humidities in the 35 available sensors, one prediction step ahead (10 minutes). Each cell represents a different sensor. The figures on the left show the real data, and the ones on the right show the generated fake data using the random exploration of the latent space method.

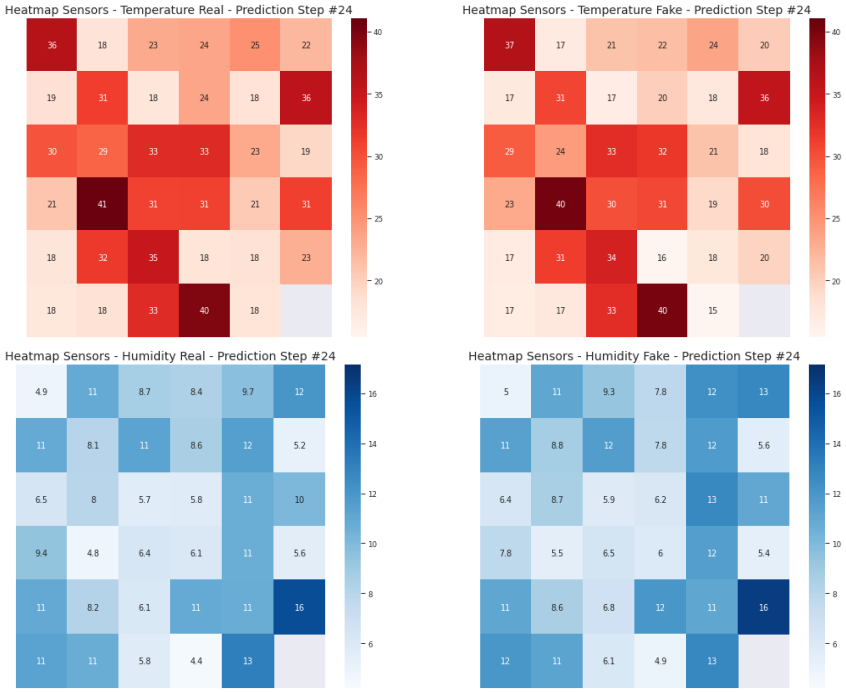


Figure 4.11: Heat maps of temperatures and humidities in the 35 available sensors, 24 prediction steps ahead (4 hours). Each cell represents a different sensor. The figures on the left show the real data, and the ones on the right show the generated fake data using the random exploration of the latent space method.

4.2 Scenario Generation: MH-GAN

In this section, we will explore the generation of scenarios through the alternative sampling method Metropolis-Hastings Generative Adversarial Network (MH-GAN). In section 3.2.2 we have explained how the MH-GAN works, a proposal to improve latent space sampling through a Markov Chain Monte Carlo (MCMC) approach. According to the authors, the results quality and computation time is a trade-off that depends on the number of samples K evaluated at each step. Based on some experiments we have performed, we decided to fix K to 20 samples to avoid extending too much execution times.

In the following, we find some examples of scenarios generated in some sensors, which helps us visualize the results' quality. Specifically, six different scenarios have been generated in each example, with a duration of 24 time-steps concatenated predictions (which represents a prediction of 4 hours ahead). As in the previous section, we obtain examples with low (Figure 4.12), increasing (Figure 4.13 and 4.14), and high uncertainty (Figure 4.15) in the predictions.

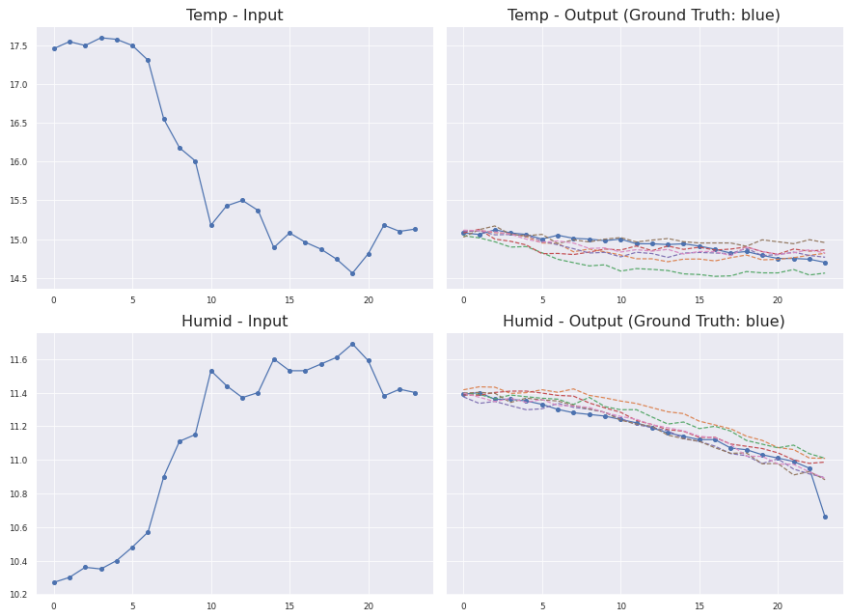


Figure 4.12: Scenario Generation example of 24 time-steps (4 hours) ahead using the MH-GAN method. Low uncertainty. The scenarios were generated by concatenating each different prediction at the input of the Generator.

Figure 4.13: Scenario Generation example of 24 time-steps (4 hours) ahead using the MH-GAN method. Low uncertainty. The scenarios were generated by concatenating each different prediction at the input of the Generator.

Figure 4.14: Scenario Generation example of 24 time-steps (4 hours) ahead using the MH-GAN method. Increasing uncertainty. The scenarios were generated by concatenating each prediction at the input of the Generator.

Figure 4.15: Scenario Generation example of 24 time-steps (4 hours) ahead using the MH-GAN method. High uncertainty. The scenarios were generated by concatenating each different prediction at the input of the Generator.

To provide an overview of the scenarios generated in all sensors, in Figures 4.17 and 4.23, we find heat maps of temperature and humidity, where each cell represents a different sensor. Figure 4.17 shows the heat maps for an example scenario where one prediction step has been made, comparing the real data with the generated data. Figure 4.23 shows the same comparison, yet after 24 predicted time steps (4 hours later). In these figures, we can recognize how the heterogeneity of the data is consistently maintained, thus demonstrating that the MH-GAN approach also allows the generation of realistic data that captures the peculiarities of the scenario under study.

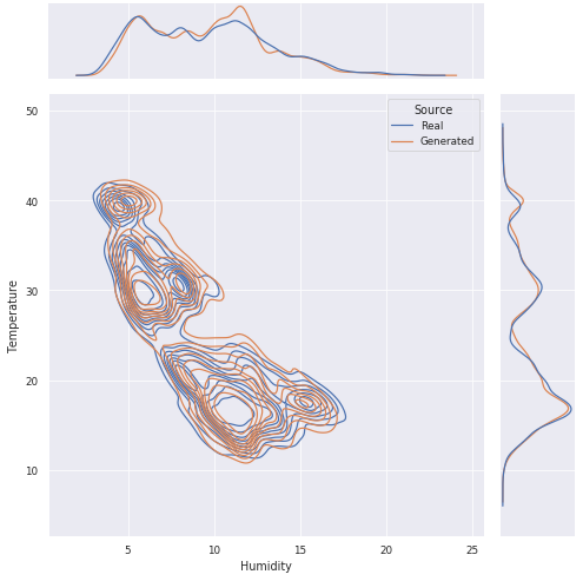


Figure 4.16: Kernel Density Estimation plot confronting Real and Generated data distributions. The Generated data consist of random scenarios of 24 time-steps duration from a test set of data.

To compare this approach's performance compared to the latent space's random exploration, we have performed experiments in the test set, obtaining a KL divergence of 1.059 bits and an RMSE accuracy error of 1.001ºC for temperature and 0.661% for humidity. Figure 4.16 compares the test set data and the generated data (24 prediction time steps), computing the Kernel Density Estimation. At first sight, it seems that this method does indeed offer higher fidelity than the case of random sampling. Unfortunately, MH-GAN has a significant drawback that cannot be avoided. It requires a much more considerable amount of computational resources than in the case of random exploration. The time required to obtain results in the test set has increased from a few seconds in random exploration to more than one day of execution time when using MH-GAN. This significant increase in run time is due to the fact that we need to access (K) 20 times to memory for each result, which is very slow, and check the outcome produced by the Discriminator network. A more in-depth analysis of the MH-GAN method's possible advantages will be necessary in our proposal. Nevertheless, we do not consider that the use of MH-GAN in our architecture is necessary with the performed tests.

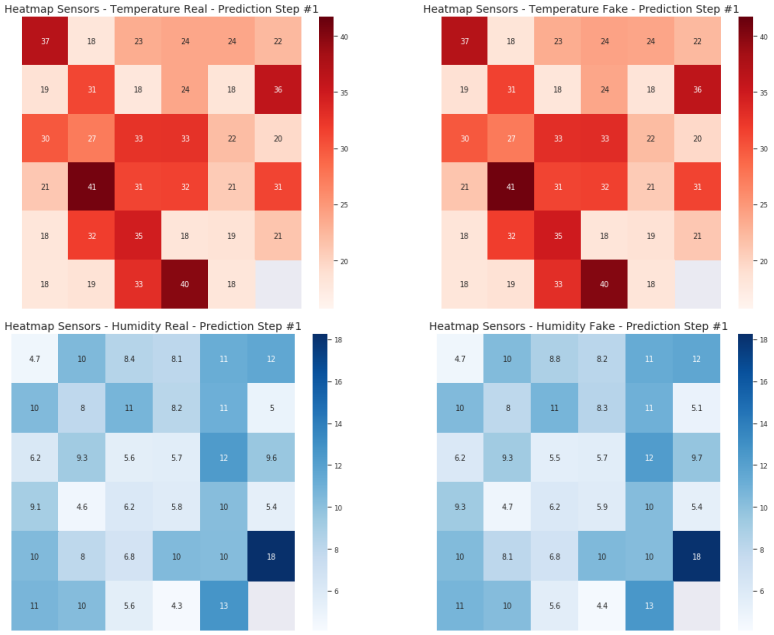


Figure 4.17: Heat maps of temperatures and humidities in the 35 available sensors, one prediction step ahead (10 minutes). Each cell represents a different sensor. The figures on the left show the real data, and the ones on the right show the generated fake data using the MH-GAN method.

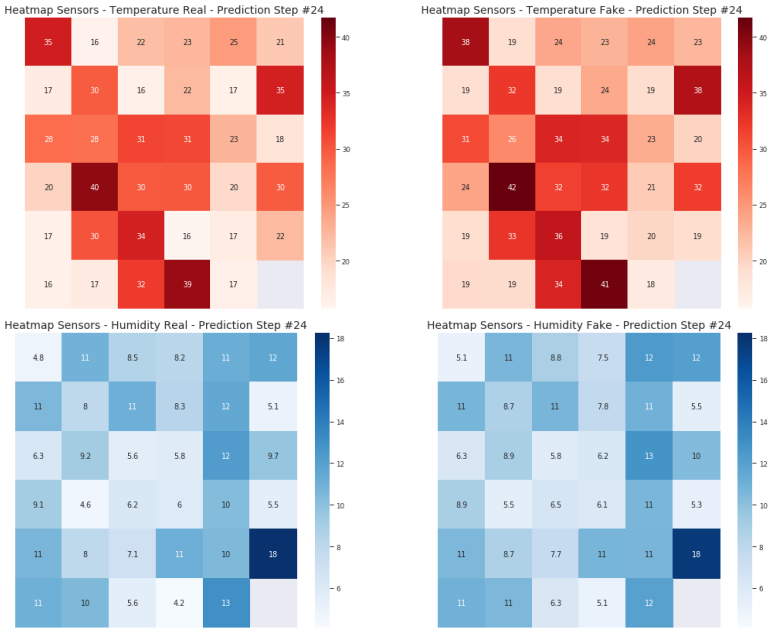


Figure 4.18: Heat maps of temperatures and humidities in the 35 available sensors, 24 prediction steps ahead (4 hours). Each cell represents a different sensor. The figures on the left show the real data, and the ones on the right show the generated fake data using the MH-GAN method.

4.3 On-Demand Anomalus Scenario Generation

In this section, we propose an innovative approach for the generation of anomalous scenarios on demand. We propose an elegant solution that does not require additional efforts or rigid limitations imposed manually. Consequently, the coherence with the real data is maintained, and no expert knowledge is required to establish the constraints.

As previously explained, we have introduced randomly sampled vectors from a multidimensional Gaussian space during each GAN training step. Gaussian distributions are mathematically parameterized by their mean and standard deviation. We have employed a distribution with 0 mean and standard deviation of 1 to sample during the training process. Our proposal consists of increasing the standard deviation of the Gaussian noise distribution when the anomaly is desired, obtaining results that are distant from the average. In Figures 53 and 54, we have increased the standard deviation from 1 to 9 in the 10th prediction step. We can appreciate that many scenarios produced the desired anomaly and that the following steps of the forecasting are consistent with this event.

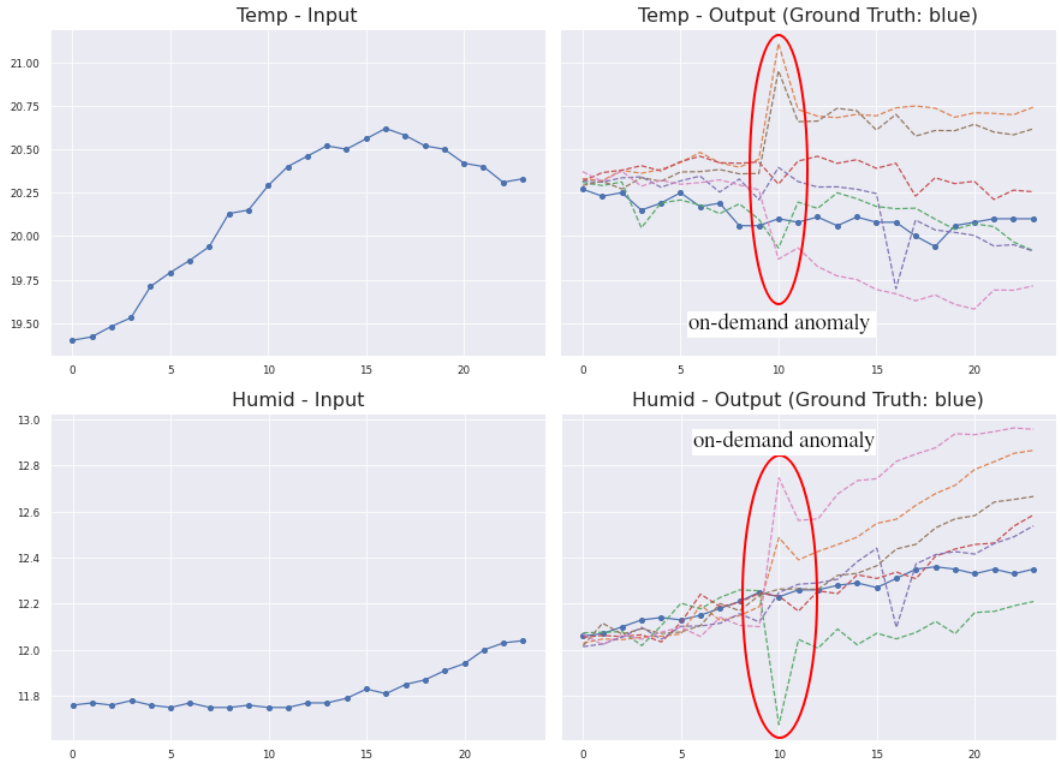


Figure 4.19: On-demand anomaly generation scenario example. The generated scenario has a duration of 24 time-steps (4 hours), and the desired anomaly was introduced in the 10th step.

Figure 4.20: On-demand anomaly generation scenario example. The generated scenario has a duration of 24 time-steps (4 hours), and the desired anomaly was introduced in the 10th step.

To provide a general overview of introducing an anomaly in a particular sensor, in Figures 4.21, 4.22, and 4.23 we find heat maps of temperature and humidity where each cell represents a different sensor. Figure 4.21 shows the heat maps of a scenario in the 10th step, before introducing any anomaly. Figure 4.22 shows the heat map in the time instant where an anomaly was introduced into the sensor 18. We note that there is a trend change in that sensor both in the temperature and humidity. In Figure 4.23, we observe the heat map 13 steps after introducing the anomaly in sensor 18, where we can notice that the generated scenario is consistent with the anomaly event that we have introduced artificially, and the correlation between temperature and humidity variables is preserved (temperature increases while humidity decreases).

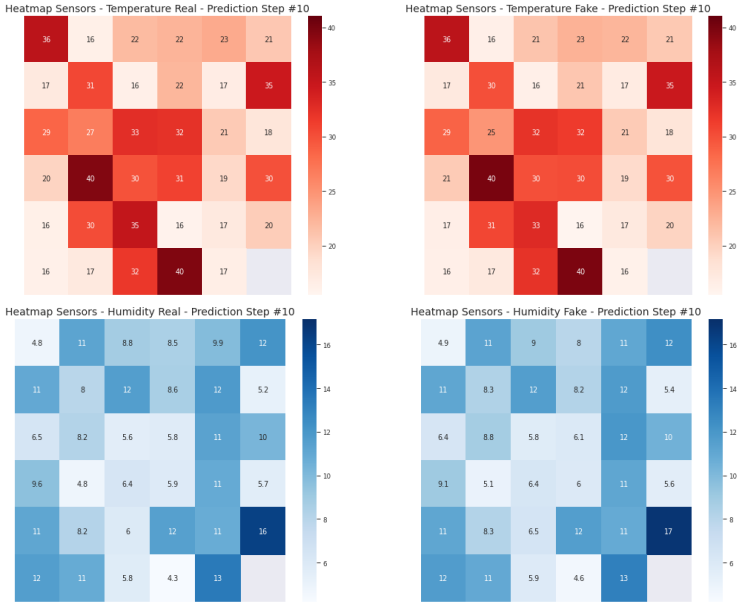


Figure 4.21: Heat maps of temperatures and humidities in the 35 available sensors, 10 prediction steps ahead 1 hour and 40 minutes). Each cell represents a different sensor. The figures on the left show the real data, and the ones on the right show the generated fake data.

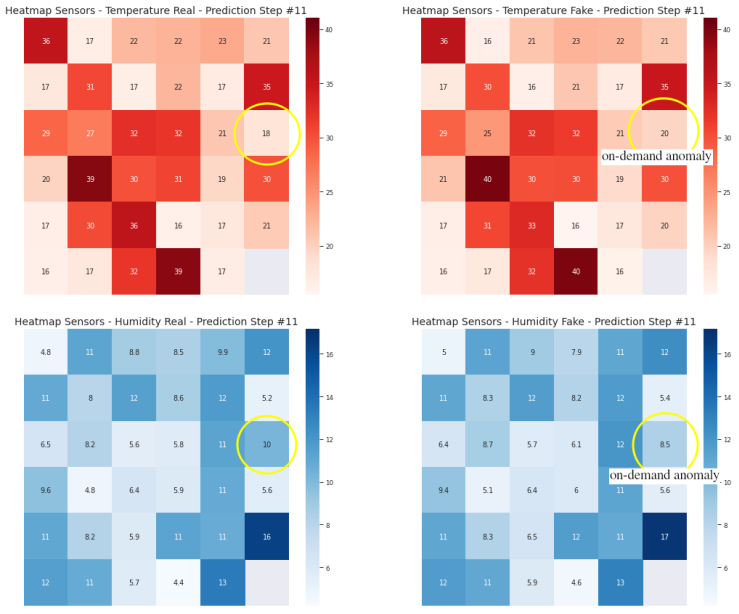


Figure 4.22: Heat maps of temperatures and humidities in the 35 available sensors, 11 prediction steps ahead (1 hour and 50 minutes). Each cell represents a different sensor. The figures on the left show the real data, and the ones on the right show the generated fake data. The cell marked with the yellow circle indicates the sensor where the anomaly has been introduced.

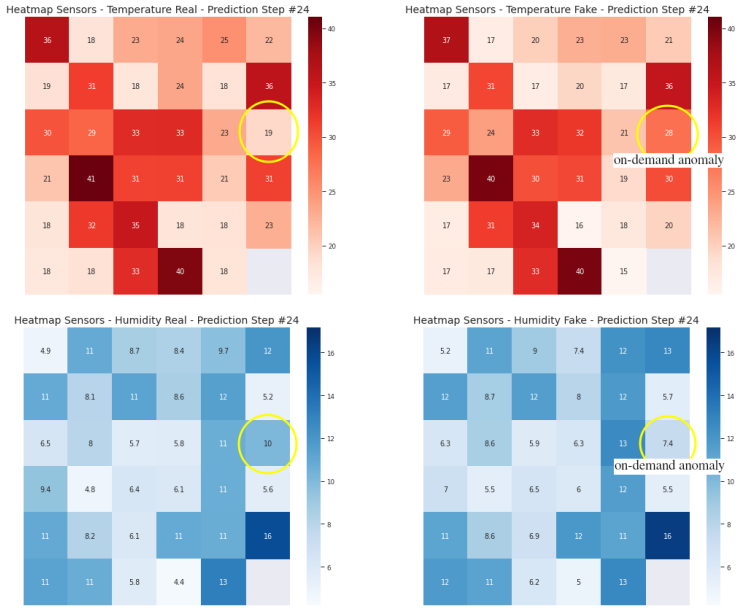


Figure 4.23: Heat maps of temperatures and humidities in the 35 available sensors, 24 prediction steps ahead (4 hours). Each cell represents a different sensor. The figures on the left show the real data, and the ones on the right show the generated fake data. The cell marked with the yellow circle indicates the sensor where the anomaly has been introduced.

Chapter 5

Reflections on Research

This chapter covers reflections on the project and future work that originates from it. This MSc Thesis expands the knowledge and applications of a domain scarcely researched by the scientific community, synthetic time-series scenario generation using GANs.

5.1 Conclusions

Chapter 1 introduced the motivation for developing this project. Our society and economy's unprecedented digital transformation will trigger an abnormal amount of data traffic and computing resources. The current Cloud Computing paradigm urgently needs improved methods of optimization and operation. Since data-driven optimization methods have proven to be the most accurate for optimizing complex systems, greater public datasets and modeling capabilities are indispensable for efficiently understanding and operating Data Centers.

Chapter 2 discusses state-of-the-art solutions on the synthetic generation of time-series data, which is the nature of data produced in a DC facility. A large number of statistical methods are available for this purpose. However, they require significant expert knowledge and produce poor results when nonlinear relationships between features are dominant. We also studied proposals for synthetic sequential data generation based on GANs, where we found a research domain that is beginning to make tangible progress but is still poorly explored. Furthermore, we review the theory supporting generative models, and especially GANs. Our main contribution to the state of the art is the conditional generation of multi-variable scenarios using GANs. Our approach allows on-demand generation of scenarios and anomalous situations, in particular time instants and specific data sources.

Chapter 3 considers a multi-variable sensor data scenario in a real DC facility, used to verify our work's usefulness. Synthetic data generation is presented as an alternative with a huge potential to boost optimization in DCs, where data gathering with significant variability is expensive, and the equipment's physical integrity may be at risk. A comprehensive exploratory analysis of the available data is performed.

We also describe the methodology applied in the experiments, the evaluation metrics, and the improvements implemented to stabilize the training and results' obtention.

Finally, Chapter 4 describes the experiments carried out, where we perform an intensive hyperparameter tuning to find the architecture that produces better results. Eventually, we obtained satisfactory results that demonstrate the Generator's consistency. In particular, the best result obtained from scenario generation in the random validation set is a KL divergence of the 1.432 bits and an RMSE accuracy error of 0.988ºC for temperature and 0.661% for humidity. Afterward, we present scenario generation and on-demand anomaly examples from a randomly selected test set, demonstrating the proposed architecture's reliability.

The results obtained in this work establish a methodology that extends synthetic time-series data applications, enabling to use categorical variables and the generation of multi-variable scenarios at specific moments in time. The generation of anomalous situations on-demand introduces significant data variability without additional effort or endangering electronic equipment's physical integrity. Moreover, the proposed methodology can be employed in any similar time-series-like problems in other fields of application. GANs are a type of generative algorithm with only five years of life. Therefore, the advances that will be made in the next years in this field can be implemented to improve the obtained results and address its limitations.

This MSc Thesis has successfully achieved the goals set at the beginning of the project in Section 1.4:

- **Provide a comprehensive study of Generative Adversarial Networks (GAN) and their possible applications in DC optimization**: This objective has been satisfactorily addressed in sections 1.2 and 2.2, where we have present the most successful trends in DC optimization and a comprehensive study of GANs and the theory behind them.
- **Extend the state-of-the-art methodology to produce synthetic time-series data using GANs, enabling the use of multi-variable and categorical data:**: Section 3.2 describes the proposed methodology and architecture, which enables using multi-variable data and categorical variables at the GAN input.
- **Generate realistic synthetic data to enable better analytics and models for DC optimization**: In chapter 4, dedicated to the conducted experiments, we have demonstrated the generated data's realism using the two proposed methods: Random Exploration of Latent Space and MH-GAN. Obtaining slightly higher fidelity results through the MH-GAN method, yet at a very high computational cost.
- **Generate on-demand anomalous synthetic situations in computation systems, consistent with available real data**: Section 4.3 shows examples of on-demand anomaly scenario generation, which are coherent with real data. Thus demonstrating that our approach allows the generation of potentially dangerous situations, saving costs both at an economic standpoint and in terms of time consumption.

5.2 Open Issues

Despite the satisfactory results obtained in this work, it is crucial to correctly study the proposal's limitations due to the use of GANs. The GAN field of study is still young and needs to address its main limitations. In the following, we will explain the main barriers derived from the use of GANs that we have found during this work.

Scarce research on the use of GANs for time-series data

GANs have been mostly used for image generation due to the outstanding results they achieve in this field. The scientific community has made an enormous effort to find the hyperparameters that produce the best results and the most stable training. However, the use of GANs in time-series data has been very little researched. This leads to unstable training processes and the need for a very intensive hyperparameter search to obtain satisfactory results.

Generated Data Bias

The use of GANs offers a flexible tune generation that provides exceptional advantages. Nevertheless, the data generated variability is limited and biased by the data used during the training. This is also provoked by the use of the similarity between the generated data and the real data distributions as a measure of realism and usefulness.

Human Supervision is Needed

The Generator's final purpose is to recreate the distribution of real data. However, the training process has limitations as there are points in the latent space that do not correspond to any real situation, and only a limited amount of data is accessible. This implies that some of the samples produced by the Generator are not part of the real situations. Therefore, some human supervision is still necessary for the selection of the best outcomes.

Relatively Large Amounts of Data are Needed

Some approaches to synthetic data need very little real data to obtain satisfactory results. However, GANs require a relatively large amount of data for stable training. Still, this amount is tiny compared to the amount of data needed to achieve state-of-the-art results in large Deep Learning algorithms.

5.3 Future Work

In the last section of this document, some possible new steps for the future of this research are discussed. This MSc Thesis extends the application of GANs for use in time-series data augmentation, but these algorithms have only five years of life. Hence the advances that will be made in the next years in this field can be implemented to improve the obtained results and address its limitations. The possible future lines can be divided as follows:

Hyperparameter Optimization

As previously discussed, GANs have been scarcely explored for applications other than image generation. Therefore, a comprehensive and consistent study of the best hyperparameters required for stable training in time-series data is crucial.

Scalability of Multi-Variable Datasets

For the proposed use case, DC optimization, the use of only three variables (temperature, relative humidity, and the analyzed sensor ID) is sufficient to demonstrate the results' utility. Nevertheless, it is necessary to empirically demonstrate the proposed architecture's scalability with a larger number of variables.

Further Study on the Usefulness of the Results

We have demonstrated the realism of the data generated in this work. Yet further research is needed on the real utility for other Deep Learning algorithms (e.g., Train on Synthetic Data - Test on Real Data).

Disentangled Latent Space

In the literature review section, we have highlighted a fascinating work presented recently by Ji Qiao *et al.* [33]. In this work, the authors use orthogonal regularization techniques in the latent space, making each dimension of this space represent differentiated characteristics in the scenario generation. We strongly believe that this work makes significant progress in explaining the results generated by GANs, making the use of these algorithms more accessible.

Add Supplementary Conditional Information

In Deep Learning, experiments have shown that providing the input with additional useful information improves the results and the created algorithms' reliability. In the field of time series, we could add frequency spectrum information,

predictions with classical methods (e.g., ARIMA), or time information (e.g., time, day of the week, time of the year).

Explore Alternative Evaluation Metrics

In this work, we have proposed three metrics to verify the realism of the data generated, highlighting the KL Divergence particularly. Alternative metrics can be explored that, for example, contemplate the frequency spectrum.

Explore Further GAN Architectures and Improvements

The field of research of the GANs is just beginning to be born. New architectures and improvements are proposed every year. Soon it will be necessary to evaluate the incorporation of state-of-the-art architectures in image generation (e.g., *StyleGAN*) and new training improvements (e.g., Model weight averaging).

Bibliography

[1] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey, "Recalibrating global data center energy-use estimates," Science, vol. 367, pp. 984–986, Feb. 2020. Publisher: American Association for the Advancement of Science Section: Policy Forum.

[2] "Trends in data centre energy consumption under the European Code of Conduct for data centre energy efficiency," text, Team FPFIS, European Commission.

[3] R. Evans and J. Gao, "DeepMind AI Reduces Google Data Centre Cooling Bill by 40%." Deepmind Blog.

[4] N. Jones, "How to stop data centres from gobbling up the world's electricity," Nature, vol. 561, pp. 163–166, Sept. 2018. Publisher: Nature Publishing Group.

[5] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised Image-to-Image Translation Networks," Advances in Neural Information Processing Systems, vol. 30, pp. 700–708, 2017.

[6] Beaulieu-Jones Brett K., Wu Zhiwei Steven, Williams Chris, Lee Ran, Bhavnani Sanjeev P., Byrd James Brian, and Greene Casey S., "Privacy-Preserving Generative Deep Neural Networks Support Clinical Data Sharing," Circulation: Cardiovascular Quality and Outcomes, vol. 12, p. e005122, July 2019. Publisher: American Heart Association.

[7] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled Generative Adversarial Networks," arXiv:1611.02163 [cs, stat], May 2017. arXiv: 1611.02163.

[8] R. Turner, J. Hung, E. Frank, Y. Saatchi, and J. Yosinski, "Metropolis-Hastings generative adversarial networks," in Proceedings of the 36th International Conference on Machine Learning (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of Proceedings of Machine Learning Research, (Long Beach, California, USA), pp. 6345–6353, PMLR, 09–15 Jun 2019.

[9] "Cisco Annual Internet Report (2018–2023) White Paper," tech. rep., Cisco, 2020.

[10] "Digital economy and society statistics - enterprises - Statistics Explained," tech. rep., European Commission, 2020.

[11] "Digital economy and society statistics - households and individuals - Statistics Explained," tech. rep., European Commission, 2020.

[12] "Global Internet Phenomena," tech. rep., Sandvine, 2019.

[13] "Cloud computing - statistics on the use by enterprises - Statistics Explained," tech. rep., European Commission, 2018.

[14] "Annual Data Center Survey Results," tech. rep., Uptime Institute - Intelligence, 2020.

[15] "Cisco Global Cloud Index:Forecast and Methodology, 2016–2021," tech. rep., Cisco, 2018.

[16] N. Lazic, C. Boutilier, T. Lu, E. Wong, B. Roy, M. K. Ryu, and G. Imwalle, "Data center cooling using model-predictive control," Advances in Neural Information Processing Systems, vol. 31, pp. 3814–3823, 2018.

[17] R. Lebaredian, "Synthetic Data will Drive Next Wave of Business Applications | GTC Silicon Valley 2019," 2019.

[18] K. Duemig, "Accelerating time-to-market with fabricated test data," 2017. IBM Big Data & Analytics Hub.

[19] K. Duemig, "Protect your customer data with relevant test data," 2017. IBM Big Data & Analytics Hub.

[20] T. Kohlberger and Y. Liu, "Generating Diverse Synthetic Medical Image Data for Training Machine Learning Models," 2020. Google AI Blog.

[21] "Modernization of Statistical Disclosure Limitation at US Census Bureau," tech. rep., US Census Bureau, 2020. Section: Government.

[22] "Artificial Intelligence: Emerging Opportunities, Challenges, and Implications for Policy and Research," tech. rep., U. S. Government Accountability Office, June 2018.

[23] J. Bughin, E. Hazan, S. Ramaswamy, M. Chui, T. Allas, P. Dahlström, N. Henke, and M. Trench, "Artificial Intelligence, The Next Digital Frontier?," tech. rep., McKinsey Global Institute, 2017.

[24] "State of AI in the Enterprise, 2nd Edition," tech. rep., Deloitte Insights, 2018.

[25] "Simulacrum: Artificial patient-like cancer data to help researchers gain insights," 2019. http://simulacrum.healthdatainsight.org.uk/.

[26] "Generating and Editing High-Resolution Synthetic Images with GANs," 2018. NVIDIA Developer News Center, https://news.developer.nvidia.com/generating-and-editing-high-resolution-synthetic-images-with-gans/.

[27] S. Assefa, D. Dervovic, M. Mahfouz, T. Balch, P. Reddy, and M. Veloso, "Generating Synthetic Data in Finance: Opportunities, Challenges and Pitfalls," June 2020. SSRN.

[28] T. T. Um, F. M. J. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić, "Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks," in Proceedings of the 19th ACM International Conference on Multimodal Interaction, ICMI '17, (New York, NY, USA), p. 216–220, Association for Computing Machinery, 2017.

[29] C. M. Bishop, "Training with noise is equivalent to tikhonov regularization," Neural Computation, vol. 7, no. 1, pp. 108–116, 1995.

[30] K. M. Rashid and J. Louis, "Time-warping: A time series data augmentation of imu data for construction equipment activity identification," in Proceedings of the 36th International Symposium on Automation and Robotics in Construction (ISARC) (M. Al-Hussein, ed.), (Banff, Canada), pp. 651–657, International Association for Automation and Robotics in Construction (IAARC), May 2019.

[31] A. L. Guennec, S. Malinowski, and R. Tavenard, "Data augmentation for time series classification using convolutional neural networks," 2016.

[32] B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," 2020. arXiv preprint arXiv:2007.15951.

[33] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," Neurocomputing, vol. 50, pp. 159–175, Jan. 2003.

[34] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, "ARIMA models to predict next-day electricity prices," IEEE Transactions on Power Systems, vol. 18, pp. 1014–1020, Aug. 2003.

[35] A. J. Conejo, M. A. Plazas, R. Espinola, and A. B. Molina, "Day-ahead electricity price forecasting using the wavelet transform and ARIMA models," IEEE Transactions on Power Systems, vol. 20, pp. 1035–1042, May 2005.

[36] M. Valipour, M. E. Banihabib, and S. M. R. Behbahani, "Comparison of the ARMA, ARIMA, and the autoregressive artificial neural network models in forecasting the monthly inflow of Dez dam reservoir," Journal of Hydrology, vol. 476, pp. 433–441, Jan. 2013.

[37] I. L. MacDonald and W. Zucchini, Hidden Markov and Other Models for Discrete- valued Time Series. CRC Press, Jan. 1997.

[38] W. Zucchini, I. L. MacDonald, and R. Langrock, Hidden Markov Models for Time Series: An Introduction Using R, Second Edition. CRC Press, Dec. 2017.

[39] M. R. Hassan and B. Nath, "Stock market forecasting using hidden Markov model: a new approach," in 5th International Conference on Intelligent Systems Design and Applications (ISDA'05), pp. 192–196, Sept. 2005. ISSN: 2164-7151.

[40] A. M. Gonzalez, A. M. S. Roque, and J. Garcia-Gonzalez, "Modeling and forecasting electricity prices with input/output hidden Markov models," IEEE Transactions on Power Systems, vol. 20, pp. 13–24, Feb. 2005.

[41] A. Gupta and B. Dhingra, "Stock market prediction using Hidden Markov Models," in 2012 Students Conference on Engineering and Systems, pp. 1–4, Mar. 2012.

[42] G. Koop and D. Korobilis, Bayesian Multivariate Time Series Methods for Empirical Macroeconomics. Now Publishers Inc, 2010.

[43] M. F. J. Steel, "Bayesian time series analysis," in Macroeconometrics and Time Series Analysis (S. N. Durlauf and L. E. Blume, eds.), The New Palgrave Economics Collection, pp. 35–45, London: Palgrave Macmillan UK, 2010.

[44] S. L. Scott and H. R. Varian, "Predicting the Present with Bayesian Structural Time Series," SSRN Scholarly Paper ID 2304426, Social Science Research Network, Rochester, NY, June 2013.

[45] D. Barber, A. T. Cemgil, and S. Chiappa, Bayesian Time Series Models. Cambridge University Press, Aug. 2011.

[46] W. Härdle, J. Horowitz, and J.-P. Kreiss, "Bootstrap Methods for Time Series," International Statistical Review, vol. 71, no. 2, pp. 435–459, 2003. Publisher: Wiley, International Statistical Institute (ISI).

[47] P. Bühlmann, "Sieve bootstrap for time series," Bernoulli, vol. 3, pp. 123–148, June 1997. Publisher: Bernoulli Society for Mathematical Statistics and Probability.

[48] J.-P. Kreiss and S. N. Lahiri, "1 - Bootstrap Methods for Time Series," in Handbook of Statistics (T. Subba Rao, S. Subba Rao, and C. R. Rao, eds.), vol. 30 of Time Series Analysis: Methods and Applications, pp. 3–26, Elsevier, Jan. 2012.

[49] D. N. Politis, "The Impact of Bootstrap Methods on Time Series Analysis," Statistical Science, vol. 18, no. 2, pp. 219–230, 2003. Publisher: Institute of Mathematical Statistics.

[50] X. Zhou, Z. Pan, G. Hu, S. Tang, and C. Zhao, "Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets," Apr. 2018. ISSN: 1024-123X, Publisher: Hindawi.

[51] K. Zhang, G. Zhong, J. Dong, S. Wang, and Y. Wang, "Stock Market Prediction Based on Generative Adversarial Network," Procedia Computer Science, vol. 147, pp. 400–406, Jan. 2019.

[52] E. L. Zec, H. Arnelid, and N. Mohammadiha, "Recurrent conditional gans for time series sensor modelling," in Time Series Workshop at International Conference on Machine Learning,(Long Beach, California), 2019.

[53] A. Koochali, P. Schichtel, A. Dengel, and S. Ahmed, "Probabilistic Forecasting of Sensory Data With Generative Adversarial Networks – ForGAN," IEEE Access, vol. 7, pp. 63868–63880, 2019.

[54] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs," arXiv:1706.02633 [cs, stat], Dec. 2017.

[55] J. Lan, Q. Guo, and H. Sun, "Demand Side Data Generating Based on Conditional Generative Adversarial Networks," Energy Procedia, vol. 152, pp. 1188–1193, Oct. 2018.

[56] 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, p. 130, 2019. Publisher: Multidisciplinary Digital Publishing Institute.

[57] C. Zhang, S. R. Kuppannagari, R. Kannan, and V. K. Prasanna, "Generative Adversarial Network for Synthetic Time Series Data Generation in Smart Grids," in 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pp. 1–6, Oct. 2018.

[58] M. Alzantot, S. Chakraborty, and M. Srivastava, "SenseGen: A deep learning architecture for synthetic sensor data generation," in 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp. 188–193, Mar. 2017.

[59] F. Alharbi, L. Ouarbya, and J. A. Ward, "Synthetic Sensor Data for Human Activity Recognition," in 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–9, July 2020. ISSN: 2161-4407.

[60] S. Norgaard, R. Saeedi, K. Sasani, and A. H. Gebremedhin, "Synthetic Sensor Data Generation for Health Applications: A Supervised Deep Learning Approach," in 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 1164–1167, July 2018. ISSN: 1558-4615.

[61] S. Harada, H. Hayashi, and S. Uchida, "Biosignal Generation and Latent Variable Analysis With Recurrent Generative Adversarial Networks," IEEE Access, vol. 7, pp. 144292–144302, 2019.

[62] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series Generative Adversarial Networks," in Advances in Neural Information Processing Systems (H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, pp. 5508–5518, Curran Associates, Inc., 2019.

[63] 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 Proceedings of the ACM Internet Measurement Conference, IMC '20, (New York, NY, USA), pp. 464–483, Association for Computing Machinery, Oct. 2020.

[64] P. Pinson, H. Madsen, H. A. Nielsen, G. Papaefthymiou, and B. Klöckl, "From probabilistic forecasts to statistical scenarios of short-term wind power production," Wind Energy, vol. 12, no. 1, pp. 51–62, 2009.

[65] Y. Wang, Y. Liu, and D. S. Kirschen, "Scenario Reduction With Submodular Optimization," IEEE Transactions on Power Systems, vol. 32, pp. 2479–2480, May 2017.

[66] Y. Chen, X. Wang, and B. Zhang, "An Unsupervised Deep Learning Approach for Scenario Forecasts," in 2018 Power Systems Computation Conference (PSCC), pp. 1–7, June 2018.

[67] C. Jiang, Y. Mao, Y. Chai, M. Yu, and S. Tao, "Scenario Generation for Wind Power Using Improved Generative Adversarial Networks," IEEE Access, vol. 6, pp. 62193–62203, 2018.

[68] J. Liang and W. Tang, "Sequence Generative Adversarial Networks for Wind Power Scenario Generation," IEEE Journal on Selected Areas in Communications, vol. 38, pp. 110–118, Jan. 2020.

[69] Y. Zhang, Q. Ai, F. Xiao, R. Hao, and T. Lu, "Typical wind power scenario generation for multiple wind farms using conditional improved Wasserstein generative adversarial network," International Journal of Electrical Power & Energy Systems, vol. 114, p. 105388, Jan. 2020.

[70] J. Qiao, T. Pu, and X. Wang, "Renewable scenario generation using stable and controllable generative adversarial networks with transparent latent space," Chinese Society for Electrical Engineering (CSEE) Journal of Power and Energy Systems, pp. 1–12, 2020.

[71] A. Ng and M. Jordan, "On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes," Advances in Neural Information Processing Systems, vol. 14, pp. 841–848, 2001.

[72] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in Advances in Neural Information Processing Systems (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, eds.), vol. 27, pp. 2672–2680, Curran Associates, Inc., 2014.

[73] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Computing Research Repository, vol. abs/1412.6980, 2015.

[74] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," in 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2242–2251, Oct. 2017. ISSN: 2380-7504.

[75] "Hot Aisle/Cold Aisle Layout." `https://www.energystar.gov/products/low_carbon_it_campaign/12_ways_save_energy_data_center/hot_aisle_cold_aisle_layout`.

[76] R. Rahmani, I. Moser, and M. Seyedmahmoudian, "A complete model for modular simulation of data centre power load," Computing Research Repository, vol. abs/1804.00703, 2018.

[77] P. Tsilingiris, "Thermophysical and transport properties of humid air at temperature range between 0 and 100 °c," Energy Conversion and Management, vol. 49, pp. 1098–1110, 2008.

[78] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in Proceedings of the 34th International Conference on Machine Learning (D. Precup and Y. W. Teh, eds.), vol. 70 of Proceedings of Machine Learning Research, (International Convention Centre, Sydney, Australia), pp. 214–223, PMLR, 06–11 Aug 2017.

[79] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in Advances in Neural Information Processing Systems (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, pp. 5767–5777, Curran Associates, Inc., 2017.

[80] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in International Conference on Learning Representations, 2018.

[81] 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 Advances in Neural Information Processing Systems (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, pp. 6626–6637, Curran Associates, Inc., 2017.

[82] Y. Yu, Z. Gong, P. Zhong, and J. Shan, "Unsupervised Representation Learning with Deep Convolutional Neural Network for Remote Sensing Images," in Image and Graphics (Y. Zhao, X. Kong, and D. Taubman, eds.), Lecture Notes in Computer Science, (Cham), pp. 97–108, Springer International Publishing, 2017.

[83] "NIPS 2016 Workshop on Adversarial Training - Soumith Chintala - How to train a GAN," Feb. 2017.

[84] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," Journal of Machine Learning Research, vol. 15, no. 56, pp. 1929–1958, 2014.

[85] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML'15, p. 448–456, JMLR.org, 2015.

[86] S. Arora, A. Risteski, and Y. Zhang, "Do GANs learn the distribution? some theory and empirics," in International Conference on Learning Representations, 2018.

[87] "Google Colaboratory." `https://colab.research.google.com/`.

[88] "TensorFlow." `https://www.tensorflow.org/`.

[89] "pandas - Python Data Analysis Library." https://pandas.pydata.org/.

[90] "scikit-learn: machine learning in Python — scikit-learn 0.23.2 documentation." https://scikit-learn.org/stable/.

[91] "NumPy." https://numpy.org/.

[92] "Matplotlib: Python plotting — Matplotlib 3.3.3 documentation." https://matplotlib.org/.

[93] "seaborn: statistical data visualization — seaborn 0.11.0 documentation." https://seaborn.pydata.org/.

[94] J. P. Sánchez, "Code GAN scenario forecasting." https://github.com/jaimeperezsanchez/GAN_Scenario_Forecasting.

[95] J. Zhuang, T. Tang, Y. Ding, S. Tatikonda, N. Dvornek, X. Papademetris, and J. Duncan, "Adabelief optimizer: Adapting stepsizes by the belief in observed gradients," Conference on Neural Information Processing Systems, 2020.

[96] C. Bowles, L. Chen, R. Guerrero, P. Bentley, R. N. Gunn, A. Hammers, D. A. Dickie, M. del C. Valdéz Hernández, J. M. Wardlaw, and D. Rueckert, "GAN Augmentation: Augmenting Training Data using Generative Adversarial Networks.," Computing Research Repository, vol. abs/1810.10863, 2018.

[97] 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.

[98] G. Mariani, F. Scheidegger, R. Istrate, C. Bekas, and C. Malossi, "BAGAN: Data Augmentation with Balancing GAN," arXiv:1803.09655, June 2018.

[99] X. Yi, E. Walia, and P. Babyn, "Generative adversarial network in medical imaging: A review," Medical Image Analysis, vol. 58, p. 101552, Dec. 2019.

Appendices

Appendix A

Ethic, Economic, Social and Environmental Aspects

A.1 Introduction

Our society is immersed in a new digital revolution led by emerging technologies such as the Internet of Things (IoT), Artificial Intelligence (AI), and Big Data. This radical transformation of our world points to a bright future but involves very significant side effects.

One of the most serious problems we must address is energy efficiency since it has important implications in the fight against climate change because of the CO_2 emissions when producing this energy. In the coming years, it is expected that the most relevant agent in the increase of energy demand related to the digital revolution will be the DCs [4]. This inevitably points to the urgent need to optimize these facilities.

To address this problem, once again, AI has enormous potential to help us. Data-based optimizations are revolutionizing the management and optimization of complex systems such as DCs. *Google* has reduced the cooling energy spent in its DCs by 40%, using AI-based techniques [3]. However, to apply this approach globally, it is necessary to collect massive amounts of data to feed the algorithms. Furthermore, collecting abundant data with significant variability is expensive, time-consuming, and can endanger the IT equipment's physical integrity.

This MSc Thesis seeks to solve this problem by using realistic synthetic data to feed the algorithms and improve DCs' optimization. Our research uses an algorithm that has offered impressive results during the last years, the Generative Adversarial Networks (GAN). The production of quality synthetic data could dramatically reduce the economic and temporal cost of implementing AI-based optimization algorithms, thus enabling a greener and more sustainable future.

A.2 Description of Relevant Impacts Related to the Project

The major impacts that involve this MSc Thesis are divided as follows:

- **Economic Impact**: The generation of realistic synthetic data lowers the costs of implementing AI-based optimization algorithms, both in time and economics. Besides, the very use of these optimization algorithms can have a considerable impact on the energy efficiency of complex systems such as DCs. Moreover, our proposal can generate anomalous situations on demand. Reducing the possible damage to IT equipment, and therefore, considerably reducing the economic cost.
- **Social Impact**: The field of research in synthetic data generation can significantly impact many aspects of society that involve the use of AI, accelerating the adoption of this technology more widely. Data synthesis can provide realistic data to work with, efficiently, and at a scale.
- **Environmental Impact**: Synthetic data generation enables faster adoption and higher performance of optimization algorithms. This implies significant improvements in the energy efficiency of complex facilities, and therefore, reducing CO_2 emissions to the atmosphere, which plays an essential role in the fight against climate change.
- **Legal Ethics Impact**: The use of synthetic data can also have a relevant impact in sectors where data privacy is a concern. In our use case, it reduces information leakage compared to using only real data. Hence, encouraging the exchange of data between institutions and organizations, without compromising privacy and avoiding the malicious use of data.

A.3 Conclusions

This work provides value to synthetic data generation's research field through the use of novel algorithms such as Generative Adversarial Networks (GAN). The ICT sector must adapt to the growing demand for innovative applications that can change people's way of life for the better, yet also present a wide range of requisites and challenges. Optimizing DC facilities is a fundamental step in establishing sustainable social and economic growth necessary for an environmentally friendly future.

Appendix B

Budget

B.1 Budget of Material Execution

Table B.1: Budget of material execution.

HARDWARE RESOURCES				
Concept	Cost	Usage (months)	Depreciation	Real Cost
Laptop Lenovo Ideapad	700.00	6	20%	40.00
Total cost:				40.00
LABOUR COSTS				
Position		Yearly Cost	Usage (months)	Real Cost
Graduated Engineer		24,500	6	12,250.00
Total Cost:				12,250.00
SOFTWARE				
Concept		Cost Per Month	Usage (months)	Total Cost
Google Colab Pro Subscription		8.60	4	34.40
Total Cost:				34.40
TOTAL COST:				**12,324.40**

B.2 General Expenses and Industrial Benefits

Table B.2: General expenses and industrial benefits.

Concept	Cost
Budget of Material Execution (BME)	12,324.40
General expenses (16% of BME)	1,971.90
Industrial benefits (6% of BME)	739.46
Subtotal Budget:	**15,035.76**

B.3 Total Budget

Table B.3: Total budget.

Subtotal Budget	15,035.76
VAT (21% of Subtotal Budget)	3,157.51
Total Budget:	**18,193.27€**