DEGREE PROJECT IN INFORMATION AND COMMUNICATION TECHNOLOGY,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, *SWEDEN 2018*

Generation of Synthetic Data with Generative Adversarial Networks

DOUGLAS GARCIA TORRES

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Abstract

The aim of synthetic data generation is to provide data that is not real for cases where the use of real data is somehow limited. For example, when there is a need for larger volumes of data, when the data is sensitive to use, or simply when it is hard to get access to the real data. Traditional methods of synthetic data generation use techniques that do not intend to replicate important statistical properties of the original data. Properties such as the distribution, the patterns or the correlation between variables, are often omitted. Moreover, most of the existing tools and approaches require a great deal of user-defined rules and do not make use of advanced techniques like Machine Learning or Deep Learning. While Machine Learning is an innovative area of Artificial Intelligence and Computer Science that uses statistical techniques to give computers the ability to learn from data, Deep Learning is a closely related field based on learning data representations, which may serve useful for the task of synthetic data generation.

This thesis focuses on one of the most interesting and promising innovations of the last years in the Machine Learning community: Generative Adversarial Networks. An approach for generating discrete, continuous or text synthetic data with Generative Adversarial Networks is proposed, tested, evaluated and compared with a baseline approach. The results prove the feasibility and show the advantages and disadvantages of using this framework. Despite its high demand for computational resources, a Generative Adversarial Networks framework is capable of generating quality synthetic data that preserves the statistical properties of a given dataset.

Keywords
Synthetic Data Generation, Generative Adversarial Networks, Machine Learning, Deep Learning, Neural Networks

Abstract

Syftet med syntetisk datagenerering är att tillhandahålla data som inte är verkliga i fall där användningen av reella data på något sätt är begränsad. Till exempel, när det finns behov av större datamängder, när data är känsliga för användning, eller helt enkelt när det är svårt att få tillgång till den verkliga data. Traditionella metoder för syntetiska datagenererande använder tekniker som inte avser att replikera viktiga statistiska egenskaper hos de ursprungliga data. Egenskaper som fördelningen, mönstren eller korrelationen mellan variabler utelämnas ofta. Dessutom kräver de flesta av de befintliga verktygen och metoderna en hel del användardefinierade regler och använder inte avancerade tekniker som Machine Learning eller Deep Learning. Machine Learning är ett innovativt område för artificiell intelligens och datavetenskap som använder statistiska tekniker för att ge datorer möjlighet att lära av data. Deep Learning ett närbesläktat fält baserat på inlärningsdatapresentationer, vilket kan vara användbart för att generera syntetisk data.

Denna avhandling fokuserar på en av de mest intressanta och lovande innovationerna från de senaste åren i Machine Learning-samhället: Generative Adversarial Networks. Generative Adversarial Networks är ett tillvägagångssätt för att generera diskret, kontinuerlig eller textsyntetisk data som föreslås, testas, utvärderas och jämförs med en baslinjemetod. Resultaten visar genomförbarheten och visar fördelarna och nackdelarna med att använda denna metod. Trots dess stora efterfrågan på beräkningsresurser kan ett generativt adversarialnätverk skapa generell syntetisk data som bevarar de statistiska egenskaperna hos ett visst dataset.

Nyckelord
Syntetisk Datagenerering, Generativa Adversariella Nätverk, Maskinlärande, Djupt Lärande, Neurala Nätverk.

Contents

1 Introduction **1**
- 1.1 Problem . . . 2
- 1.2 Purpose . . . 3
- 1.3 Goals . . . 3
- 1.4 Benefits, ethics and sustainability . . . 3
- 1.5 Research methodology . . . 4
- 1.6 Delimitations . . . 5
- 1.7 Outline . . . 5

2 Background **6**
- 2.1 Machine Learning . . . 6
- 2.2 Deep Learning . . . 7
- 2.3 Deep Generative Models . . . 8
- 2.4 Generative Adversarial Networks . . . 9
- 2.5 Related work . . . 12

3 Methodology **14**
- 3.1 Data collection . . . 14
- 3.2 Data analysis . . . 16
- 3.3 Quality assurance . . . 17
  - 3.3.1 Validity, reliability and replicability . . . 17

4 Generation of Synthetic Data with GANs **19**
- 4.1 The data generation process . . . 19
  - 4.1.1 Schema detection . . . 20
  - 4.1.2 Pattern analysis . . . 21
  - 4.1.3 Feature engineering . . . 21
  - 4.1.4 Model training . . . 22
  - 4.1.5 Data production . . . 22
  - 4.1.6 Feature reverse-engineering . . . 22

4.2 Implementation 22
4.2.1 A GAN-based synthetic data generator 23
4.2.2 A baseline synthetic data generator 27
4.3 Experiments 28
4.3.1 Test system 29
4.3.2 Software 29
4.4 Evaluation 29
4.4.1 Overall metrics 30
4.4.2 Correlation (Euclidean) distance 30
4.4.3 Wasserstein distance 32
4.4.4 Perplexity 32

5 Analysis and results **34**
5.1 Overall analysis and results 34
5.2 Efficiency 35
5.2.1 Training time 36
5.2.2 Data generation time 37
5.3 Preserving the data distribution 38
5.4 Preserving the correlation patterns 39
5.5 Generating quality text 42

6 Conclusions and future work **46**
6.1 Discussion 48
6.2 Stakeholders feedback 50
6.3 Future work 51

Bibliography **53**

A Results of case study 1 **57**

B Results of case study 2 **62**

C Results of case study 3 **69**

Chapter 1

Introduction

It is a common practice to use real-world data for the evaluation or demonstration of new technologies in areas such as software development, data analytics or machine learning. However, there are important constraints when using real data for such purposes. These limitations range from the difficulty to obtain the data, the need for large volumes of records –e.g. to train Deep Learning (Artificial Intelligence) models–, or privacy concerns when the data is sensible –e.g. in the case of customer data.

A synthetic data generation mechanism can serve useful in many situations. Researchers, engineers, and software developers can make use of safe datasets without accessing the original real-world data, keeping them apart from privacy and security concerns as well as letting them generate larger data sets that would not even be available using real data [13]. Furthermore, the latest innovations in machine learning and deep learning techniques seem promising to help generate synthetic data with higher quality and in larger quantities.

By the year 2014, the most striking successes in deep learning had involved discriminative models, i.e. models naturally used to classify a given sample of data into a class. Meanwhile, due to implementation difficulties, generative models have had less impact. These are models that starting from a data sample, can generate similar data [27]. In this context, Generative Adversarial Networks (GANs) were introduced as an idea of a new generative model estimation procedure that sidesteps the implementation difficulties [18]. GANs can be thought of as anal-

ogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency.

In the latest years, the original GAN model has been studied and improved by several researchers. Many new ideas have been proposed to enhance their performance [26] [1], to add new features like the ability to generate more types of data [20] [2] [14] [5] [37], or the ability to take the surrounding context and semantics to get higher quality results [11]. In summary, GANs have emerged as a powerful framework for learning complex data distributions and produce samples which are as close to real data as possible.

1.1 Problem

The aim of this thesis is to generate synthetic data using Generative Adversarial Networks, which is a non-trivial task for various reasons. To begin with, Generative Adversarial Networks has been proven to successfully generate new data representing images (i.e. continuous numeric values), but have not been extensively tested for the case of most common types of data, like discrete categorical data or text data. Moreover, there are many considerations and different approaches when it comes to generating synthetic data [32]. Most synthetic data generators [13] [24] require a great deal of user-defined specifications with knowledge of the underlying distribution of the data to be generated. In addition, these approaches do not guarantee that the resulting datasets provide the desired data distribution and attribute correlations [32].

Under this context, the research question addressed by this work is: *Can Generative Adversarial Networks be used to generate synthetic continuous, discrete and text data, effectively and efficiently, while preserving the underlying distribution and patterns of the real data?*.

In the research question above it is important to point out that the effectiveness and efficiency of the solution involve the notion of high quality of the results –in this case, a high similarity with the real dataset– and low use of resources (i.e. training/running time or processing power required).

1.2 Purpose

The purpose of this thesis is to design, implement and test a synthetic data generator based on Generative Adversarial Networks. For the purpose of a proof-of-concept, sample datasets are used to train the models of the proposed solution that subsequently will generate new synthetic data. The result is an analysis that compares the proposed GAN-based framework with a simpler baseline generator in order to conclude with the advantages and disadvantages of using Generative Adversarial Networks for synthetic data generation.

1.3 Goals

The aim of this project is to aid the development of a synthetic data generator such that it requires minimal user interaction –i.e. definition of rules and specifications– while generates quality synthetic data with the same patterns and the underlying distribution of a real dataset. Considering that synthetic data generation has received more focus recently as it helps to validate new technologies more quickly, also preserving the privacy related to the original data, the hope is to come up with better data generation tools in terms of both quality and quantity of the output, but also better in terms of usability. Therefrom, this project evaluates whether an innovative approach like Generative Adversarial Networks can be useful to create better data generation tools.

1.4 Benefits, ethics and sustainability

With the ongoing advances in Machine Learning, Artificial Intelligence and the handling of Big Data, new issues and concerns have risen in relation to the use of private and sensible information of individuals. These concerns have promoted new regulations [17] that are aiming to make these advanced technologies more sustainable. On the other hand, the fields of data analytics, Artificial Intelligence, or traditional software engineering, often require large amounts of data to develop innovative solutions. In this line of reasoning, a data generation tool can be a valuable asset that benefits both parts –individuals and researchers– while making more sustainable the processes of development and validation of new technologies. These benefits are in

line with the Sustainable Development Goals (SDGs) set down by the United Nations [33]. More specifically with the 9th goal: *"Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation"*.

1.5 Research methodology

The framework of *Research Methods and Methodologies for Research Projects and Degree Projects* (see figure 1.1) proposed by Anne Håkansson [22], is a referential tool that can be used to choose and apply the most suitable methods to execute a research. According to this portal, the basic categories of research methodologies are *qualitative* and *quantitative*.

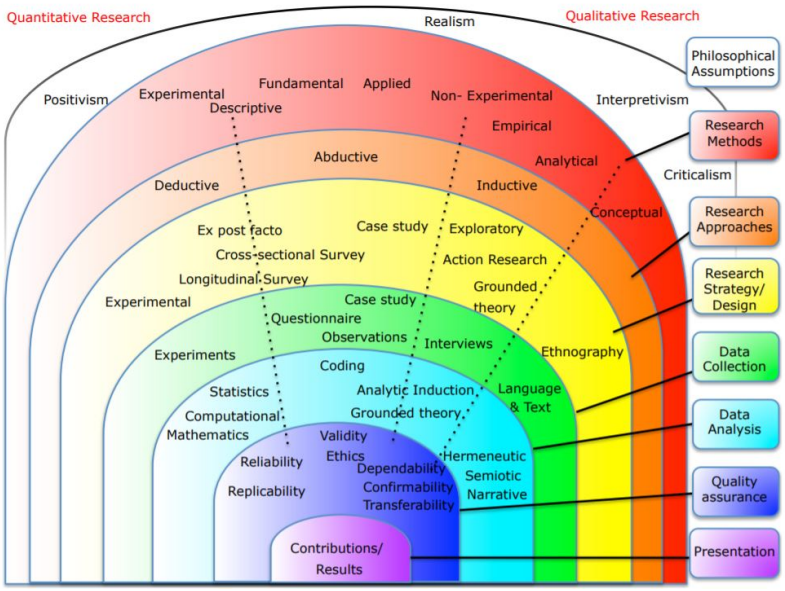


Figure 1.1: The portal of research methods and methodologies, by Anne Håkansson. The left side corresponds to quantitative methodologies. The right side belongs to qualitative methods [22].

Since the outcome of this thesis is to find out the feasibility, advantages and disadvantages of a new technology applied to a specific problem, *quantitative* results will be collected. Accordingly, the "curiosity-driven" nature of this study, makes it suitable for applying a *fundamental research* method in order to accomplish the research tasks and generate new ideas for a new solution to the old problem of synthetic data

generation. This is complemented by an *applied research* method, that builds on existing related work, applying those ideas to solve tasks in a practical way. Therefrom, the employed research strategies and designs –which are the guidelines for organizing, planning, designing and conducting the research– are related to a *creative research method*, involving the design and development of a procedure to generate synthetic data. The data collection method (and the research strategy) is limited to the in-depth analysis of particular case studies of selected customer-related data sets (i.e. *the case study method*). And due to the quantitative essence of the results collected, an *abductive* approach is used to draw conclusions and answer the research questions.

1.6 Delimitations

The outcome of this thesis is meant to be the results and conclusions of the analysis instead of the development of a software tool. Nevertheless, for the interest of the parts involved in this project, the solution should be designed considering the resources and limitations of Microsoft's Cloud Enterprise Software [25]. In addition, this work does not focus on data privacy aspects or in the anonymization of sensible data. The outcome of this work is a generator of realistic but synthetic data, hence it does not have to be subject of privacy concerns.

1.7 Outline

The rest of this thesis is organized as follows. The next chapter formally introduces the concept of Generative Adversarial Networks along with related work towards synthetic data generation that is relevant to this project. Then, in chapter 3, the methodology chosen to approach the problem is presented with the data collection details. Chapter 4 explains how to generate synthetic data using GANs, along with the designed experiments to evaluate the solution. The analysis over the results, with a comparison between the built solution and a simpler baseline data generation approach, is provided in chapter 5. Finally, chapter 6 summarizes the conclusions, answers the research questions, and suggests a future line of work.

Chapter 2

Background

The McGraw-Hill Dictionary of Scientific and Technical Terms defines synthetic data as *"any production data applicable to a given situation that are not obtained by direct measurement"* [28]. Its use has been traditionally tied to the field of data anonymization. However, there are many other uses and motives to generate synthetic data and meet specific needs or certain conditions that real data cannot fulfill. Synthetic data generation has received considerable focus in the recent years not only for its usage in the test and validation of new software technologies but also for maintaining the privacy of confidential data. There are many different approaches for generating either fully synthetic –without taking any data sample as an input– or partially synthetic datasets –using and even including sample data in the final outcome. Nevertheless, all the approaches have certain limitations, and interestingly, few of them make use of the latest innovations in the fields of Machine Learning and Artificial Intelligence [36].

2.1 Machine Learning

Machine Learning is an area within the fields of Artificial Intelligence and Computer Science that provides computer systems the ability to learn by using data and advanced statistical methods, without the necessity of being explicitly programmed [31]. Briefly explained, Machine Learning consists in the creation of models that receive data as an input and usually returns a classification, a label, or a prediction related to the input. The data received by the machine learning model has to be encoded and structured in a specific format that it can under-

stand and process. Typically, the input structure is a multidimensional array where often the first dimension corresponds to an instance (e.g. a customer) and a second dimension corresponds to the characteristics of that instance (e.g. name, birthday, address, etc.). The traditional tasks that machine learning models perform are classification, regression, prediction, and clustering.

The machine learning model is tied to a learning algorithm that performs the learning using a set of data (i.e. the training data). The traditional machine learning algorithms are widely used in the industry and effectively provide solutions to a great variety of problems. Nevertheless, there are still many tasks –like common Artificial Intelligence problems such as object or speech recognition– where they have not achieved the required performance. This is were the closely related field of *Deep Learning* emerged with innovative solutions [19].

2.2 Deep Learning

Deep Learning techniques are based on the concept of *Artificial Neural Networks*, which are networks of connected layers of nodes (neurons) inspired by the biological neural networks of animal brains. These networks are called *deep* because they are composed of more than 2 layers: an input layer, an output layer, and at least one hidden layer that enable the network to learn complex non-linear functions required to carry out the complicated tasks of Artificial Intelligence.

There are multiple types of artificial neural networks, and their implementation differences makes them suitable to solve specific problems more efficiently. The simplest and most common type is called the *feed-forward network* since the data flow from the input layer to the output layer (figure 2.1). Despite their simplicity, these networks are useful for tasks such as classification, regression, or even image segmentation. Note that in this simple network, there are no feedback or recurrent connections within the neurons.

Another type of neural networks that will be mentioned later in this thesis is called *recurrent neural networks* (figure 2.2), because instead of

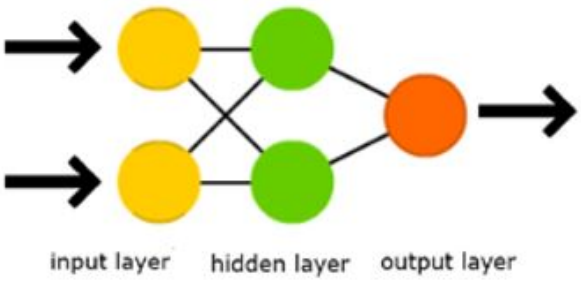


Figure 2.1: A Feed Forward Neural Network, where data flows from the input layer to the output layer. Adapted from [34].

having only neurons with connections from the input to the output, it also has neurons with connections from the output, again to the input. This additional connection can be used to store information over time providing the network with dynamic memory, a feature that makes them particularly useful for time series predictions, natural language processing, and text generation [12].

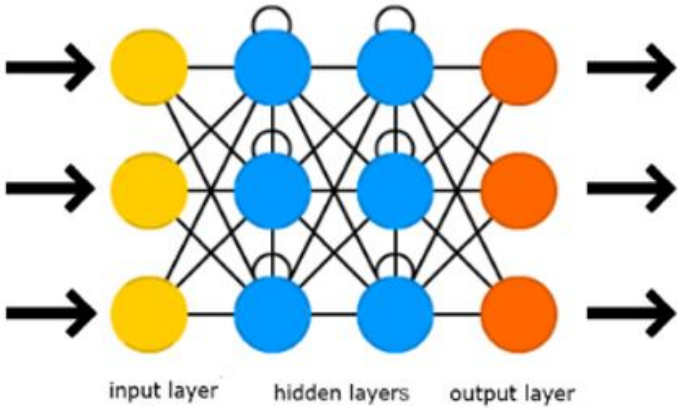


Figure 2.2: A Recurrent Neural Network, instead of having only neurons with connections from the input to the output, it also has neurons with connections from the output, again to the input. Adapted from [34].

2.3 Deep Generative Models

According to the probability and statistics theory, there are two types of probabilistic models: the so-called generative models, and the dis-

criminative models [36]. By the year 2014, the most popular successes in Deep Learning usually involved discriminative models, i.e. models naturally used to classify a given sample of data into a class. Meanwhile, generative models that generate data from a given sample, have had less impact, arguably because of the complexity to implement them [27].

One distinction that is usually made between deep learning and machine learning models is whether they perform supervised or unsupervised learning. While most discriminative models fall in the category of *supervised learning*, which means that they require labeled data in order to learn (usually labeled by humans), most deep generative models are categorized as *unsupervised*, which means they do not require labeled data[36]. This is something that makes deep generative models very attractive in the paths of automation and artificial intelligence.

Deep generative models have multiple short-term applications like image de-noising, image resolution enhancement, or the reconstruction of deteriorated parts of an image [36]. And at the end, they hold the potential to automatically learn all the features of any type of dataset, of any kind of data.

2.4 Generative Adversarial Networks

The Generative Adversarial Networks (GAN) idea was introduced in 2014 as a new generative framework that sidesteps the implementation difficulties of generative models. These have demonstrated impressive results producing images, such as pictures of faces and bedrooms.

The Generative Adversarial Network proposed by *Goodfellow et al.* is a machine learning model made up of to components: a *generator* G, and a *discriminator* D. This framework results in a generative network after an adversarial game in which the two models, D and G, are simultaneously trained: the generative model G captures the data distribution, and the discriminative model D estimates the probability that a sample came from the training data rather than G. Formally, the goal of

this adversarial game can be expressed as:

$$\min_G \max_D \mathbb{E}_{x\ p_{data}}[\log D(x)] + \mathbb{E}_{z\ p_{(z)}}[\log(1 - D(G(z)))]$$

Where the generator function *G*(z) maps samples from *p*(z) to the data space while it is trained to confuse the discriminator to believe that these samples come from the original data distribution p_{data}.

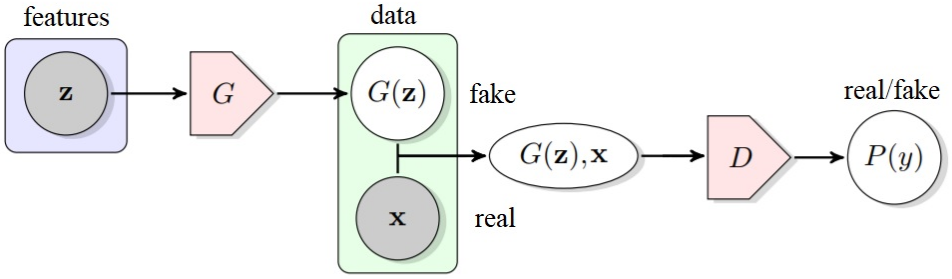


Figure 2.3: The structure of Generative Adversarial Networks. The generator model is trained to produce synthetic data *G(z)*, that can confuse the discriminator model to estimate with a high probability *P(y)* that these samples come from an original data distribution.

In simpler words, *Goodfellow et al.* explain in the paper that the generative model can be thought of as analogous to a team of counterfeiters, who are trying to produce fake currency and use it without being detected. Meanwhile, the discriminative model is analogous to the police, who try to detect the fake currency. The competition in this game drives both parts to improve their methods until the counterfeits cannot be distinguished from the genuine currency [18].

After the publication of the original paper proposed by *Goodfellow et al.* further researches have demonstrated many drawbacks of the GAN model. On one side, it does not offer the possibility to control what data to generate, nor the possibility to generate categorical data. On the other side, GANs are well known to be complex machine learning models to train.

One of the first approaches to improve the GAN model was published by *Mirza et al.* with the title of *Conditional Generative Adversarial Nets*

[26]. A work that shows the possibility to control what data to generate by simply adding as an input –of both the generator and the discriminator– a one-hot encoded vector indicating which class to generate. Years later, the *Wasserstein GAN* was proposed by *Arjovsky et al.* claiming to provide training stability and interpretability to the original GAN model [1]. In addition, later researches showed that using the Wasserstein approach, also provides the GAN with the ability to generate categorical data just by having a corresponding *Softmax* output in the generator network with a dimensionality equal to the number of possible discrete values for each categorical variable [20] [2].

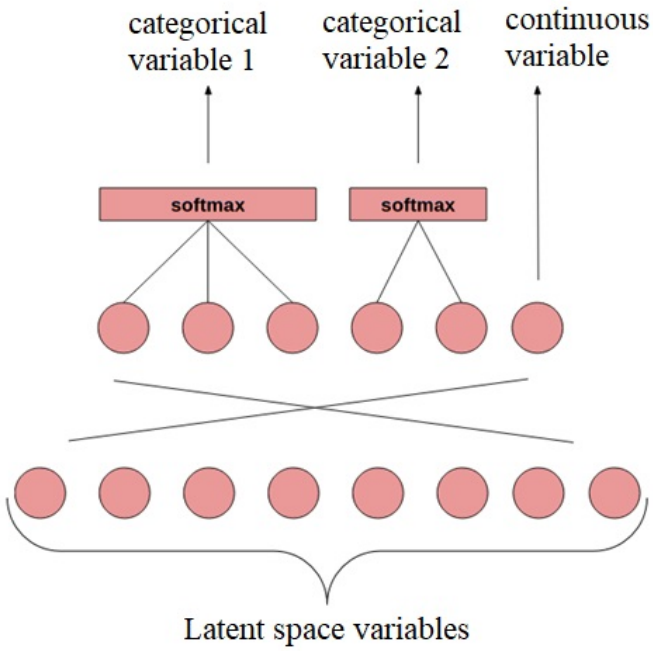
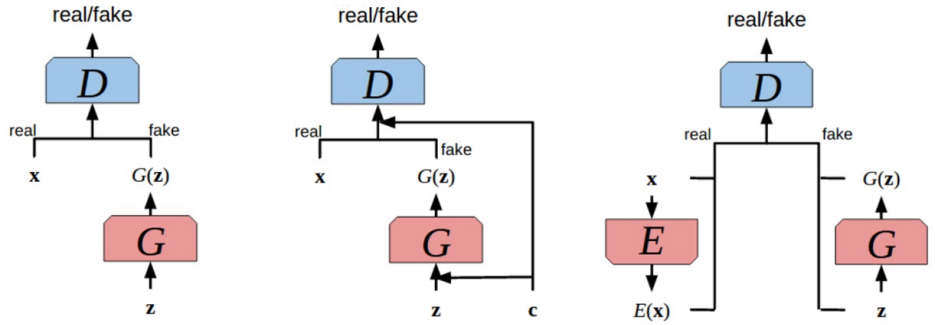


Figure 2.4: Example of a generator of mixed categorical and continuous variables. Here, categorical variable 1 takes 1 of 3 possible values, categorical variable 2 takes 1 of 2 possible values and there's one continuous variable. Adapted from [15].

From there, other alternative GAN architectures have contributed to improve the original model in significantly useful ways. One of these contributions was published in a paper titled *Adversarial Feature Learning* which proposes the *Bidirectional GAN*, a model with the means to access the latent space representation of the data [11]. A study that is motivated on the idea that the latent space of generative models cap-

tures rich semantic information. Therefore, accessing these semantic latent representation may serve useful not only for controlling what data to generate, but also for feature representation in tasks where semantics are relevant.



a) Generative Adversarial Networks b) Conditional GAN c) Bidirectional GAN

Figure 2.5: Simplified structures of a) an original GAN framework [19], b) a Conditional GAN introducing an additional input of data [26], and c) a Bidirectional GAN, which introduces an encoder E that should learn to invert the generator G [11]. Adapted from [15].

2.5 Related work

In March of 2017, *Surendra et al.* published a paper in the *International Journal of Scientific and Technology Research* [32] reviewing several synthetic data generation methods. This paper concludes that the main limitations of most of the studies reviewed are that they require a great deal of user interaction and expertise. Most of the synthetic data generator approaches mentioned require the definition of a detailed set of rules and constraints prior to the data generation. In addition, they require the users to have a good understanding of the domain of the data.

Most of these approaches were not based on generative machine learning models, they were based on simpler algorithms for the development of synthetic data generators. Two of the first (and older) approaches mentioned could be also two of the most relevant, since they address a wide possibilities of data types. These were proposed by *Lin et al.* in *Development of a Synthetic Data Set Generator for Building and*

Testing Information Discovery Systems [24], and *Eno et al.* in their work of *Generating Synthetic Data to Match Data Mining Patterns* [13]. The latter defined a method that confirmed the viability of using data mining models and inverse mapping to inject realistic patterns into synthetic data sets, while the first one proposed an architecture for a tool that generates synthetic datasets on a to-be-decided semantic graph, and also developed a prototype capable of generating synthetic data for a particular scenario of credit card transactions.

More recently, it is possible to find similar approaches but using machine learning techniques. For example, in *Towards a Synthetic Data Generator for Matching Decision Trees*, *Peng et al.* [29] propose an algorithm, an architecture and the design of a tool that is able to generate datasets with intrinsic patterns, claiming the possibility to create almost a million rows in a few seconds using a laptop with basic specifications.

There have been also some studies focused on the generation of specific kinds of data –other than images– using Generative Adversarial Networks. To begin with, *Esteban et al.* experimented on replacing the multi-layer perceptron in the original GAN model with a Recurrent Neural Network (RNN) to generate real-valued time-series medical data in a conditional setting [14]. On the same line, and motivated by privacy concerns, *Choi et al.* proposed a new model called *medical Generative Adversarial Network (medGAN)* in order to generate realistic synthetic patient records –including high-dimensional discrete variables– via a combination of an auto-encoder and a GAN [5]. However, *Yu et al.* claim that their *Sequence Generative Adversarial Nets with Policy Gradient (SeqGAN)* is the first work extending GANs to generate discrete tokens of data. An approach that involves the use of reinforcement learning techniques in the generator [37].

In summary, the context of the studies presented in this section suggests that the idea of developing a synthetic data generation tool based on deep generative machine learning models such as GANs, that considers continuous, categorical, and also text data, definitely require more research that is bound to be done.

Chapter 3

Methodology

This chapter presents the chosen methods to execute the research. The data collection is explained in detail with selected case studies for testing and validating the proposed data generators. Then, the data analysis methods are introduced along with a last section discussing the quality assurance.

3.1 Data collection

In a quantitative research, the most commonly used data collection methods are experiments, questionnaires, case studies, and observations [22]. As the idea of this thesis is basically to evaluate the performance of a tool, a simple and suitable method is to select various case studies that comprise as most scenarios as possible (i.e. types of data attributes) and perform an in-depth analysis of the performance of the tool for each case.

The built prototypes are tested with 3 different datasets. The main case study was provided arbitrarily from a customer database that contains 200 records of master data of individuals and their addresses. As can be seen in the Entity-relationship (ER) model of figure 3.1, the dataset is separated in two database tables: *Customer Main* and *Customer Address*. Each customer main record has at exactly one corresponding record in the customer address table. The relevant attributes of the provided dataset are 11, and they mainly correspond to identity data (i.e. name, gender, date of birth, etc.) and contact data (i.e. address, telephone number, postal code, etc.). While 4 of the attributes are

treated as categorical data (e.g. gender, city, country, and time zone), 1 is a date (continuous data), and the rest can be considered strings of characters with either a variable format (e.g. the address), or a well defined format (e.g. mobile phone number).

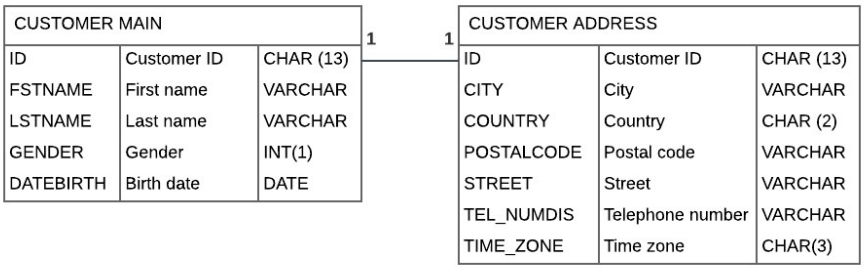


Figure 3.1: Entity-Relation (ER) model of the main selected case study, which is provided arbitrarily from a private customer database. Any original sample will remain anonymous.

The first alternative case study contemplated in this thesis is based on an experiment that was part of the KDD 2009 Cup, which offered the opportunity to work on large marketing databases from the French Telecom company Orange, to predict the probability of customers to churn (switch providers) or to buy new products and services [21].

The version of the dataset used in this thesis is also separated in two tables: *Customer Main* and *Customer Statistics*. Each customer main record has at least one corresponding record in the customer statistics table. However, it can have more than one related record. In total, the dataset contains 22 customer related attributes, from which 14 are numeric continuous values (real numbers and integers), and 8 are categorical variables. Therefore, it is a suitable dataset for considering other data types and other data generation scenarios that cannot be found in the main case of study. The Entity-relationship model of this dataset is shown on figure 3.2.

A second alternative case study is also considered (see figure 3.3). This is the case of the Credit Card Fraud detection dataset from Kaggle [8]. The dataset contains transactions made by credit cards in September 2013 by European cardholders. With 284,807 transactions that oc-

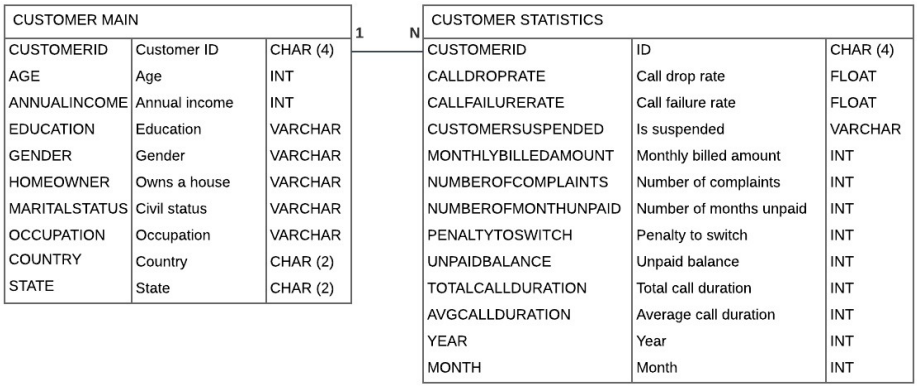


Figure 3.2: Entity-Relation (ER) model of a case based from the KDD 2009 Cup dataset, which offered the opportunity to work on large marketing databases from the French Telecom company Orange, to predict the probability of customers to churn (switch providers) or to buy new products and services [21].

curred in two days, having 492 frauds. The version of the dataset considered in this thesis contains 12 continuous values, including a time dimension, plus 1 label (categorical) attribute. The dataset is already anonymized and was specifically engineered for a machine learning exercise. Hence it contains implicit correlation patterns between the attributes and the label, however, it is highly unbalanced.

3.2 Data analysis

To analyze the collected experiment results, a process of inspecting, cleaning and transforming data has to be carried out to support the decision making and draw the conclusions. The most common analysis methods suitable for a quantitative research are statistics and computational mathematics [22]. While the later can be used to calculate numerical methods, modeling, and simulations, descriptive and inferential statistics may constitute a natural approach to evaluate the results of the case studies (samples) of the research.

In this thesis, for every measurable aspect of the research question, a specific metric is carefully selected to compare the results. All the

CREDIT CARD FRAUD		
TIME	Hour of the day	INT
V1	Statistic	FLOAT
V2	Statistic	FLOAT
V3	Statistic	FLOAT
V4	Statistic	FLOAT
V5	Statistic	FLOAT
V6	Statistic	FLOAT
V7	Statistic	FLOAT
V8	Statistic	FLOAT
V9	Statistic	FLOAT
V10	Statistic	FLOAT
AMOUNT	Statistic	FLOAT
CLASS	Fraud or not	BINARY

Figure 3.3: Entity-Relation (ER) model of a credit card fraud detection dataset from Kaggle [8]. The dataset contains transactions made by credit cards in September 2013 by European cardholders.

statistics and metrics used for this analysis are explained in detail in section 4.4.

3.3 Quality assurance

Based on the portal of research methods and methodologies [22], the quality assurance is defined as the validation and verification of the research material. It also states that a quantitative research should discuss aspects as the validity, the reliability, and the replicability of the results. While the validity aspect makes sure that the research measures what is expected to be measured, the reliability refers to the consistency of the results, and the replicability is the possibility to repeat the experiments and reach similar results.

3.3.1 Validity, reliability and replicability

In order to guarantee a level of reliability in the designed experiments, each experiment variant is executed 3 times, taking the average of each

one of the metrics. The validity of the proposed synthetic data generators is evaluated by running them over 3 different and unrelated datasets, making use of different metrics for every aspect to be measured. And concerning replicability, before every experiment is performed, the random seed –which is the internal state of the random number generator associated to the programming language engine– is set to the constant zero. In addition, all the experimental setup variables are specified in detail in the next chapter.

Chapter 4

Generation of Synthetic Data with GANs

Generative Adversarial Networks, as well as other generative machine learning techniques, can be used for synthetic data generation. The main contribution of this research is the design and testing of an approach for generating synthetic data with a GAN framework. Nevertheless, a generic data generation process –in which the GAN framework is applied– is also proposed in this section. Additionally, an alternative approach to the GAN-based framework is proposed as a baseline for comparison purposes.

The overall data generation process is extensively described in section 4.1., while the core implementation details are explained in section 4.2. Then, the experimental framework is presented. This includes the setup with the specifications of the used hardware and software, and the evaluation framework with the metrics chosen for the analysis.

4.1 The data generation process

In this project, the data generation process is designed to be executed in 8 subsequent steps. The input is a set of two-dimensional related structures corresponding to the datasets to generate, while the output is a similar set, with the same format, filled with synthetic data. The first processing step starts by detecting the data schema and the data types. Next, a pattern analysis is performed to detect the co-relations between the data attributes. This is followed by a feature engineer-

ing process so that the input data can be understood by the machine learning models and the statistical functions that are used to generate data. Then, the required machine learning models are trained, saved and validated. After that, the data production step is executed using the best-trained models, and finally, a process of feature reverse-engineering is carried out so that the output data is presented in the same format of the input data. An overview of this process is visible in figure 4.1 and each step is explained thoroughly in the following sections.

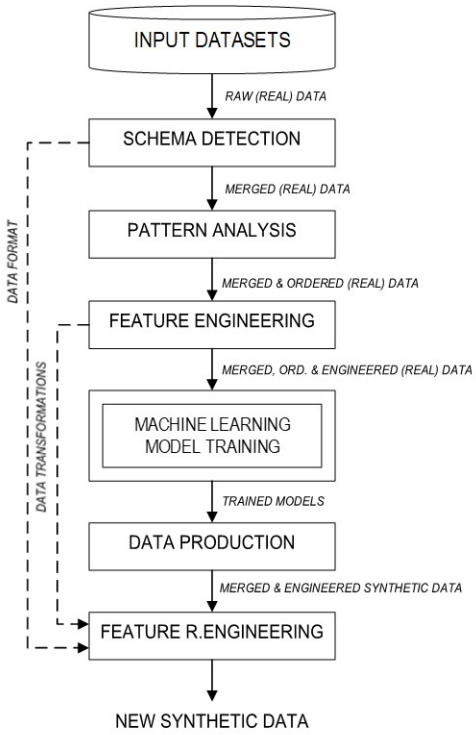


Figure 4.1: The proposed data generation process consist of 6 generic steps. In the fourth step (Machine Learning model training) different machine learning/statistical approaches can be applied.

4.1.1 Schema detection

The first step after the input data is entered to the processing pipeline, consist on an automatic schema detection of the datasets. This involves detecting the technical data types of each attribute (e.g. Inte-

$$z = \frac{X - \mu}{\sigma}$$

Figure 4.2: All continuous variables of the training data are standardized by removing the mean and scaling to a unit of variance.

ger, Boolean, String of characters, etc.) and also detecting whether the values of each attribute are continuous, categorical or free text.

4.1.2 Pattern analysis

The main idea behind this step is to detect the correlations between the attributes of the dataset. This is calculated as a correlation matrix with two intentions in mind. The first one is to determine the level of influence of each attribute over the rest, then establish which attributes are going to be used as influential conditions for each machine learning model, and finally determine a precedence order in which these attributes are going to be trained. The second is for evaluation purposes, to compare the correlations matrices of the original dataset with the generated datasets.

4.1.3 Feature engineering

The feature engineering process consists of encoding all the attributes of the dataset so that they can be understood and best processed by any machine learning or statistical model. This process involves various steps. The first one is to detect the attributes with skewed values in the dataset and perform any necessary transformations (e.g. logarithmic or cube root). Then, knowing that having standardized values of a dataset is a requirement for many machine learning models –since they might behave badly if the data does not look like normally distributed data–, the continuous attributes are standardized to have a mean equal to zero and a standard deviation equal to one (see the equation in figure 4.2). Also, when categorical values are involved, these are encoded as one-hot-encoded (OHE) vectors. And lastly, if text attributes are involved, these are also encoded as OHE vectors with the corresponding possible dictionary character values.

4.1.4 Model training

This step is relevant if the data generator is based on machine learning models. For each dataset attribute, one model is created and trained with a random sample subset of the original data. The training takes place for a pre-specified number of iterations, saving the corresponding weights and losses every pre-specified interval of steps, so that the best performing model states (according to a pre-defined criteria) can be selected for the data production stage.

4.1.5 Data production

Once the models are trained and in a productive state, it is possible to generate new synthetic records for the defined dataset. The new synthetic dataset is going to be generated attribute by attribute, in the sequential order determined in step 4.1.2. The first input of the process is the number of records to be produced. Then, for each attribute i in the dataset, a model M_i is selected according to pre-defined criteria (e.g. the last trained step or the best step of the model in terms of the loss). Its input is in the form of a vector V_i filled with either training, or noisy data –depending on the data generation approach used–, and an optional multidimensional conditional vector C_i, which is filled with synthetically generated data of the previously generated attributes. Finally, the output of each model M_i is another vector O_i with a new synthetic set of records for the attribute i that is immediately concatenated with C_i. This algorithm is presented in figure 4.3.

4.1.6 Feature reverse-engineering

After the new data is produced, all the transformations performed in the step 4.1.3 have to be reversed. Then, before the data generation process is completed, the new synthetic de-normalized dataset is reshaped to have the original format of the input.

4.2 Implementation

This section explains in a technical way the implementation of the non-trivial and core parts of the two (2) proposed data generator frameworks. A high level overview of both implementations can be ob-

$DataGeneration(N, Cr, realdata)$:

1: $\triangleright$ N *is the number of records to generate*
2: $\triangleright$ Cr *is the criteria for the model selection*
3: $\triangleright$ $realdata$ *is the training dataset*
4: **for each** *attribute* i **in** $realdata$ **do**
5: $\quad Ci \leftarrow newdata$
6: $\quad Vi \leftarrow getInput(N, realdata[i])$
7: $\quad Mi \leftarrow getModel(i, Cr)$
8: $\quad Oi \leftarrow generateData(Mi, Ci, Vi)$
9: $\quad newdata \leftarrow concatenate(Ci, Oi)$
10: **end for**
11: **return** $newdata$

Figure 4.3: The proposed algorithm for data generation is a generic iterative approach to generate new structured data, one attribute (dimension/column) per each iteration.

served in figures 4.4 and 4.6. The non-technical reader can omit it without missing any information required to understand the rest of this document.

4.2.1 A GAN-based synthetic data generator

The main approach of this thesis is based on Generative Adversarial Networks. However, for reasons that will be more clear in the next paragraphs, this implementation does not follow the original "vanilla" architecture of GANs. Instead, two variants are implemented: a Wasserstein GAN (WGAN) and a Wasserstein Conditional GAN (WCGAN).

Loss function

In a vanilla Generative Adversarial Network, the error between the output of the discriminator and the real labels is determined with the cross-entropy loss. This implies that what gets measured, is specifically how accurate is the discriminator when it comes to classifying what is real and what is fake. Nevertheless, as *Arjovsky et al.* showed in the *Wasserstein GAN* paper [1], the cross-entropy as a loss function turns out to be unstable for GANs training. On the other hand, this paper also demonstrates that the Wasserstein distance –which instead measures how different are the distributions of the real and the

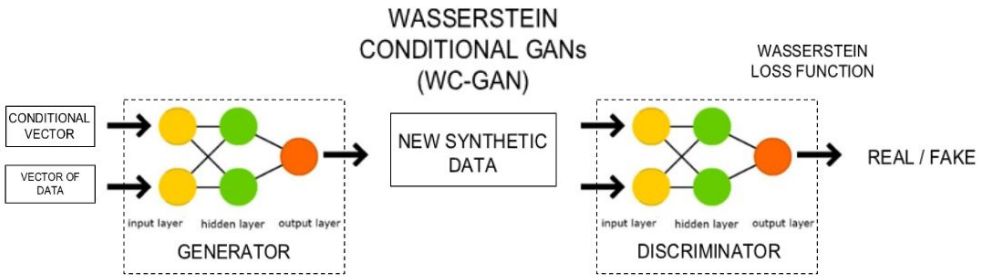


Figure 4.4: The GAN-based data generator is implemented with a Wasserstein Conditional GAN (WC-GAN). Both the generator and discriminator are feed-forward neural networks. The discriminator receives data and a conditional vector as an input, and outputs new synthetic data. The discriminator receives the output of the generator and outputs its probability to be real or fake data.

generated data– performs better in many cases. In fact, in the model proposed by *Arjovsky et al.*, the discriminator is removed, and a new model is introduced as "the critic", which intuitively will try to tell whether the data is looking real or not. However, in order to avoid any confusion, this critic model is addressed as "the discriminator" in the rest of this document.

In this context, the discriminator is implemented to use the Wasserstein distance as the loss function. More details about this metric can be found in section 4.4.

Model definition

The first step to train Generative Adversarial Networks is to define the generator and discriminator models. The way these models are designed for the implementation of this thesis slightly differs in various aspect from the way they are commonly defined. The main reason is that most GAN implementations are designed to generate images. Secondly, the loss function used is the Wasserstein distance. The remaining aspects are the incorporation of the ability to generate data based on certain conditions, and the ability to generate categorical data. However, a premise on this implementation is to define the simplest neural network architectures considering that the GAN framework is complex enough.

Both the generator and the discriminator are defined as feed-forward networks with 1 to 5 hidden and fully connected layers, with the number of layers defined as a parameter for the experiments. The output layer for the discriminator is simply a fully connected layer with just one unit. On the other side, the output layer of the generator varies depending on four scenarios of two variables: if the model has a conditional vector, and if the data to generate is categorical or continuous (at this point, free text data is encoded as categorical data). Hence, 4 generator models were defined. If the model is meant to generate continuous data, the output is a fully connected layer with a *Sigmoid* activation and N units (the data dimension size). If the model is meant to generate continuous data, the setup is the same but the activation function has to be *Softmax*. For models with conditional data, there is an additional layer that concatenates the conditional vector and the prediction, delivering this concatenation as the output. For a list of additional parameter and variables involved in the definition and training of these models, please refer to table 4.1.

Variable	Value
Neural Network architecture	Feed-forward networks
Hidden layers	{1,3,5} fully connected
Number of neurons	$batchsize * 2^{layernumber}$
Activation function	ReLu
Loss function	Wasserstein distance
Optimizer	Adam optimizer
Batch size	128
Learning rate	5e-5

Table 4.1: Additional variables and parameters related to the definition and training of the GAN framework. Any parameter not listed in this table or mentioned before, was left as default [7].

Adversarial Training

Most of the code implemented for training these models is based on a GAN-sandbox repository available online [9]. This repository contains various popular GAN frameworks implemented in Python language with the Keras and Tensorflow libraries. All these implementations are

heavily based on the underlying theory proposed in their corresponding papers.

Basically, once the generator and the discriminator models are defined, the training of a Generative Adversarial Network aims to find a distribution $\mathbb{P}_\theta$ that is as similar as possible to the real input, an introduced data distribution $\mathbb{P}_r$. Therefore, the idea is to train a function $g_\theta(Z)$, where Z is a random variable, so that its distribution $\mathbb{P}_\theta$ finally matches $\mathbb{P}_r$.

In a Wasserstein GAN, the training algorithm as it was originally proposed by *Arjovsky et al.*, consist of training the "critic" (discriminator) to compute the best estimate for the loss function $W(\mathbb{P}_r, \mathbb{P}_\theta)$, perform back-propagation to update the θ gradient and hence g_θ, to make g_θ even more similar to $\mathbb{P}_r$. This process is repeated until θ converges. The training algorithm of the original paper [1] can be seen in figure 4.5. For this research, each model is trained for a total of 10.000 iterations, and the algorithm is designed for saving the weights and the losses (of both generator and discriminator) every 100 steps.

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.
while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Figure 4.5: The Wasserstein GAN adversarial training algorithm as it is proposed in its original paper by *Arjovsky et al.* [1].

4.2.2 A baseline synthetic data generator

An alternative approach that does not make use of Generative Adversarial Networks is implemented for comparison purposes. This approach is a simpler method to generate data, and it has two components: a basic probabilistic function to generate continuous and discrete data, and a Recurrent Neural Network to generate free-text data.

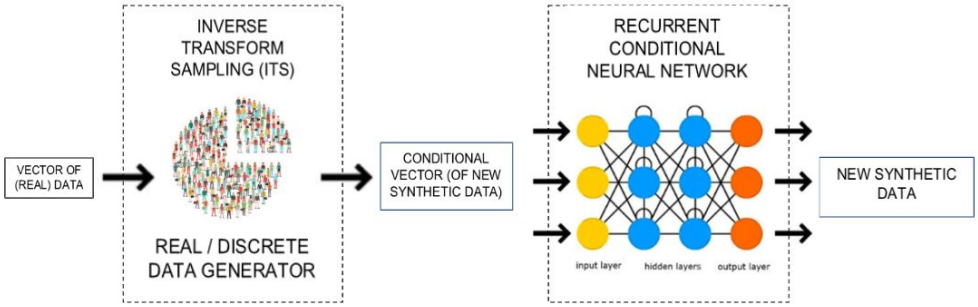


Figure 4.6: The alternative data generator is implemented with two components. The first component is a probabilistic Inverse Transform Sampling (ITS) function which receives real data (continuous or categorical) and outputs new synthetic data (continuous or categorical). The second component is a Recurrent Neural Network that receives the data generated by the ITS function and outputs new free-text data when needed.

Inverse Transform Sampling

The first component is based on the Inverse Transform Sampling (ITS), a basic probabilistic method to sample pseudo-random numbers from any statistical distribution. This method is also called the *Smirnov transform* and basically computes the cumulative distribution function (CDF) of the the input data, inverts that function, and generates random samples following its distribution. More details about this function and how to implement it can be found in [35]. By using this method there is no need to create a model, and there is no need for a training phase. Therefore, it already has some advantages over the GAN-based generator as the new data can be generated on the fly.

On the other side, such a simple approach would not be able to preserve the correlation patterns between the data attributes as a Gener-

ative Adversarial Network would. Therefore, an additional variant of the ITS method was implemented in order to carry out a fair comparison. This variant is defined as *Cross-relational Inverse Transform Sampling* (CR-ITS) and basically consist of sampling a vector with all the attributes instead of just a single attribute at a time.

Recurrent Conditional Neural Network

The only purpose of this second component is to serve as a text generator. However, it is expected that the GAN-based approach with the use of a Conditional Wasserstein GAN will generate synthetic text data preserving certain patterns depending on the conditional attributes (e.g. the first name of a person with the attribute *Gender* equals to *Female* should be a text similar to a woman's name). And a simple Recurrent Neural Network would not be a good match for a fair comparison, as it would not necessarily be influenced by the rest of the data attributes. Therefore, the proposed component is an RNN enhanced with a conditional vector that contains the attribute(s) that "influence" the target text data that will be generated.

The defined neural network is more specifically a *Gated Recurrent Unit* RNN, that receives a two-dimensional structure containing the conditional vector, and a pre-defined number (the window size) of one-hot-encoded structures corresponding to the previous letters of the text to be generated. The output of the neural network is a dictionary-vector of probabilities for the next letter in the text. For a list of parameter and variables involved in the definition and training of this model please refer to table 4.2. For more information about Gated Recurrent Unit neural networks, please refer to the original paper of *Chung et al.* [6].

4.3 Experiments

The research question that this thesis intends to answer is: *Can Generative Adversarial Networks be used to generate synthetic continuous, discrete and text data, effectively and efficiently, while preserving the underlying distribution and patterns of the real data?*.

With the research question in mind, the experimental part of this study is composed by 99 experiment variants (see table 4.3). Each experi-

Variable	Value
Neural network architecture	Gated Recurrent Unit (GRU)
Hidden layers	1 fully connected
Number of neurons	Size of dictionary (128 aprox.)
Activation function	Softmax
Loss function	Categorical cross-entropy
Optimizer	Adam optimizer
Batch size	128
Window size	3 letters

Table 4.2: Variables and parameters related to the definition and training of the Recurrent Conditional Neural Network. Any parameter not listed in this table was left as default [30].

ment consist in the execution of the synthetic data generation process, with a specific combination of parameters (variant), in order to evaluate the following aspects: the efficiency, the preservation of the underlying data distribution, and the preservation of patterns in continuous, discrete and text data.

4.3.1 Test system

All experiments, with one exception, were performed on a cloud computer cluster with the same hardware specifications (table 4.4). In the case of the experiments for the GAN variant with 5 neural-network-hidden layers, due to the low performance of this cluster (in terms of the time to train the neural network), the test system used was a more computationally powerful cloud cluster (table 4.5).

4.3.2 Software

All the software implemented is written in Python 2.7 on Apache Spark 2.3.0 with the Python libraries listed in table 4.6.

4.4 Evaluation

To begin with, the efficiency of the data generators is measured with the training time and data generation time in seconds. Then, the pattern preservation in the new data is evaluated by calculating the Eu-

Approach	Variant	Model selection (Step)	Size (of data)
GAN	1 Layer	Last step	2X
	3 Layers	Best generator	10X
	5 Layers	Best discriminator	100X
Non-GAN	ITS + RNN	Last step	2X
	CR-ITS + RNN		10X
			100X

Table 4.3: All the experiment variants. For the GAN-based generator, 9 different models are tested, i.e. 1, 3 and 5-layered models selecting the last, best-generator and best-discriminator weights. For the non-GAN-based generator, 2 different models are tested. One using simple Inverse Transform Sampling, and another one using Cross-Relational Inverse Transform Sampling, both with a Recurrent Conditional Neural Network for text generation. All experiment variants are executed to generate 2 times, 10 times, and 100 times the size of the training data, over the 3 case studies to sum up a total of 99 experiments.

clidean distances between the correlation matrices of the new data and the real data. The preservation of the underlying data distribution is assessed with the *Wasserstein* metric. And finally, to evaluate the quality of the new free-text data, the *Perplexity* metric is used.

4.4.1 Overall metrics

The metrics considered to test the overall performance are model-training time in seconds, data-generation time in seconds, the number of duplicated records generated, and the number of records of the generated dataset that are repeated records from the original dataset. From these, the model training times and the data generation times are considered to analyze the efficiency of the data generation approach. The number of duplicated and repeated records are additional overall metrics that are useful for the final analysis of the results.

4.4.2 Correlation (Euclidean) distance

Having calculated the correlation matrices withing the attributes of the original dataset and the attributes of the generated dataset, a suitable way to measure the similarity of these is to calculate the sum of their

Specification	Value
Cloud platform	Databricks Community
Processor	0.88 Cores, 1 DBU (1 Driver + 1 Worker)
Memory size	6.00 GB
GPU	Not available
Languages	Python 2.7 on Apache Spark 2.3.0

Table 4.4: Specifications of the test system used to train all the models and run all the experiments with the expection of the corresponding to the GAN model with 5 neural-network hidden layers.

Specification	Value
Cloud platform	Aure Databricks
Processor	16 Cores, 3 DBU (1 Driver + 1 Worker)
Memory size	56.00 GB
GPU	Not available
Languages	Python 2.7 on Apache Spark 2.3.0

Table 4.5: Specifications of the test system used to train and run the experiments corresponding to the GAN model variant with 5 neural-network hidden layers.

pair-wise euclidean distances -i.e. the sum of the euclidean distances of every X_{ij} and Y_{ij} of the correlations matrices X and Y (see figure 4.7).

These results are a suitable way to measure the preservation of the intrinsic patterns occurring between the attributes of the original dataset in the new synthetic dataset. The lower this metric is, the better the data generation tool preserves the patterns.

$$d(R, F) = \sum_{i=0}^{n} \sum_{j=i}^{n} \sqrt{(R_{ij} - F_{ij})^2}$$

Figure 4.7: The sum of the pairwise Euclidean distances of the correlation matrices R *(real data)* and F *(fake data)*.

Package name	Version	Purpose
Tensorflow	1.3.0	Machine Learning framework
Keras	2.0.8	High level neural networks API
Scipy	1.0.0	Statistical functions
Scikit-learn	0.19.1	Data analysis
Pandas	0.22.0	Data structures
Numpy	1.14.2	Data structures
Matplotlib	2.2.2	Data visualizations

Table 4.6: All the Python software-packages used in the implementation and analysis of the solution and the experiments.

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)} \gamma[\|x - y\|]$$

Figure 4.8: The *Earth Mover's distance* is the "cost" of the optimal transport plan. In the equation above, $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. The expression $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$ [1].

4.4.3 Wasserstein distance

The Wasserstein distance (see figure 4.8) is a proper metric to measure the difference between two probability distributions. Therefore, it provides a suitable way to see how different is the underlying data distribution of a new synthetic dataset compared with the distribution of a set of real data. Intuitively, it can be interpreted as the minimum cost of turning one pile of dirt into the another pile of dirt (where the cost is the amount of dirt moved times the distance). The lower this cost is, the more similar the piles of dirt (data distributions) are. This metric is only applied to continuous an categorical data.

4.4.4 Perplexity

The text perplexity (figure 4.9) is usually used to measure the quality of the text results of a machine learning model. Intuitively, it measures how surprised (or perplexed) the model was to see the output.

$$P(x_i, y_i) = e^{l(x_i, y_i)}$$

Figure 4.9: The perplexity metric. If the cross-entropy loss for an input x_i and its corresponding output y_i is $l(x_i, y_i)$ then the expression above would be the *Perplexity* [16].

The lower the perplexity, the more similar is the generated text to the original text dataset [16].

Chapter 5

Analysis and results

This section presents in detail the analysis and the results of the experiments. To begin with, the overall aspects and overall metric results are presented and analyzed. Then, for each one of the specific aspects measured –i.e. the efficiency, the preservation of the data distribution, the preservation of the correlation patterns, and the generation of quality text–, there is a section presenting the corresponding analysis and results. For a more detailed table of results, please refer to the appendix.

5.1 Overall analysis and results

The table 5.1 depicts the amount of data generated in the experiments for every one of the three case studies. It is important to point out that the original data size stated for the cases D1 and D2 is the size of their training dataset after randomly sampling from the original dataset size. For the case of D3, the dataset size was considerable small, therefore all the data was used for training.

Dataset	Original	2X	10X	100X
D1	1.000	2.000	10.000	100.000
D2	1.000	2.000	10.000	100.000
D3	199	398	1.990	19.900

Table 5.1: Size of the 3 original datasets and the number of records generated for the experiments: 2 times (2X), 10 times (10X), and 100 times (100X) the original size.

From the overall metrics gathered, it was noteworthy to verify the number of duplicated synthetic records generated, as well as the number of synthetic records that were repeated in the real datasets. In the case studies of D1 and D2, there were no duplicated or repeated records in any of the experiments. This suggests that the models did a good job generating new samples of not duplicated data, and without simply copying records from the input. Regarding the case study D3, as can be seen in table 5.2, all experiments generated a very small amount of duplicated records. Nevertheless, the results are substantially better than expected since the original dataset is considerably small, with only one attribute of continuous values, and a fairly small number of categorical variable. In addition, the number of repeated values from the original dataset was zero in all variants. All the detailed results of all the metrics and all the case studies can be found in the Appendix section.

Variant	2X	%	10X	%	100X	%
GAN-1L LAST	0	0,0%	11	0,6%	1.363	6,8%
GAN-1L GEN	1	0,3%	25	1,3%	2.380	12,0%
GAN-1L DISC	17	4,3%	184	9,2%	1.997	10,0%
GAN-3L LAST	0	0,0%	9	0,5%	1.419	7,1%
GAN-3L GEN	1	0,3%	31	1,6%	1.970	9,9%
GAN-3L DISC	0	0,0%	2	0,1%	257	1,3%
GAN-5L LAST	0	0,0%	6	0,3%	858	4,3%
ITS LAST	1	0,3%	10	0,5%	787	4,0%
CR-ITS LAST	0	0,0%	4	0,2%	906	4,6%

Table 5.2: Number and percentage of duplicated records (based on the size of the original dataset) in the generated data for all the experiment variants with dataset D3.

5.2 Efficiency

The efficiency of each variant is analyzed by measuring the time it takes to train or create any required machine learning or statistical model, and the time it takes to generate a specified number of records.

5.2.1 Training time

All the models created in this research were trained for 10.000 iterations. The training times in minutes were registered every 500 steps and are reflected in figure 5.1 for the 3 case studies.

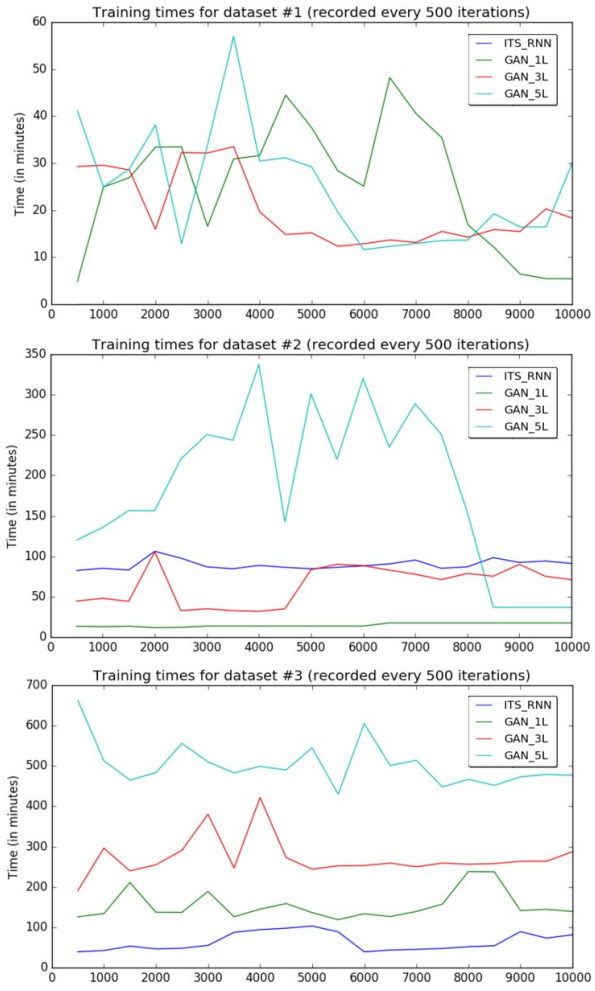


Figure 5.1: Training times (in minutes) of 10.000 iterations, recorded every 500 iterations, for the GAN and the RNN models. For the first dataset, there are no free-text attributes involved. There is no machine learning model required for the alternative approach, and therefore, the purple line does not show in the first plot (top of the figure).

In the first time-line graph, the ITS-RNN approach does not participate since there was no necessity to train a text generation model for a dataset without free text data. In the remaining figures, it is clear

how computationally expensive it is to train these neural networks with hardware resources not specialized for it (i.e. without a Graphics Processing Unit). It was also expected that in the case of the GAN approach, the training times increased with the addition of more hidden layers to the models. However, it was not expected that these training times were that high (reaching almost 700 minutes for 500 iterations) with a training set of 200 records.

The training times for the RNN models were considerably lower, and it is important to point out that the comparison between the GAN and non-GAN approaches is only fair in the third graph, for a dataset composed mainly by text attributes.

5.2.2 Data generation time

The data generation times for all experiments were registered in seconds without considering any pre-processing or post-processing of the data in the generation pipeline. These results are shown in figure 5.2 for the three case studies (left to right) and for the generation of two times (2X), ten times (10X), and 100 times (100X) the size of the original datasets.

As it was expected, the data generation times are almost insignificant for the ITS variants (without using neural network models), as it is shown in the graphs corresponding to the first and second case studies. In the third case study, although when generating two times the amount of the original data the results were as expected, it is clear that as the number of records increased the performance of the RNN-based variants was considerably less efficient than the GAN-based variants. However, the fact that on the first experiments (2X) the RNNs were faster than the GANs, could suggest that there might be an overhead in the data generation algorithm of the ITS-RNN approach that could be causing these poor results. More research and work is bound to be done in order to improve the efficiency of the algorithm in charge of assembling the new synthetic data in a structured way.

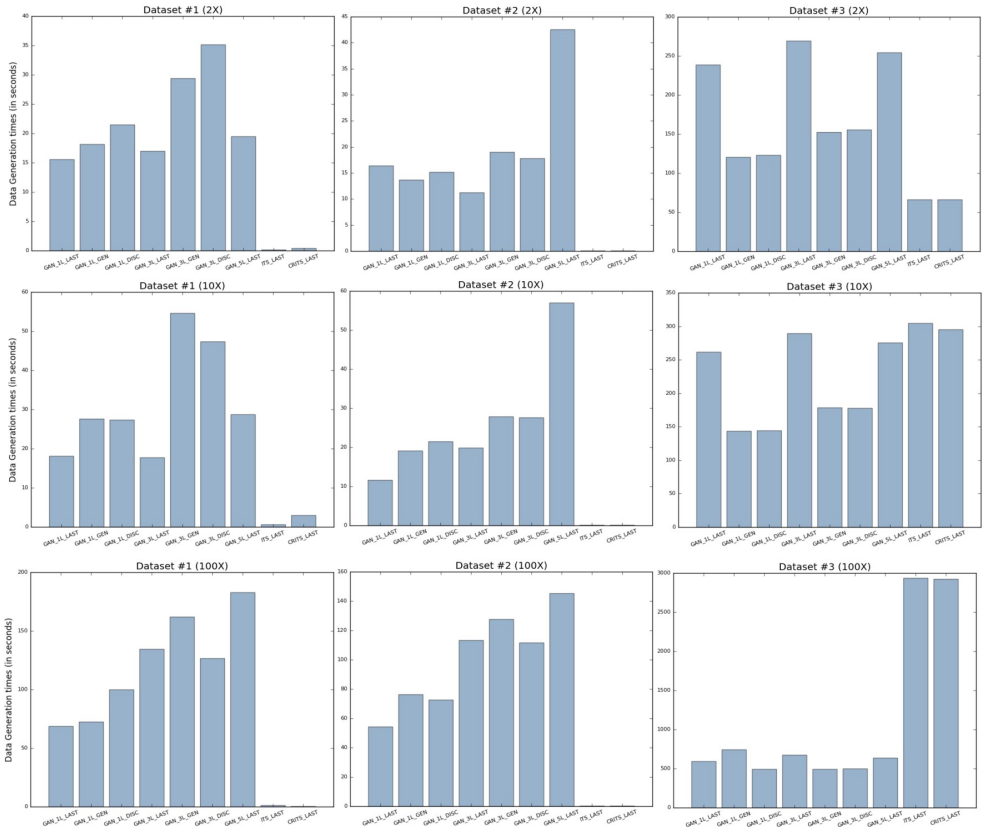


Figure 5.2: Average data generation times (in seconds). Each bar correspond to an experiment variant and the required data generation time. A larger version of this figure can be found in the appendix.

5.3 Preserving the data distribution

When it comes to preserving the data distribution of the original datasets, one can argue that the results obtained leave almost no room for discussion. As can be seen in figure 5.3, the ITS-based data generators are both faster and better at creating synthetic data with a similar distribution of a real set.

All the figures show the ITS and CR-ITS generators having the lowest average Wasserstein distances of their attributes with respect to the original data. Moreover, all the figures above with the exception of the ones corresponding to the dataset #3 for 10X and 100X times the training data size (the last two of the bottom-right side), demonstrate these simpler generators to be considerably faster. The long process-

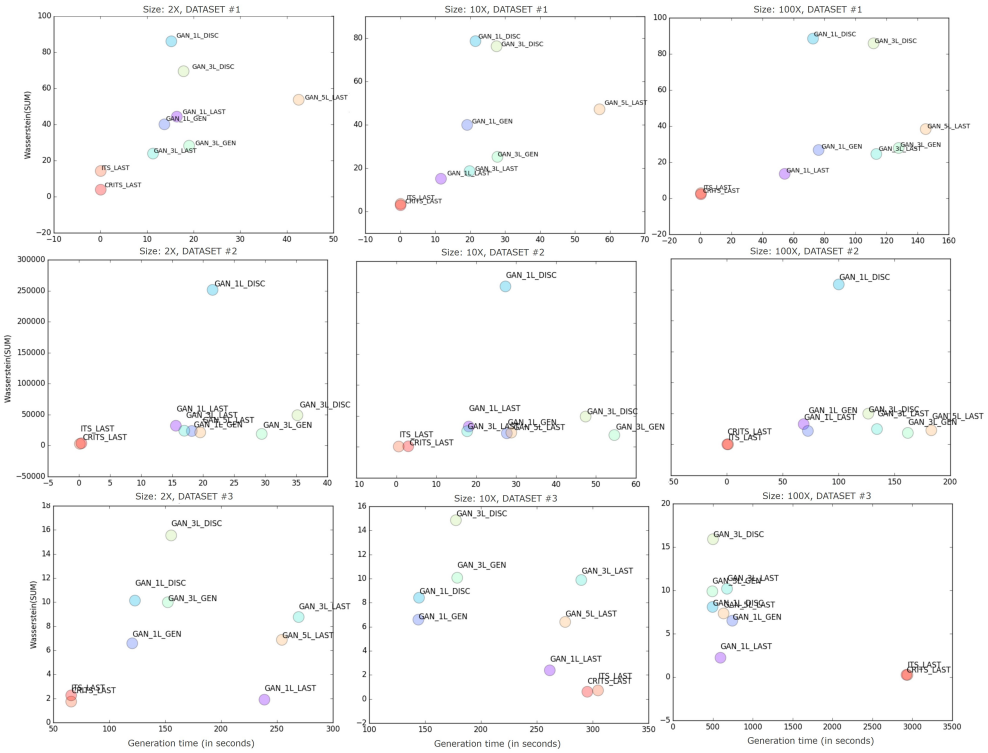


Figure 5.3: Wasserstein distances and data generation times (in seconds). The best models in terms of preserving the data distribution of their original datasets and being efficient (data generation time), are the ones shown at the bottom left corner of each plot. A larger version of this figure can be found in the appendix.

ing times of these two exceptions are due to the free-text generation components, the Recurrent Neural Networks.

In summary, without even considering the long training times of the neural-network approaches, it was more effective to generate categorical and continuous data without deep learning. An example of the results obtained for one of the most complex categorical attributes of the three case studies is presented in figure 5.4.

5.4 Preserving the correlation patterns

The evaluation results for the preservation of the correlation patterns of the original datasets can be seen in figure 5.5. The initial expecta-

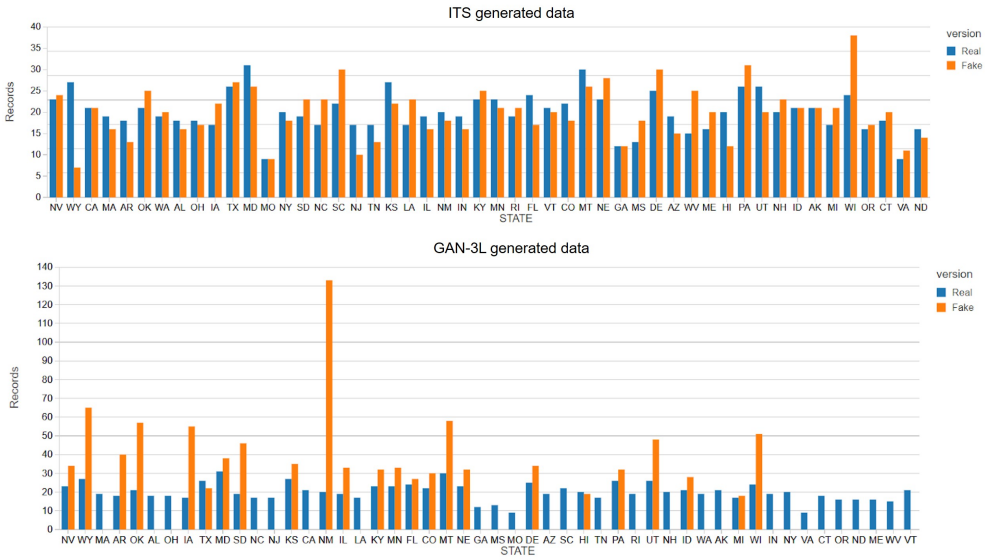


Figure 5.4: Example of how ITS-generated data preserves better the data distribution of a categorical attribute like the state of the United States (with 50 possible values) in dataset no.2, while the best 3-layered GAN model shows poor results.

tions were that the GAN-based generators were going to outperform in all cases the non-GAN approaches. Although that was the case for all the experiments with the exception of the first dataset, these results should be analyzed per case-study in separate.

The first case study has the particularity of being highly engineered for machine learning experiments. It contains only three attributes (two categorical and one continuous) that presents some weakly notorious correlations with the rest of the attributes. As it can be verified in figure 5.6, the results do not show signs of preservation of the correlation patterns of the real data, either in the best performing ITS-based generator or in the best GAN-based generator. Nevertheless, the correlation distances between the ITS-generated data and the real data was considerably smaller. These results are obviously not conclusive.

In contrast, the results of dataset 2 (figure 5.7) were as expected. The best GAN-based model clearly shows similar correlation patterns as the ones in the real dataset. It is noticeable how the stronger patters like the obvious relation between the average call and the total call

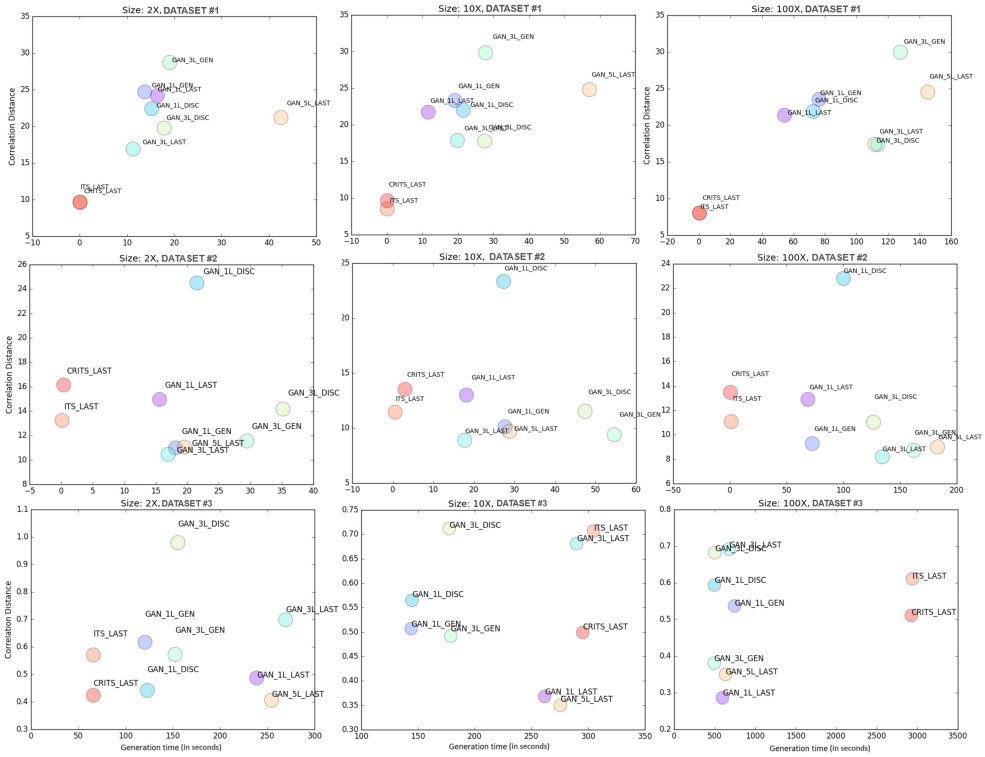


Figure 5.5: Correlation distance and data generation times (in seconds). The best models in terms of preserving the correlation patterns of their original datasets and being efficient (data generation time), are the ones shown at the bottom left corner of each plot. A larger version of this figure can be found in the appendix.

duration, or the call-drop rate and the gender, are well preserved.
For the real-world scenario of dataset number 3, although with a small number of categorical and continuous values to compare, the results were significantly better in terms of lower correlation distances for the GAN-based data generators. These can be seen in figure 5.8.

All in all, Generative Adversarial Networks showed a better preservation of the correlation patterns. On the other side, the attempt to create a simple data generator for this goal, i.e. the cross-relational inverse transform sampling approach, were rather disappointing.

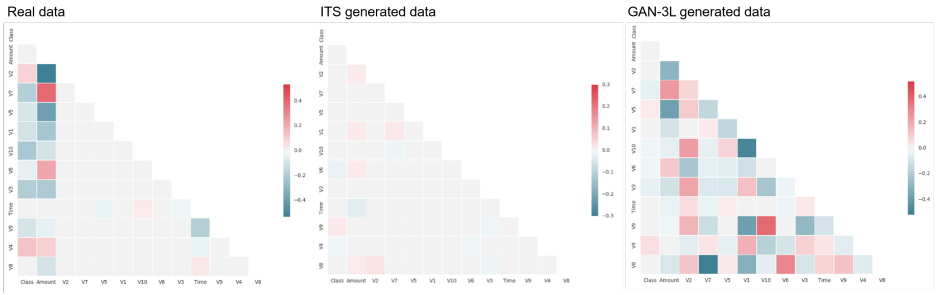


Figure 5.6: Correlation matrix comparison of the real data (left) with the best model of the GAN approach (right), and the best model of the non-GAN approach (center) in terms of preserving the data correlations of the original dataset number 1. A larger version of this figure can be found in the appendix.

5.5 Generating quality text

The generation and evaluation of synthetic free-text data is arguably the most complex task carried out in this thesis. The only case study containing several free-text attributes was the third case study, with real-world-scenario text data.

The scatter plot in figure 5.9 shows how the GAN-based data generator outperformed the RNN text generator in terms of perplexity. Following an intuitive reasoning, the GAN-based models were less perplexed of the outcome after having seen the original text data.

The mentioned figure only shows the evaluation results for the GAN-based models with the purpose of making the best performing variants clearly visible. In all cases, the GAN neural network with 5 hidden layers outperformed the rest. These results are in line with the initial expectations that more complex generator networks should be able to return better results (in the same number of iterations) as shallower generators. The results and the perplexity metrics of the RNN text generator were not as expected and were rather disappointing. These results and their significant difference with the GAN-based generators results can be seen in table 5.3.

After analyzing these results along with the efficiency measurements, it is clear from a quantitative point of view that GAN-based gener-

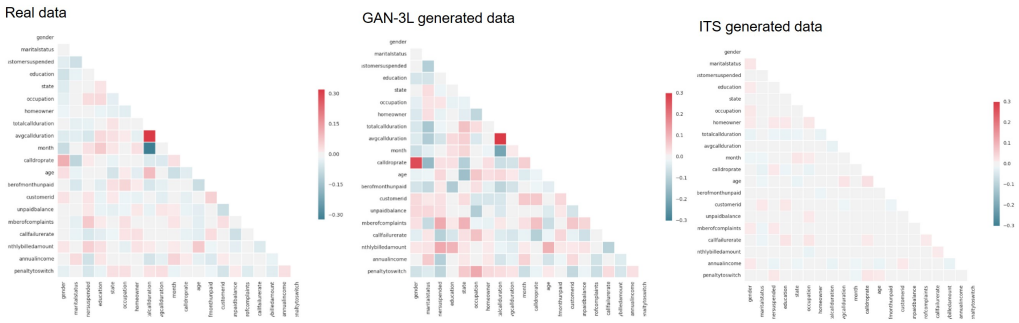


Figure 5.7: Correlation matrix comparison of the real data (left) with the best model of the GAN approach (center), and the best model of the non-GAN approach (right) in terms of preserving the data correlations of the original dataset number 2. A larger version of this figure can be found in the appendix.

ators created better synthetic text-data in a more efficient way. And although the qualitative comparisons can be quite subjective, the sample in figure 4.9 arguably supports that the these were also better from a qualitative perspective.

It is important to point out that since the dataset with free-text data contains real-world information from customers of a private company, the samples shown are rather sensitive to privacy issues. Therefore, the figures presented in this section are limited to one of the less sensitive attributes of the dataset such as street names. Nevertheless, the result patterns are similar for the remaining text variables. While the RNN-generated text usually presents unwanted patterns like repeated text strings, the GAN-generated text show promising results for the proposed approach, considering that there is still room for improvement with some additional ideas for a future line of work.

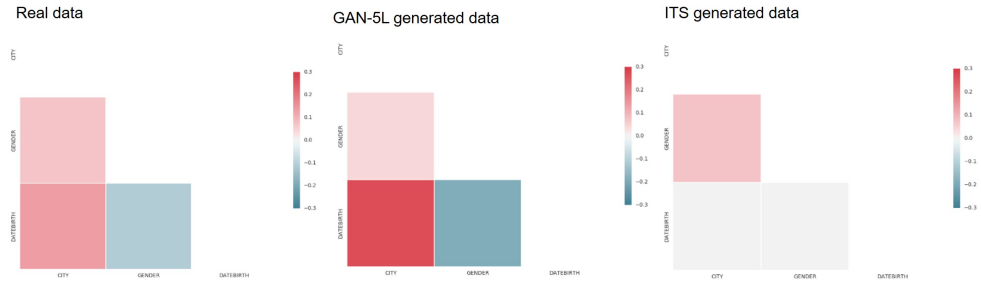


Figure 5.8: Correlation matrix comparison of the real data (left) with the best model of the GAN approach (center), and the best model of the non-GAN approach (right) in terms of preserving the data correlations of the original dataset number 3. A larger version of this figure can be found in the appendix.

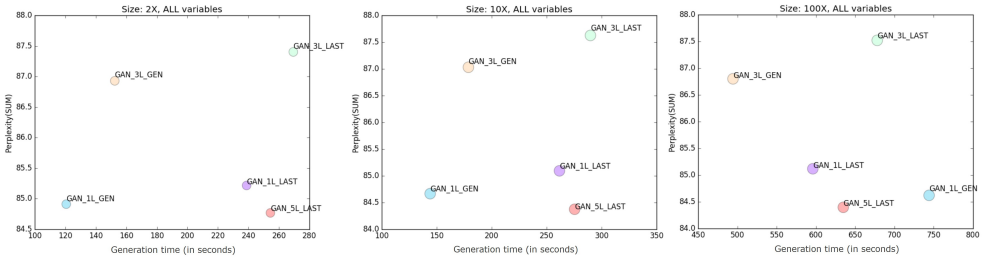


Figure 5.9: Perplexity metric and data generation times (in seconds). The best models in terms of generating quality text –i.e. text that is similar to the text in the original dataset– and being efficient (data generation time), are the ones shown at the bottom left corner of each plot. A larger version of this figure can be found in the appendix.

Variant	2X	10X	100X
GAN-1L LAST	14,20	14,18	14,19
GAN-1L GEN	14,15	14,11	14,10
GAN-1L DISC	30,69	30,64	30,63
GAN-3L LAST	14,57	14,60	14,59
GAN-3L GEN	14,49	14,51	14,47
GAN-3L DISC	22,54	22,50	22,49
GAN-5L LAST	14,13	14,06	14,07
ITS LAST	49,86	49,57	49,65
CR-ITS LAST	49,21	48,85	49,06

Table 5.3: Average Perplexity results for the synthetic data evaluated generation of 2X, 10X and 100X the original size of the dataset 3.

Real data

STREET
Arkens vag 16 lgh 1402
Carlsminnevagen 10 A
Slanvagen 2
Jan Eriks vag 7
Hesselmans Vag 16
Sagavagen 76
Katarina Vastra Kyrkogata 6 A
Milstensvagen 34
Topeliusgatan 18 lgh 1101
Prastgatan 9 F
Sagavagen 42
Rollekevagen 10
Raggardevagen 1
Kvartarvagen 173
Ringgatan 39 B 1tr
Hagalundsgatan 32 V
Munters Gata 9
Stortorpshojden 6

GAN-5L generated data

STREET
Forels ergsvagen 30
Sorensbergsvagen 1 lgh 1l01
Minkepepvagen 20l l
Bellavbsresan 1 lgh 110l
Rittarvagen 14
Artslgb rts ag e0a
Murstplsgatan 90
Skolgatan 33 6
Bellmansgatan 152 lgh 1302
Brottsiigen 6
Stigdavggen 7 B l h 13R3
Kotelibergsvagen 30
Sagelmbergsvagen 1
Altsdpavag 23 B t h 13
Lnrklevagsn 3vgh a4 4 h r
Torsgatan 2 sgh e003
Fltnlman en 6lB lgh 13al
Saeavagen 16

RNN generated data

STREET
Storgatan 100331013033Cs fregata
Kluorgstuguvagen 30 1302020202 l
Os40 llansjovasjovagsfitntevagen
Honnsga Cbyvedrgatag 3 110131140
Davavagesvogen 113s4odvagen 113s
Fmrings vatans vag 120Ekdnarag 1
Ntrajlan 104Vitben 1 B Fgatan 2
Kunbegatagen 40263A lgbtopinborg
Svagath 11 lgh 110 11 lgh 110 Hv
Bedkundvagen 28 lgh 1001002esata
Myrenggatan 39 B 1tran 39 Bedids
Sturegatgrgatanra 19kunermmrgata
Klagen 359 43 13020903020202020
Kna 14ellnn 1234 lgn 1402C 14023
Bjoangsvagen 15 Rjagen 15 Rjnloy
Hike29 lgnvagen 18 18 lgnvaeljor
A lgh 10101010101010101010101001
Fradsvradsvag Bran 202 Borgh 120

Figure 5.10: Example of how a GAN-based multilayer perceptron model can generate text with more quality (more similar to the original text), than a non-GAN approach based on a Recurrent Neural Network model.

Chapter 6

Conclusions and future work

The goal of this project is to aid the development of a synthetic data generator in a way that a future product requires minimal user interactions while generates quality synthetic data preserving similar patterns and a similar statistical distribution of a real dataset. Most of the existing tools and approaches to generate synthetic data often require a great deal of user interactions or simply are not focused on replicating the patterns of an initial real dataset. Besides, most of the related studies do not use new machine learning technologies or do not address the generation of synthetic data of diverse types. Generative Adversarial Networks can be used to unravel these limitations. Therefore, the research question of this project is: *Can Generative Adversarial Networks be used to generate synthetic continuous, discrete and text data, effectively and efficiently, while preserving the underlying distribution and patterns of the real data?*

A data generation pipeline with 6 sequential steps (excluding the data input/output steps) was designed and proposed to generate continuous, categorical, and free text data. The first step detects the data types and the schema in the dataset. Step 2 performs a pattern analysis detecting and saving the data correlations within the dataset. Next, in step 3 the data is engineered and transformed so that it can be analyzed and processed more efficiently by the required statistical and machine learning models. The model creation and training, when required, takes place in step 4. Then, in step 5 is where the data generation is executed using the models created in the previous phase. And to finalize the process, in step 6 the new data is reverse-engineered

so that it ends up with the same format and schema of the original dataset.

Based on this data generation pipeline, a Wasserstein Conditional Generative Adversarial Network was designed and implemented with the simplest possible setup for the generator and discriminator models. This framework is the main proposal of this thesis. In addition to this GAN-based approach, a simpler non-GAN based framework was also proposed to serve as a baseline for comparisons. This simpler approach used Inverse Transform Sampling to generate continuous and categorical data. This is a basic probabilistic method to sample pseudo-random numbers from any statistical distribution. To generate free text data, a Recurrent Conditional Neural Network generator is used to complement this alternative framework.

Several experiments were carried out from 3 different case studies (3 different datasets) to generate 2 times (2X), 10 times (10X) and 100 times (100X) the original size of each dataset. Every experiment was executed with different variants of the proposed GAN-based generator and different variants of the alternative generator. The results were evaluated with specific metrics measuring the efficiency, the preservation of the original data distribution, the preservation of the original data correlations, and the generation of quality text data similar to the free-text attributes in the original dataset.

As an overall conclusion, the results show that indeed Generative Adversarial Networks can be used to generate continuous, discrete and text data, while preserving the underlying distribution and patterns of the real data. The GAN-based proposal outperformed the baseline approach when it comes to preserve the correlation patterns of the original data. Moreover, promising and interesting results were obtained from the generation of quality free-text data. Nevertheless, the alternative approach was clearly better in terms of efficiency, and in the preservation of the statistical distribution of the initial data. A summary of the results is presented in tables 6.1, 6.2, 6.3, and 6.4.

Evaluation	Metric	Size	Best model	Value
Efficiency	Data generation time in seconds	2X	ITS	0,07
		10X	ITS	0,12
		100X	ITS	0,38
Data distribution	Wasserstein distance	2X	CR-ITS	0,297
		10X	CR-ITS	0,22
		100X	CR-ITS	0,168
Data correlations	Euclidean distance of correlation matrices	2X	CR-ITS	9,590
		10X	ITS	8,52
		100X	ITS	8,01

Table 6.1: Best models for each evaluation criteria over dataset 1

Evaluation	Metric	Size	Best model	Value
Efficiency	Data generation time (in seconds)	2X	ITS	0,16
		10X	ITS	0,6
		100X	CR-ITS	0,49
Data distribution	Wasserstein distance	2X	ITS	143,2
		10X	ITS	32,3
		100X	CR-ITS	11,28
Data correlations	Euclidean distance of correlation matrices	2X	GAN-3L	10,46
		10X	GAN-3L	8,89
		100X	GAN-3L	8,2

Table 6.2: Best models for each evaluation criteria over dataset 2

6.1 Discussion

Despite the promising results obtained with the use of Generative Adversarial Networks or the use of Inverse Transform Sampling, there are various drawbacks, challenges, and aspect that could have been done in a different way which are worth to be mentioned in this section.

The conclusion of this thesis is that it is indeed possible to generate continuous, discrete, and text synthetic data using Generative Adversarial Networks. However, one can start a discussion over what makes the difference between a continuous value, and a discrete value, or a discrete value and free text data. The boundaries between these categories are not set in stone, and in the implementation of the proposed

Evaluation	Metric	Size	Best model	Value
Efficiency	Data generation time in seconds	2X	ITS	66
		10X	GAN-1L	144
		100X	GAN-3L	494,4
Data distribution	Wasserstein distance	2X	ITS	0,87
		10X	CR-ITS	0,3
		100X	ITS	0,13
Data correlations	Euclidean distance of correlation matrices	2X	GAN-5L	0,46
		10X	GAN-5L	0,35
		100X	GAN-1L	0,286
Text data quality	Perplexity	2X	GAN-5L	14,12
		10X	GAN-5L	14,06
		100X	GAN-5L	14,06

Table 6.3: Best models for each evaluation criteria over dataset 3

data generation pipeline, these boundaries were a design decision. For example, what makes the dataset field *City Name* a discrete attribute (after a proper encoding) and not free-text data, is a fixed threshold parameter. This parameter is the percentage of unique values of the attribute (e.g. *City Name*) in the dataset. Naturally, as the number of unique values gets closer to 100%, the attribute should be considered as free-text data. And obviously, it is considerably less expensive to generate categorical data than free-text data. A fact that is also supported by the experiment results in this thesis. Nevertheless, this is a parameter that can be left to the user to control it, disregarding that one of the goals is to minimize user interaction with the automatic schema detection.

Another design decision that can impact the performance and the results of the data generation pipeline, is the order in which the data attributes are generated. The pattern detection phase of the proposed generation process states that after determining (quantitatively) the influence of each attribute, the dataset columns are sorted according to how influential they are. The order of precedence can be from most influential to less influential, or the other way around. This approach may not be the most optimal since it could affect negatively the training and processing times if an expensive attribute is processed at the beginning of the cue (which also will be used then as a conditional

Evaluation criteria	GAN approach	Alternative approach
Efficiency	More efficient for free-text data	More efficient for categorical and continuous data
Preservation of correlation patterns	Always better	More research is required
Preservation of data distribution	More iterations are required for a conclusion	Effective and efficient method
Quality of free-text data	Great results	Disappointing results

Table 6.4: Summary of conclusions per evaluation criteria

vector to generate the following attributes).

The points of discussion mentioned above are different variants to experiment which were let out of the scope of this work. Considering the limited hardware resources available for this thesis, many other variables were let out, and a just a small set of hyper-parameter combinations was included. It is also worth mentioning as a final discussion point, that a data generation software tool based on this framework would not be feasible with the required training times. Therefore, improving this aspect has to be one of the first lines of future work to consider.

6.2 Stakeholders feedback

This thesis was carried out with the support of BI Insight AB [3], a consulting company with more than 15 years of experience providing traditional data analytic services for companies in Stockholm, Sweden. Their core expertise is the implementation of Data Warehouse systems and Business Intelligence platforms to provide their customers a way to make their data useful and gain insights from it.

In recent years, given the development of new innovations in the Ma-

chine Learning and advanced data analytic fields, BI Insight AB has gain interest to provide solutions with these new technologies.

This thesis project has served as a first approach to develop a tool that can be useful when the required data is not accessible. Although the premise is to build a solution with the benefits that provide Machine Learning technologies like Generative Adversarial Networks, it is also important that the use of the tool is sustainable in terms of used resources –i.e. time and computational power. Therefore, as the solution provider, BI Insight AB considers that there is still room for improvement until the contribution of this thesis can be used for the development of a minimum viable software product.

6.3 Future work

The delimitation section of this thesis states that the outcome of the project is meant to be the results and conclusion of the analysis instead of a product. However, the implementation and the experiments carried out are a big step to a future development of a software tool.

The results obtained are promising either by the proposed GAN-based data generation tool, or by the baseline framework using the Inverse Transform Sampling method. Naturally there is a substantial room for improvement and multiple possibilities of new experiments. Therefore, this section presents some of the obvious ways that these frameworks can be improved, and some interesting research possibilities.

Further model training

To begin with, the models can be further trained for several additional iterations that much probably will improve the obtained results with a better preservation of the data distribution, a better preservation of the correlation patterns, and a better quality of the generated text data.

Distributed computations

One of the main disadvantages of using Generative Adversarial Networks for data generation is the computationally expensive training process. The model training phase of the proposed pipeline can be

improved without much complexity by using a Deep Learning distributed framework such as *dist-keras*[10]. Moreover, the use of proper hardware resources with Graphic Processing Units would make a considerable difference in the training times.

Experiments with a mixed framework

A synthetic data generation tool that takes advantage of the benefits of the two proposed approaches is feasible to implement. The inverse transform sampling method can be used in certain cases where there is no need to preserve data correlations. A new Generative Adversarial Network where the generator is a Recurrent Neural Network, is another possibility to experiment for free-text data generation. Likewise, a research to enhance the inverse transform sampling approach in order to preserve the correlation patters between the data attributes could give powerful and efficient results.

Other generative approaches

In the machine learning research community, the Generative Adversarial Networks are very often compared with Variational Autoencoders [23]. This is another generative approach proposed in 2013 by *Kingma et al.*, one year before *Ian Goodfellow* published the GANs paper. Therefore, an interesting research would be the comparison of GANs and Variational Autoencoders for synthetic data generation of continuous, discrete, and text data.

Another interesting possibility is to experiment with GAN variations that include the benefits and new features proposed by *Chen et al.* in the InfoGAN paper [4] or by *Donahue et al.* in *Adversarial Feature Learning* [11]. These implementations can be helpful to further improve the preservation of the intrinsic patterns in the synthetic datasets. All in all, every month new studies related to Generative Adversarial Networks are being published with improvements to the original framework. The possible future lines of work related to this thesis will continue to expand.

Bibliography

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein GAN". In: *arXiv:1701.07875 [cs, stat]* (Jan. 2017). arXiv: 1701.07875. URL: http://arxiv.org/abs/1701.07875 (visited on 02/11/2018).

[2] Marc G. Bellemare et al. "The Cramer Distance as a Solution to Biased Wasserstein Gradients". In: *arXiv:1705.10743 [cs, stat]* (May 2017). arXiv: 1705.10743. URL: http://arxiv.org/abs/1705.10743 (visited on 02/11/2018).

[3] *BIInsight*. en. URL: https://www.biinsight.se (visited on 10/06/2018).

[4] Xi Chen et al. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *arXiv:1606.03657 [cs, stat]* (June 2016). arXiv: 1606.03657. URL: http://arxiv.org/abs/1606.03657 (visited on 02/11/2018).

[5] Edward Choi et al. "Generating Multi-label Discrete Patient Records using Generative Adversarial Networks". In: *arXiv:1703.06490 [cs]* (Mar. 2017). arXiv: 1703.06490. URL: http://arxiv.org/abs/1703.06490 (visited on 04/04/2018).

[6] Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *arXiv:1412.3555 [cs]* (Dec. 2014). arXiv: 1412.3555. URL: http://arxiv.org/abs/1412.3555 (visited on 09/02/2018).

[7] *Core Layers - Keras Documentation*. URL: https://keras.io/layers/core/#dense (visited on 09/04/2018).

[8] *Credit Card Fraud Detection*. URL: https://www.kaggle.com/mlg-ulb/creditcardfraud (visited on 08/19/2018).

[9] Michael Dietz. *Vanilla GAN implemented on top of keras/tensorflow enabling rapid experimentation & research. Branches correspond to implementations of stable GAN variations (i.e. ACGan, InfoGAN) and other pro..* original-date: 2017-02-10T18:01:12Z. Aug. 2018. URL: https://github.com/mjdietzx/GAN-Sandbox (visited on 09/04/2018).

[10] *Distributed Deep Learning with dist-keras — Databricks Documentation*. URL: https://docs.databricks.com/applications/deep-learning/distributed-deep-learning/dist-keras.html (visited on 09/01/2018).

[11] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. "Adversarial Feature Learning". In: *arXiv:1605.09782 [cs, stat]* (May 2016). arXiv: 1605.09782. URL: http://arxiv.org/abs/1605.09782 (visited on 02/11/2018).

[12] Jeffrey L. Elman. "Finding structure in time". In: *COGNITIVE SCIENCE* 14.2 (1990), pp. 179–211.

[13] J. Eno and C. W. Thompson. "Generating Synthetic Data to Match Data Mining Patterns". In: *IEEE Internet Computing* 12.3 (May 2008), pp. 78–82. ISSN: 1089-7801. DOI: 10.1109/MIC.2008.55.

[14] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. "Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs". In: *arXiv:1706.02633 [cs, stat]* (June 2017). arXiv: 1706.02633. URL: http://arxiv.org/abs/1706.02633 (visited on 02/11/2018).

[15] Pedro Ferreira. *Towards data set augmentation with GANs*. Oct. 2017. URL: https://medium.com/jungle-book/towards-data-set-augmentation-with-gans-9dd64e9628e6 (visited on 05/08/2018).

[16] Thushan Ganegedara. *Natural Language Processing with TensorFlow*. en. Packt Publishing, May 2018. ISBN: 978-1-78847-831-1. URL: https://www.safaribooksonline.com/library/view/natural-language-processing/9781788478311/ (visited on 08/23/2018).

[17] *General Data Protection Regulation (GDPR) – Final text neatly arranged*. en-US. URL: https://gdpr-info.eu/ (visited on 02/12/2018).

[18] Ian J. Goodfellow et al. “Generative Adversarial Networks”. In: *arXiv:1406.2661 [cs, stat]* (June 2014). arXiv: 1406.2661. URL: http://arxiv.org/abs/1406.2661 (visited on 02/11/2018).

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[20] Ishaan Gulrajani et al. “Improved Training of Wasserstein GANs”. In: *arXiv:1704.00028 [cs, stat]* (Mar. 2017). arXiv: 1704.00028. URL: http://arxiv.org/abs/1704.00028 (visited on 02/11/2018).

[21] Isabelle Guyon et al. “Analysis of the KDD Cup 2009: Fast Scoring on a Large Orange Customer Database”. In: *Proceedings of the 2009 International Conference on KDD-Cup 2009 - Volume 7*. KDDCUP’09. Paris, France: JMLR.org, 2009, pp. 1–22. URL: http://dl.acm.org/citation.cfm?id=3000364.3000365 (visited on 05/12/2018).

[22] Anne Håkansson. “Portal of Research Methods and Methodologies for Research Projects and Degree Projects”. en. In: *Computer Engineering* (2013), p. 8.

[23] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *arXiv:1312.6114 [cs, stat]* (Dec. 2013). arXiv: 1312.6114. URL: http://arxiv.org/abs/1312.6114 (visited on 04/04/2018).

[24] P. J. Lin et al. “Development of a Synthetic Data Set Generator for Building and Testing Information Discovery Systems”. In: *Third International Conference on Information Technology: New Generations (ITNG’06)*. Apr. 2006, pp. 707–712. DOI: 10.1109/ITNG.2006.51.

[25] *Microsoft Azure Cloud Computing Platform & Services*. en. URL: https://azure.microsoft.com/en-us/ (visited on 02/12/2018).

[26] Mehdi Mirza and Simon Osindero. “Conditional Generative Adversarial Nets”. In: *arXiv:1411.1784 [cs, stat]* (Nov. 2014). arXiv: 1411.1784. URL: http://arxiv.org/abs/1411.1784 (visited on 04/05/2018).

[27] Andrew Y. Ng and Michael I. Jordan. *On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes*. 2001.

[28] Sybil P. Parker, ed. *McGraw-Hill Dictionary of Scientific and Technical Terms (5th Ed.)* New York, NY, USA: McGraw-Hill, Inc., 1994. ISBN: 978-0-07-042333-6.

[29] Taoxin Peng and Florian Hanke. "Towards a Synthetic Data Generator for Matching Decision Trees". In: *Proceedings of the 18th International Conference on Enterprise Information Systems*. ICEIS 2016. Portugal: SCITEPRESS - Science and Technology Publications, Lda, 2016, pp. 135–141. ISBN: 978-989-758-187-8. DOI: 10.5220/0005829001350141. URL: https://doi.org/10.5220/0005829001350141 (visited on 04/04/2018).

[30] *Recurrent Layers - Keras Documentation*. URL: https://keras.io/layers/recurrent/ (visited on 09/03/2018).

[31] Arthur L. Samuel. "Some studies in machine learning using the game of Checkers". In: *Ibm Journal of Research and Development* (1959), pp. 71–105.

[32] H. Surendra and H.S. Mohan. "A Review Of Synthetic Data Generation Methods For Privacy Preserving Data Publishing". In: *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH* 6.03 (Mar. 2017). ISSN: 2277-8616. URL: http://www.ijstr.org/final-print/mar2017/A-Review-Of-Synthetic-Data-Generation-Methods-For-Privacy-Preserving-Data-Publishing.pdf (visited on 08/01/2018).

[33] *The Sustainable Development Goals Report 2018*. URL: https://unstats.un.org/sdgs/report/2018 (visited on 08/14/2018).

[34] Fjodor van Veen. *The Neural Network Zoo*. en-US. URL: https://www.asimovinstitute.org/author/fjodorvanveen/ (visited on 08/16/2018).

[35] Rick Wicklin. *Simulating Data with SAS*. en. SAS Institute, Apr. 2013. ISBN: 978-1612903323.

[36] Jungang Xu, Hui Li, and Shilong Zhou. "An Overview of Deep Generative Models". In: *IETE Technical Review* 32 (Dec. 2014), pp. 131–139. DOI: 10.1080/02564602.2014.987328.

[37] Lantao Yu et al. "SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient". In: *arXiv:1609.05473 [cs]* (Sept. 2016). arXiv: 1609.05473. URL: http://arxiv.org/abs/1609.05473 (visited on 02/11/2018).

Appendix A

Results of case study 1

This appendix section shows the data generation times figure, the overall metric results, the Wasserstein distance measurements, and the correlation distances figures for the generation of 2X, 10X and 100X the original size of dataset 1.

VARIANT	SIZE	Amount	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	SUM	AVG
GAN-1L-LAST	2X	42,00	0,56	0,37	0,15	0,21	0,16	0,13	0,19	0,13	0,18	0,11	0,10	44,30	3,41
GAN-1L-GEN	2X	37,85	0,55	0,44	0,16	0,14	0,16	0,17	0,13	0,11	0,19	0,10	0,06	40,05	3,08
GAN-1L-DISC	2X	63,45	2,39	3,00	3,83	6,56	0,47	0,95	0,49	1,73	0,49	1,37	0,27	86,00	6,62
GAN-1L-LAST	10X	12,88	0,63	0,36	0,19	0,19	0,13	0,10	0,11	0,15	0,20	0,07	0,10	15,10	1,16
GAN-1L-GEN	10X	37,78	0,61	0,36	0,15	0,18	0,13	0,14	0,15	0,11	0,16	0,08	0,07	39,93	3,07
GAN-1L-DISC	10X	56,01	2,40	2,96	3,91	6,45	0,49	0,92	0,48	1,78	0,52	1,42	0,25	78,59	6,05
GAN-1L-LAST	100X	10,99	0,81	0,40	0,24	0,19	0,14	0,11	0,14	0,14	0,17	0,07	0,10	13,52	1,04
GAN-1L-GEN	100X	24,85	0,56	0,32	0,15	0,10	0,14	0,04	0,14	0,12	0,16	0,09	0,06	26,74	2,06
GAN-1L-DISC	100X	65,54	2,62	2,99	3,87	6,49	0,47	1,06	0,48	1,80	0,55	1,40	0,21	88,46	6,80
GAN-3L-LAST	2X	19,55	1,53	0,27	0,14	0,26	0,39	0,37	0,52	0,25	0,25	0,15	0,19	23,87	1,84
GAN-3L-GEN	2X	25,54	0,70	0,35	0,19	0,17	0,12	0,12	0,20	0,18	0,20	0,18	0,26	28,21	2,17
GAN-3L-DISC	2X	64,51	0,96	0,36	0,90	0,29	0,20	0,24	0,21	1,02	0,23	0,23	0,28	69,43	5,34
GAN-3L-LAST	10X	14,04	1,39	0,36	0,16	0,19	0,51	0,40	0,61	0,17	0,26	0,26	0,25	18,62	1,43
GAN-3L-GEN	10X	22,20	0,56	0,44	0,23	0,38	0,15	0,11	0,24	0,16	0,23	0,23	0,26	25,20	1,94
GAN-3L-DISC	10X	71,24	1,21	0,33	0,85	0,18	0,15	0,29	0,25	0,99	0,27	0,22	0,25	76,23	5,86
GAN-3L-LAST	100X	20,32	1,27	0,29	0,20	0,23	0,47	0,31	0,54	0,24	0,27	0,19	0,20	24,52	1,89
GAN-3L-GEN	100X	24,84	0,80	0,39	0,18	0,30	0,21	0,13	0,23	0,14	0,20	0,21	0,22	27,86	2,14
GAN-3L-DISC	100X	80,81	1,24	0,36	0,79	0,23	0,25	0,25	0,22	1,00	0,26	0,21	0,22	85,85	6,60
GAN-5L-LAST	2X	48,66	1,39	0,42	0,60	0,50	0,20	0,11	0,25	0,27	0,30	0,52	0,44	53,64	4,13
GAN-5L-LAST	10X	42,31	1,17	0,40	0,59	0,43	0,21	0,10	0,26	0,31	0,42	0,49	0,43	47,14	3,63
GAN-5L-LAST	100X	33,46	1,17	0,46	0,64	0,38	0,22	0,12	0,24	0,26	0,35	0,53	0,49	38,32	2,95
ITS-LAST	2X	13,59	0,09	0,09	0,07	0,06	0,04	0,06	0,05	0,07	0,03	0,02	0,04	14,22	1,09
ITS-LAST	10X	3,20	0,04	0,04	0,03	0,04	0,03	0,01	0,02	0,02	0,02	0,01	0,02	3,51	0,27
ITS-LAST	100X	2,69	0,02	0,01	0,02	0,01	0,00	0,02	0,01	0,01	0,01	0,01	0,01	2,83	0,22
CRITS-LAST	2X	3,27	0,11	0,12	0,06	0,05	0,05	0,04	0,03	0,04	0,04	0,03	0,02	3,86	0,30
CRITS-LAST	10X	2,66	0,04	0,02	0,02	0,02	0,02	0,02	0,01	0,01	0,02	0,02	0,02	2,88	0,22
CRITS-LAST	100X	1,98	0,02	0,01	0,02	0,02	0,01	0,02	0,02	0,02	0,04	0,01	0,01	2,19	0,17

Table A.1: Wasserstein distance results for dataset 1.

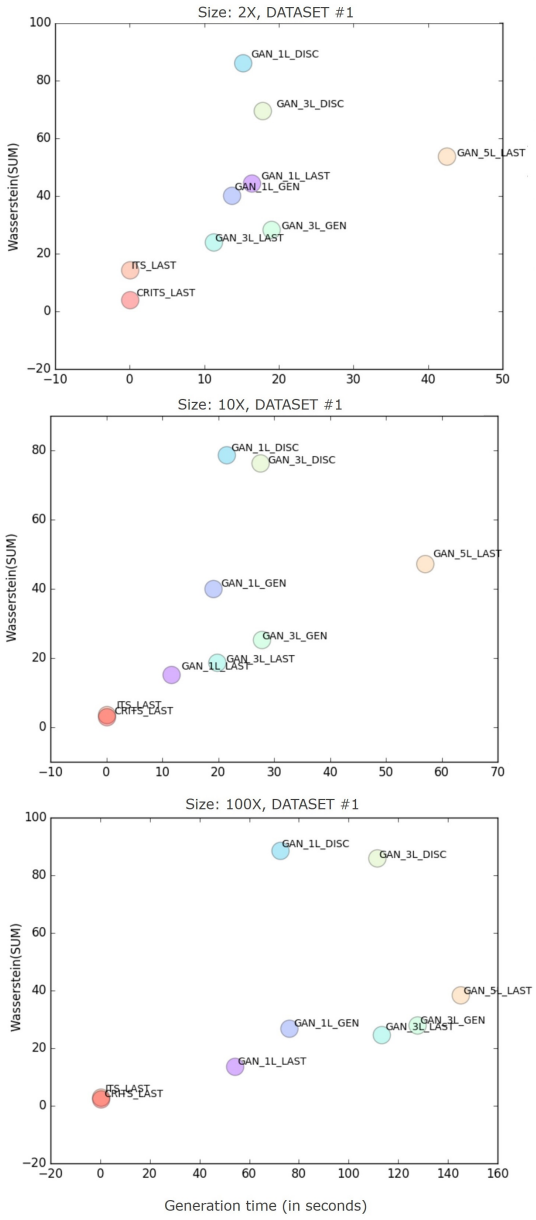


Figure A.1: Wasserstein distances and data generation times (in seconds). The best models in terms of preserving the data distribution of their original datasets and being efficient (data generation time), are the ones shown at the bottom left corner of each plot.

VARIANT	SIZE	TIME	Orig.	Gen.	Corr.	Nil	Dup.	%	Rep.	%
GAN-1L-LAST	2X	16,4	1000	2000	24,2	0	0	0	0	0
GAN-1L-GEN	2X	13,73	1000	2000	24,7	0	0	0	0	0
GAN-1L-DISC	2X	15,22	1000	2000	22,39	0	0	0	0	0
GAN-1L-LAST	10X	11,67	1000	10000	21,73	0	0	0	0	0
GAN-1L-GEN	10X	19,17	1000	10000	23,31	0	0	0	0	0
GAN-1L-DISC	10X	21,55	1000	10000	22	0	0	0	0	0
GAN-1L-LAST	100X	54,34	1000	100000	21,36	0	0	0	0	0
GAN-1L-GEN	100X	76,2	1000	100000	23,52	0	0	0	0	0
GAN-1L-DISC	100X	72,6	1000	100000	21,88	0	0	0	0	0
GAN-3L-LAST	2X	11,29	1000	2000	16,91	0	0	0	0	0
GAN-3L-GEN	2X	19,03	1000	2000	28,73	0	0	0	0	0
GAN-3L-DISC	2X	17,85	1000	2000	19,76	0	0	0	0	0
GAN-3L-LAST	10X	19,86	1000	10000	17,86	0	0	0	0	0
GAN-3L-GEN	10X	27,84	1000	10000	29,83	0	0	0	0	0
GAN-3L-DISC	10X	27,59	1000	10000	17,79	0	0	0	0	0
GAN-3L-LAST	100X	113,4	1000	100000	17,36	0	0	0	0	0
GAN-3L-GEN	100X	127,8	1000	100000	29,96	0	0	0	0	0
GAN-3L-DISC	100X	111,6	1000	100000	17,41	0	0	0	0	0
GAN-5L-LAST	2X	42,55	1000	2000	21,2	0	0	0	0	0
GAN-5L-LAST	10X	57,06	1000	10000	24,81	0	0	0	0	0
GAN-5L-LAST	100X	145,2	1000	100000	24,53	0	0	0	0	0
ITS-LAST	2X	0,07	1000	2000	9,68	0	0	0	0	0
ITS-LAST	10X	0,12	1000	10000	8,52	0	0	0	0	0
ITS-LAST	100X	0,38	1000	100000	8,01	0	0	0	0	0
CRITS-LAST	2X	0,07	1000	2000	9,59	0	0	0	0	0
CRITS-LAST	10X	0,12	1000	10000	9,64	0	0	0	0	0
CRITS-LAST	100X	0,38	1000	100000	8,03	0	0	0	0	0

Table A.2: Overall metrics for the experiments with case study 1. This table shows the generation time in seconds, the original and generated size, the correlation distance, the number of null values, the number and the percentage of duplicates, and the number and the percentage of repetitions from the original data.

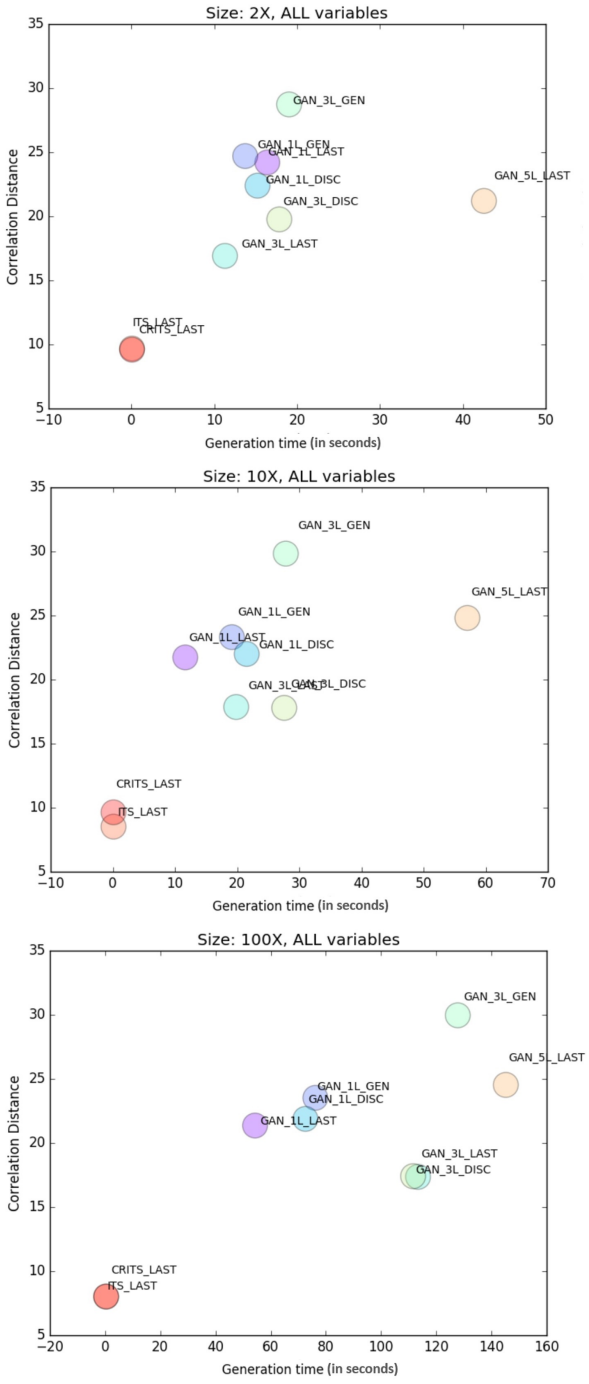


Figure A.2: Correlation distance and data generation times (in seconds). The best models in terms of preserving the correlation patterns of their original datasets and being efficient (data generation time), are the ones shown at the bottom left corner of each plot.

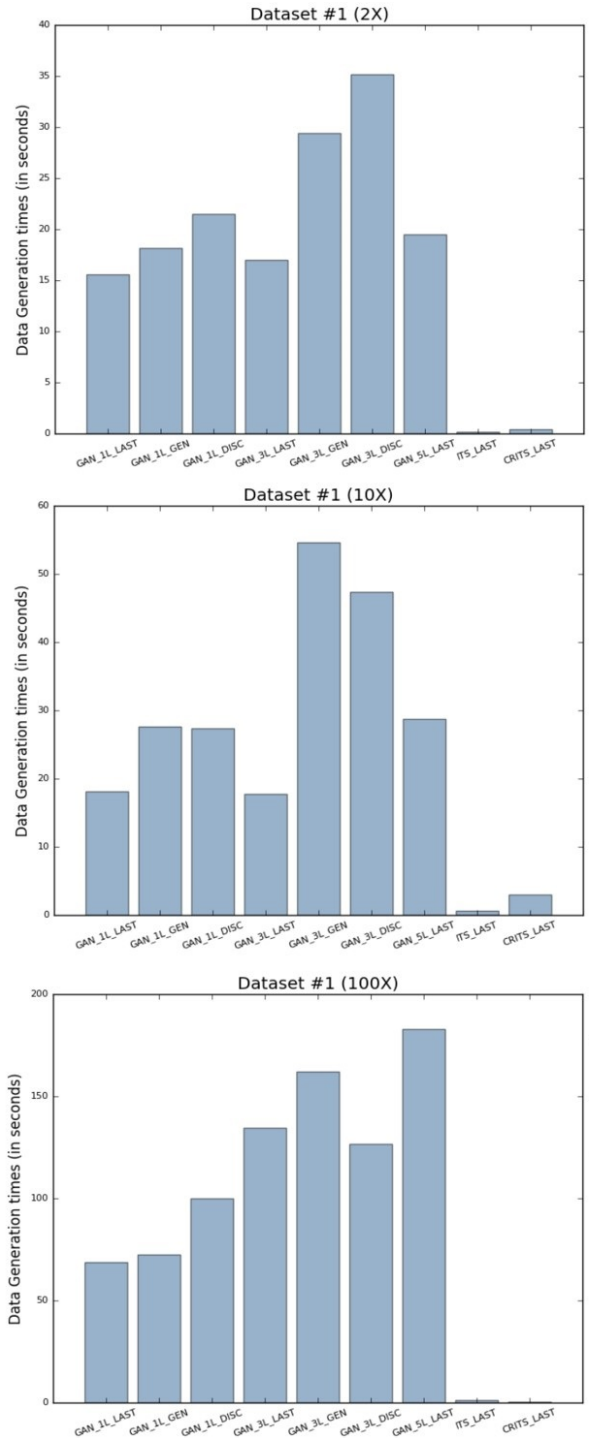


Figure A.3: Average data generation times (in seconds). Each bar correspond to an experiment variant and the data generation time.

Appendix B

Results of case study 2

This appendix section shows the overall metric results and the Wasserstein distance measurements for the generation of 2X, 10X and 100X the original size of dataset 2. Because of the high number of categorical and continuous values, the Wasserstein distance measurements are displayed in three different tables.

VARIANT	SIZE	gender	maritalstat.	cust.susp.	educ.	state	occupation	homeowner
GAN-1L-LAST	2X	0,01	0,03	0,00	0,08	1,63	0,00	0,01
GAN-1L-GEN	2X	0,01	0,02	0,00	0,11	3,50	0,02	0,02
GAN-1L-DISC	2X	0,01	0,00	0,89	0,16	6,53	0,89	0,77
GAN-1L-LAST	10X	0,00	0,00	0,00	0,09	1,22	0,01	0,01
GAN-1L-GEN	10X	0,00	0,01	0,00	0,10	3,60	0,00	0,01
GAN-1L-DISC	10X	0,00	0,00	0,88	0,17	7,13	0,89	0,76
GAN-1L-LAST	100X	0,00	0,00	0,00	0,10	1,29	0,01	0,00
GAN-1L-GEN	100X	0,00	0,00	0,00	0,10	3,56	0,01	0,01
GAN-1L-DISC	100X	0,00	0,01	0,88	0,16	6,96	0,89	0,76
GAN-3L-LAST	2X	0,00	0,01	0,01	0,04	1,08	0,05	0,05
GAN-3L-GEN	2X	0,02	0,03	0,00	0,01	1,95	0,04	0,01
GAN-3L-DISC	2X	0,00	0,01	0,00	0,02	14,32	0,22	0,69
GAN-3L-LAST	10X	0,01	0,01	0,00	0,01	1,34	0,04	0,04
GAN-3L-GEN	10X	0,00	0,00	0,00	0,01	1,94	0,01	0,02
GAN-3L-DISC	10X	0,01	0,00	0,00	0,02	14,75	0,22	0,69
GAN-3L-LAST	100X	0,01	0,01	0,00	0,02	1,25	0,03	0,04
GAN-3L-GEN	100X	0,00	0,01	0,01	0,02	2,02	0,01	0,01
GAN-3L-DISC	100X	0,01	0,00	0,00	0,03	14,43	0,23	0,70
GAN-5L-LAST	2X	0,00	0,03	0,01	0,04	1,14	0,16	0,04
GAN-5L-LAST	10X	0,01	0,03	0,00	0,05	1,28	0,14	0,05
GAN-5L-LAST	100X	0,01	0,02	0,00	0,06	1,28	0,12	0,05
ITS-LAST	2X	0,01	0,00	0,00	0,03	0,48	0,03	0,01
ITS-LAST	10X	0,00	0,00	0,00	0,01	0,27	0,01	0,00
ITS-LAST	100X	0,00	0,00	0,00	0,00	0,07	0,00	0,00
CRITS-LAST	2X	0,49	0,31	0,01	0,15	0,91	0,01	0,03
CRITS-LAST	10X	0,49	0,31	0,01	0,17	1,16	0,01	0,02
CRITS-LAST	100X	0,49	0,31	0,01	0,17	1,10	0,02	0,02

Table B.1: Wasserstein distance results for dataset 2 (Part 1).

VARIANT	SIZE	tot.call.dur.	avg.call.dur.	month	c.d.rate	age	num.m.unpaid	id
GAN-1L-LAST	2X	649,24	75,91	0,31	0,01	8,15	1,01	1054,14
GAN-1L-GEN	2X	154,78	26,55	0,27	0,02	8,80	0,47	379,62
GAN-1L-DISC	2X	425,47	1447,21	2,81	0,01	81,25	7,99	16287,32
GAN-1L-LAST	10X	647,78	78,03	0,31	0,01	8,08	1,02	1076,61
GAN-1L-GEN	10X	138,29	16,84	0,27	0,03	8,34	0,49	383,74
GAN-1L-DISC	10X	459,81	1452,48	2,84	0,01	82,99	7,97	16028,60
GAN-1L-LAST	100X	639,59	76,93	0,31	0,01	8,13	1,01	1073,31
GAN-1L-GEN	100X	123,07	23,12	0,27	0,03	8,26	0,48	373,71
GAN-1L-DISC	100X	447,40	1467,54	2,84	0,01	82,09	8,01	16287,26
GAN-3L-LAST	2X	473,88	58,64	0,27	0,01	6,84	1,17	820,62
GAN-3L-GEN	2X	239,01	47,53	0,29	0,02	13,19	2,61	515,69
GAN-3L-DISC	2X	1238,67	35,13	0,42	0,02	64,16	2,92	2871,56
GAN-3L-LAST	10X	486,42	60,69	0,27	0,01	6,48	1,09	809,36
GAN-3L-GEN	10X	249,08	49,41	0,28	0,02	14,07	2,60	574,59
GAN-3L-DISC	10X	1256,62	37,83	0,41	0,02	63,85	2,96	2876,33
GAN-3L-LAST	100X	471,26	58,53	0,27	0,01	6,49	1,09	812,42
GAN-3L-GEN	100X	257,02	49,57	0,28	0,02	13,96	2,61	551,00
GAN-3L-DISC	100X	1264,79	36,80	0,41	0,02	63,86	2,94	2905,18
GAN-5L-LAST	2X	526,33	76,05	0,27	0,01	6,11	1,16	846,61
GAN-5L-LAST	10X	556,32	70,87	0,27	0,01	5,96	1,12	887,21
GAN-5L-LAST	100X	541,87	71,75	0,27	0,01	5,92	1,13	881,94
ITS-LAST	2X	49,02	7,83	0,02	0,00	0,28	0,05	70,60
ITS-LAST	10X	18,98	2,71	0,01	0,00	0,21	0,01	23,75
ITS-LAST	100X	6,05	1,21	0,00	0,00	0,10	0,01	6,25
CRITS-LAST	2X	36,34	8,78	0,01	0,00	0,52	0,03	70,72
CRITS-LAST	10X	39,99	2,43	0,01	0,00	0,23	0,02	18,76
CRITS-LAST	100X	9,22	0,88	0,00	0,00	0,09	0,02	10,42

Table B.2: Wasserstein distance results for dataset 2 (Part 2).

VARIANT	SIZE	u.bal.	n.comp.	c.f.rate	m.b.a.	a.income	pen.	SUM	AVG
GAN-1L-LAST	2X	31,97	0,50	0,00	16,03	30210,51	58,64	32108,19	1605,41
GAN-1L-GEN	2X	15,60	11,43	0,05	11,55	22905,57	42,63	23561,02	1178,05
GAN-1L-DISC	2X	41,41	7,87	0,00	169,62	232232,14	684,76	251397,11	12569,86
GAN-1L-LAST	10X	30,86	0,51	0,00	15,88	31040,30	57,22	32957,94	1647,90
GAN-1L-GEN	10X	14,88	11,56	0,04	10,04	21404,58	47,28	22040,10	1102,01
GAN-1L-DISC	10X	40,38	7,71	0,00	169,40	240260,80	706,26	259229,07	12961,45
GAN-1L-LAST	100X	31,08	0,51	0,00	15,92	30942,92	58,33	32849,46	1642,47
GAN-1L-GEN	100X	14,80	11,57	0,04	9,84	21229,73	43,83	21842,44	1092,12
GAN-1L-DISC	100X	41,58	7,66	0,00	167,53	239198,22	696,19	258415,99	12920,80
GAN-3L-LAST	2X	23,04	0,44	0,00	11,56	22591,20	49,75	24038,67	1201,93
GAN-3L-GEN	2X	20,63	0,49	0,00	8,86	17969,09	47,13	18866,58	943,33
GAN-3L-DISC	2X	42,36	1,89	0,01	25,94	44665,56	100,21	49064,12	2453,21
GAN-3L-LAST	10X	22,47	0,43	0,00	11,44	23924,88	50,64	25375,65	1268,78
GAN-3L-GEN	10X	19,68	0,49	0,00	8,22	18179,96	42,74	19143,13	957,16
GAN-3L-DISC	10X	43,30	1,99	0,01	26,55	44625,28	103,23	49054,10	2452,71
GAN-3L-LAST	100X	22,66	0,42	0,00	11,46	23450,23	50,29	24886,51	1244,33
GAN-3L-GEN	100X	19,77	0,50	0,00	8,36	17716,85	41,76	18663,79	933,19
GAN-3L-DISC	100X	43,43	2,01	0,01	27,03	45320,62	105,18	49787,67	2489,38
GAN-5L-LAST	2X	22,31	0,41	0,00	11,07	19866,73	46,69	21405,17	1070,26
GAN-5L-LAST	10X	22,56	0,40	0,00	10,95	21540,57	46,50	23144,31	1157,22
GAN-5L-LAST	100X	22,47	0,41	0,00	11,14	21342,15	47,15	22927,76	1146,39
ITS-LAST	2X	1,12	0,01	0,00	0,82	2730,76	3,66	2864,76	143,24
ITS-LAST	10X	1,09	0,01	0,00	0,42	597,34	1,37	646,20	32,31
ITS-LAST	100X	0,53	0,00	0,00	0,23	213,27	0,39	228,15	11,41
CRITS-LAST	2X	4,07	0,02	0,00	1,36	3668,69	2,06	3794,51	189,73
CRITS-LAST	10X	0,47	0,01	0,00	0,29	957,50	1,58	1023,48	51,17
CRITS-LAST	100X	0,23	0,00	0,00	0,10	202,21	0,30	225,60	11,28

Table B.3: Wasserstein distance results for dataset 2 (Part 3).

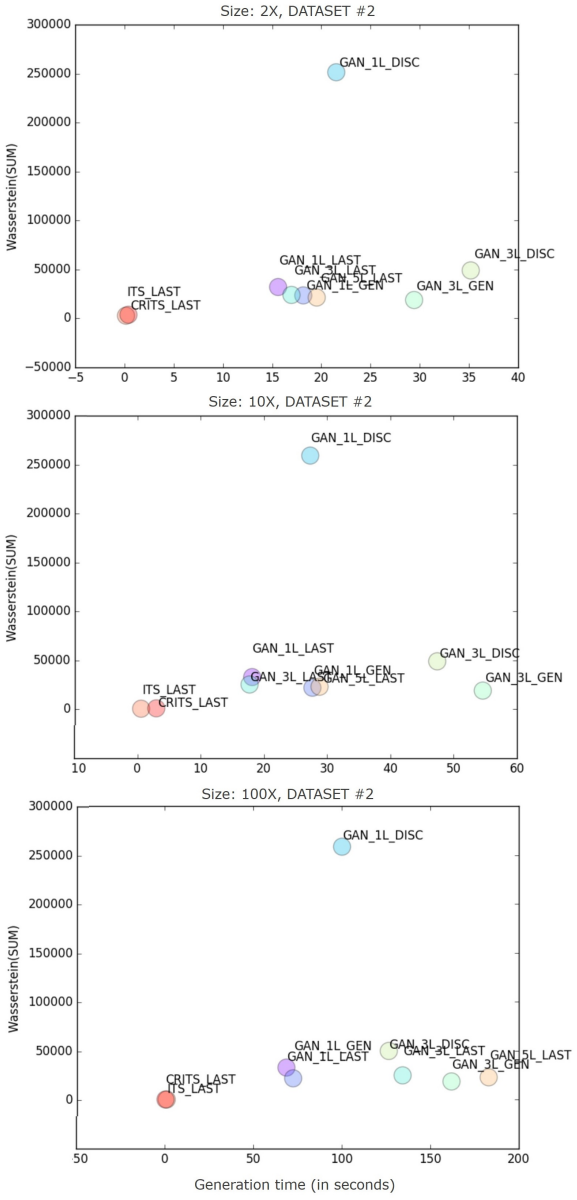


Figure B.1: Wasserstein distances and data generation times (in seconds). The best models in terms of preserving the data distribution of their original datasets and being efficient (data generation time), are the ones shown at the bottom left corner of each plot.

VARIANT	SIZE	TIME	Orig.	Gen.	Corr.	Nil	Dup.	%	Rep.	%
GAN-1L-LAST	2X	15,61	1000	2000	14,96	0	0	0	0	0
GAN-1L-GEN	2X	18,17	1000	2000	10,99	0	0	0	0	0
GAN-1L-DISC	2X	21,52	1000	2000	24,5	0	0	0	0	0
GAN-1L-LAST	10X	18,15	1000	10000	13	0	0	0	0	0
GAN-1L-GEN	10X	27,67	1000	10000	10,11	0	0	0	0	0
GAN-1L-DISC	10X	27,33	1000	10000	23,33	0	0	0	0	0
GAN-1L-LAST	100X	68,8	1000	100000	12,9	0	0	0	0	0
GAN-1L-GEN	100X	72,6	1000	100000	9,28	0	0	0	0	0
GAN-1L-DISC	100X	100,2	1000	100000	22,8	0	0	0	0	0
GAN-3L-LAST	2X	16,97	1000	2000	10,46	0	0	0	0	0
GAN-3L-GEN	2X	29,43	1000	2000	11,56	0	0	0	0	0
GAN-3L-DISC	2X	35,18	1000	2000	14,17	0	0	0	0	0
GAN-3L-LAST	10X	17,74	1000	10000	8,89	0	0	0	0	0
GAN-3L-GEN	10X	54,6	1000	10000	9,39	0	0	0	0	0
GAN-3L-DISC	10X	47,4	1000	10000	11,53	0	0	0	0	0
GAN-3L-LAST	100X	134,4	1000	100000	8,2	0	0	0	0	0
GAN-3L-GEN	100X	162	1000	100000	8,73	0	0	0	0	0
GAN-3L-DISC	100X	126,6	1000	100000	11,03	0	0	0	0	0
GAN-5L-LAST	2X	19,535	1000	2000	11,04	0	0	0	0	0
GAN-5L-LAST	10X	28,8	1000	10000	9,7	0	0	0	0	0
GAN-5L-LAST	100X	183	1000	100000	9,01	0	0	0	0	0
ITS-LAST	2X	0,16	1000	2000	13,24	0	0	0	0	0
ITS-LAST	10X	0,6	1000	10000	11,45	0	0	0	0	0
ITS-LAST	100X	1,2	1000	100000	11,08	0	0	0	0	0
CRITS-LAST	2X	0,41	1000	2000	16,14	0	0	0	0	0
CRITS-LAST	10X	3	1000	10000	13,53	0	0	0	0	0
CRITS-LAST	100X	0,49	1000	100000	13,48	0	0	0	0	0

Table B.4: Overall metrics for the experiments with case study 2. The table shows the generation time in seconds, the original and generated size, the correlation distance, the number of null values, the number and the percentage of duplicates, and the number and the percentage of repetitions from the original data.

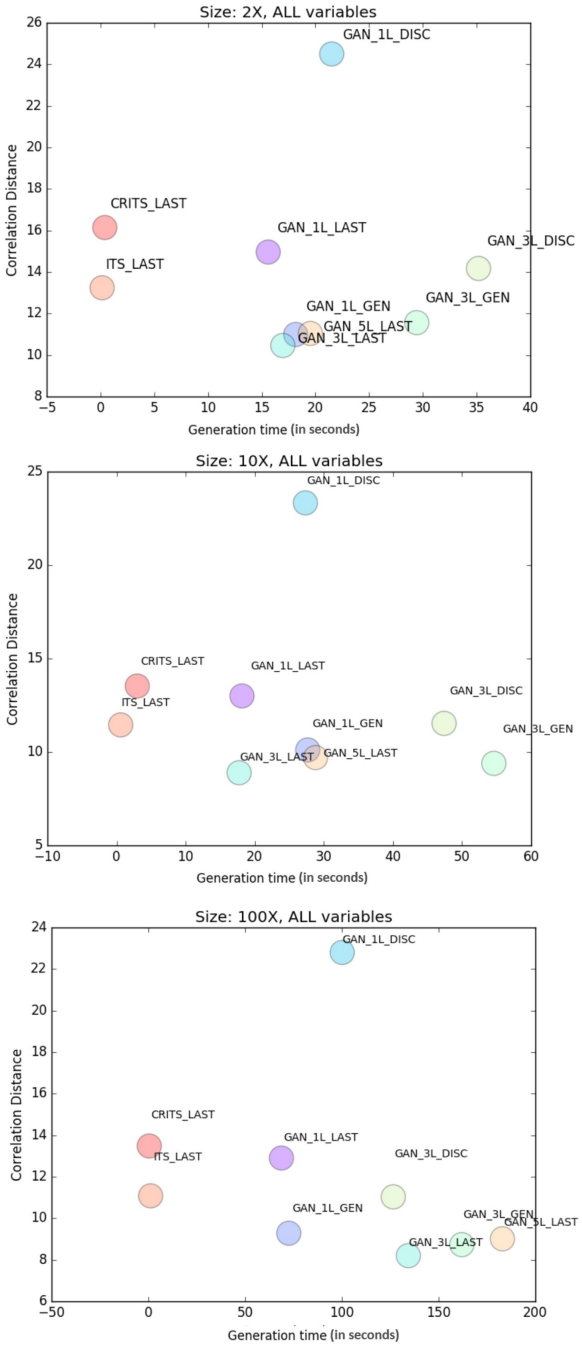


Figure B.2: Correlation distance and data generation times (in seconds). The best models in terms of preserving the correlation patterns of their original datasets and being efficient (data generation time), are the ones shown at the bottom left corner of each plot.

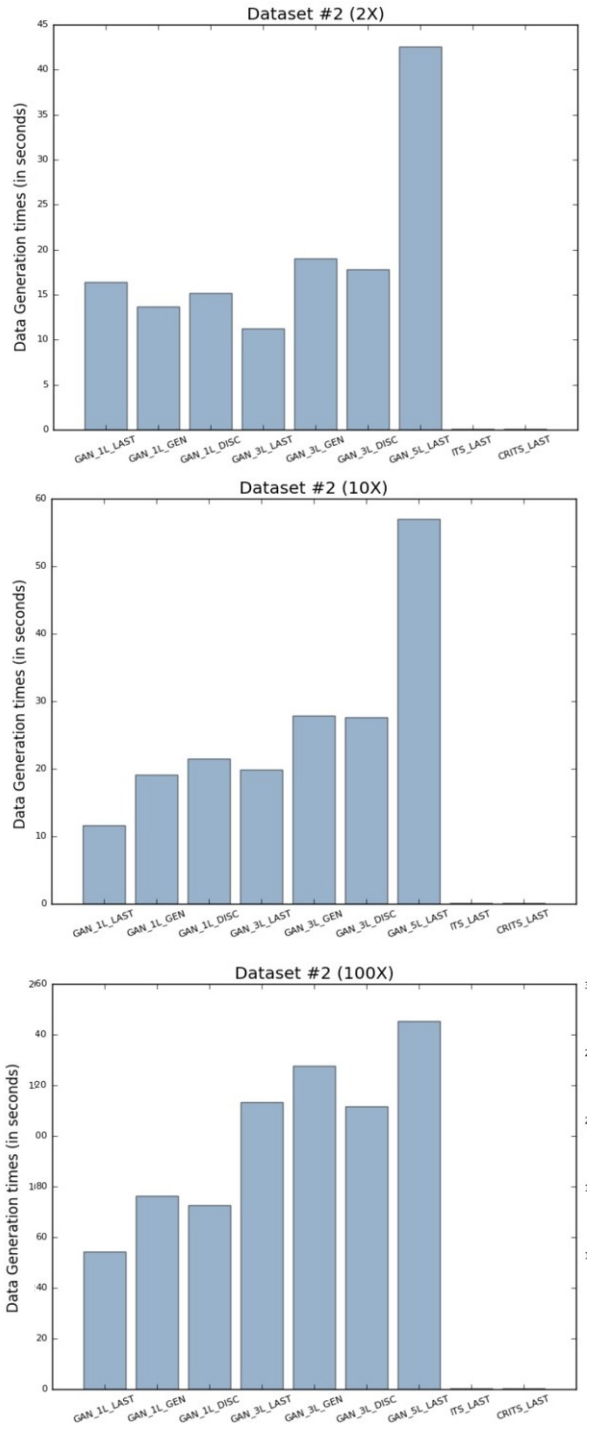


Figure B.3: Average data generation times (in seconds). Each bar correspond to an experiment variant and the data generation time.

Appendix C

Results of case study 3

This appendix section shows the overall metric results, the Wasserstein distance, and the Perplexity metric measurements for the generation of 2X, 10X and 100X the original size of dataset 3.

VARIANT	SIZE	FSTNAME	LSTNAME	PERSNR	PCODE	STREET	TELNUM	SUM	AVG
GAN-1L-LAST	2X	7,356	13,076	16,275	20,921	15,474	12,115	85,217	14,203
GAN-1L-GEN	2X	7,662	12,876	16,200	20,391	15,669	12,114	84,912	14,152
GAN-1L-DISC	2X	25,755	58,557	22,031	26,731	34,179	16,902	184,156	30,693
GAN-1L-LAST	10X	7,339	13,013	15,980	20,948	15,588	12,226	85,093	14,182
GAN-1L-GEN	10X	7,658	12,912	16,006	20,209	15,670	12,210	84,666	14,111
GAN-1L-DISC	10X	25,754	58,219	21,900	26,563	34,513	16,876	183,824	30,637
GAN-1L-LAST	100X	7,343	13,057	16,013	20,903	15,603	12,199	85,118	14,186
GAN-1L-GEN	100X	7,658	12,897	16,064	20,159	15,644	12,199	84,621	14,104
GAN-1L-DISC	100X	25,891	58,454	21,909	26,435	34,181	16,889	183,759	30,626
GAN-3L-LAST	2X	8,005	13,799	16,005	21,664	15,885	12,045	87,402	14,567
GAN-3L-GEN	2X	7,689	13,718	16,010	21,021	16,402	12,090	86,930	14,488
GAN-3L-DISC	2X	14,292	25,891	21,465	25,511	35,996	12,063	135,217	22,536
GAN-3L-LAST	10X	7,899	13,942	15,912	21,920	15,888	12,065	87,626	14,604
GAN-3L-GEN	10X	7,714	13,603	15,829	21,329	16,344	12,213	87,031	14,505
GAN-3L-DISC	10X	14,191	25,865	21,285	25,625	35,941	12,067	134,974	22,496
GAN-3L-LAST	100X	7,896	13,964	15,948	21,757	15,900	12,056	87,520	14,587
GAN-3L-GEN	100X	7,744	13,600	15,807	21,133	16,345	12,171	86,801	14,467
GAN-3L-DISC	100X	14,117	25,854	21,107	25,661	36,134	12,058	134,932	22,489
GAN-5L-LAST	2X	7,369	13,168	15,766	20,398	15,931	12,137	84,768	14,128
GAN-5L-LAST	10X	7,338	13,109	15,619	20,174	15,972	12,163	84,375	14,062
GAN-5L-LAST	100X	7,333	13,103	15,645	20,234	15,932	12,150	84,397	14,066
ITS-LAST	2X	72,464	95,149	22,387	23,970	64,607	20,569	299,147	49,858
ITS-LAST	10X	72,254	96,687	21,550	23,994	62,172	20,765	297,421	49,570
ITS-LAST	100X	71,719	96,262	21,641	23,776	63,842	20,672	297,912	49,652
CRITS-LAST	2X	72,417	92,057	21,824	23,902	64,897	20,148	295,246	49,208
CRITS-LAST	10X	71,386	95,665	21,468	23,896	60,273	20,390	293,078	48,846
CRITS-LAST	100X	71,350	94,662	21,333	23,922	62,525	20,570	294,363	49,060

Table C.1: Perplexity metric measurements for dataset 3.

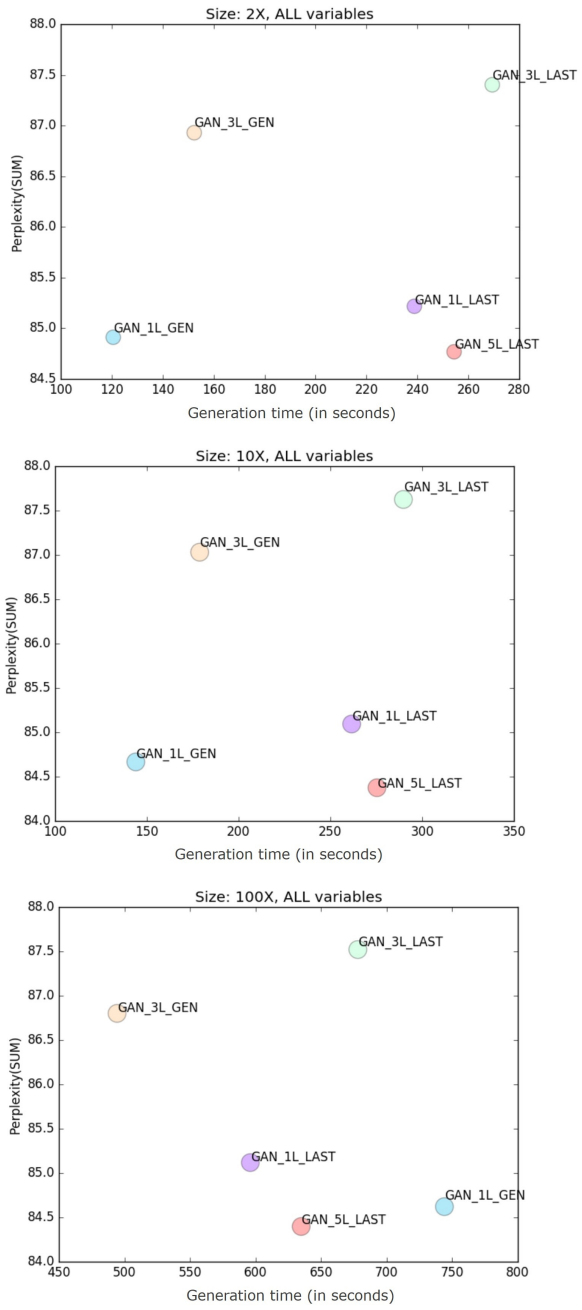


Figure C.1: Perplexity metric and data generation times (in seconds). The best models in terms of generating quality text –i.e. text that is similar to the text in the original dataset– and being efficient (data generation time), are the ones shown at the bottom left corner of each plot.

VARIANT	SIZE	CITY	GEN.	SUM	AVG
GAN-1L-LAST	2X	1,894	0,008	1,900	0,950
GAN-1L-GEN	2X	6,570	0,005	6,580	3,290
GAN-1L-DISC	2X	9,912	0,216	10,130	5,060
GAN-1L-LAST	10X	2,362	0,021	2,380	1,190
GAN-1L-GEN	10X	6,583	0,011	6,590	3,300
GAN-1L-DISC	10X	8,234	0,178	8,410	4,210
GAN-1L-LAST	100X	2,213	0,017	2,230	1,120
GAN-1L-GEN	100X	6,494	0,006	6,500	3,250
GAN-1L-DISC	100X	7,897	0,191	8,090	4,040
GAN-3L-LAST	2X	8,678	0,078	8,760	4,380
GAN-3L-GEN	2X	9,935	0,058	9,990	5,000
GAN-3L-DISC	2X	15,382	0,158	15,540	7,770
GAN-3L-LAST	10X	9,758	0,123	9,880	4,940
GAN-3L-GEN	10X	9,986	0,080	10,070	5,030
GAN-3L-DISC	10X	14,695	0,159	14,850	7,430
GAN-3L-LAST	100X	10,102	0,092	10,190	5,100
GAN-3L-GEN	100X	9,823	0,065	9,890	4,940
GAN-3L-DISC	100X	15,755	0,148	15,900	7,950
GAN-5L-LAST	2X	6,864	0,003	6,870	3,430
GAN-5L-LAST	10X	6,393	0,007	6,400	3,200
GAN-5L-LAST	100X	7,327	0,013	7,340	3,670
ITS-LAST	2X	1,736	0,003	1,740	0,870
ITS-LAST	10X	0,703	0,012	0,710	0,360
ITS-LAST	100X	0,251	0,002	0,250	0,130
CRITS-LAST	2X	2,219	0,043	2,260	1,130
CRITS-LAST	10X	0,597	0,003	0,600	0,300
CRITS-LAST	100X	0,251	0,005	0,260	0,130

Table C.2: Wasserstein distance results for dataset 3.

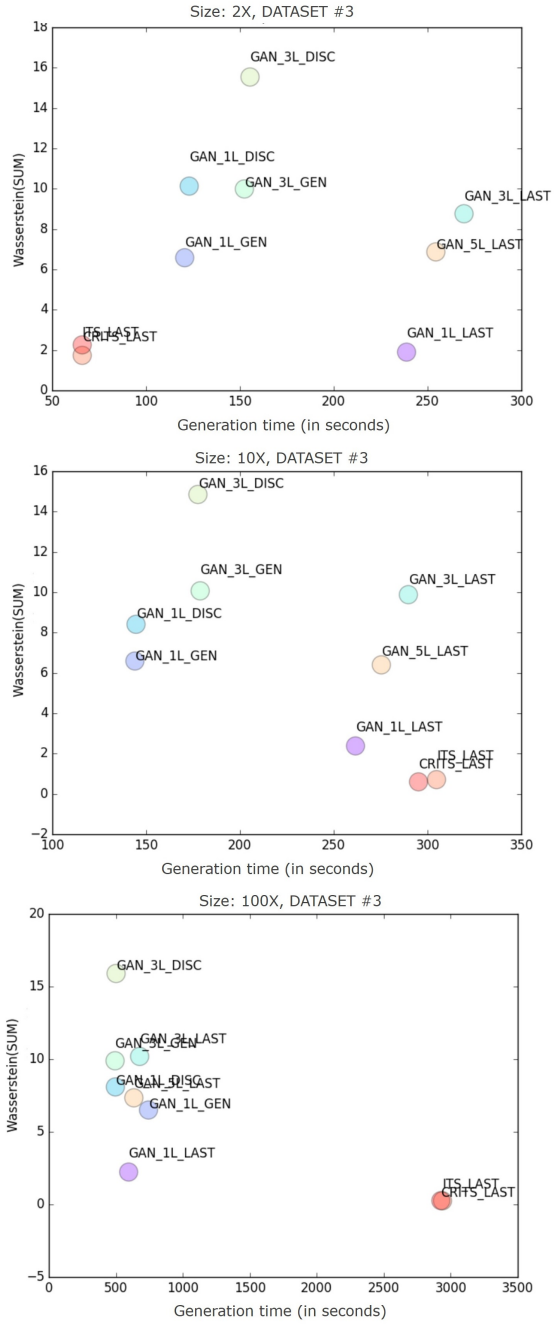


Figure C.2: Wasserstein distances and data generation times (in seconds). The best models in terms of preserving the data distribution of their original datasets and being efficient (data generation time), are the ones shown at the bottom left corner of each plot.

VARIANT	SIZE	TIME	Orig.	Gen.	Corr.	Nil	Dup.	%	Rep.	%
GAN-1L-LAST	2X	238,8	199	398	0,487	0	0	0	0	0
GAN-1L-GEN	2X	120,6	199	398	0,617	0	1	0,25	0	0
GAN-1L-DISC	2X	123	199	398	0,442	0	17	4,27	0	0
GAN-1L-LAST	10X	261,6	199	1990	0,368	0	11	0,55	0	0
GAN-1L-GEN	10X	144	199	1990	0,507	0	25	1,26	0	0
GAN-1L-DISC	10X	144,6	199	1990	0,565	0	184	9,25	0	0
GAN-1L-LAST	100X	596,4	199	19900	0,286	0	1363	6,85	0	0
GAN-1L-GEN	100X	744	199	19900	0,537	0	2380	11,96	0	0
GAN-1L-DISC	100X	496,8	199	17910	0,594	0	1997	11,15	0	0
GAN-3L-LAST	2X	269,4	199	398	0,699	0	0	0	0	0
GAN-3L-GEN	2X	152,4	199	398	0,573	0	1	0,25	0	0
GAN-3L-DISC	2X	155,4	199	398	0,98	0	0	0	0	0
GAN-3L-LAST	10X	289,8	199	1990	0,681	0	9	0,45	0	0
GAN-3L-GEN	10X	178,8	199	1990	0,492	0	31	1,56	0	0
GAN-3L-DISC	10X	177,6	199	1990	0,712	0	2	0,1	0	0
GAN-3L-LAST	100X	678	199	19900	0,692	0	1419	7,13	0	0
GAN-3L-GEN	100X	494,4	199	17910	0,381	0	1970	11	0	0
GAN-3L-DISC	100X	502,2	199	17910	0,682	0	257	1,43	0	0
GAN-5L-LAST	2X	254,4	199	398	0,406	0	0	0	0	0
GAN-5L-LAST	10X	275,4	199	1990	0,35	0	6	0,3	0	0
GAN-5L-LAST	100X	634,8	199	19900	0,35	0	858	4,31	0	0
ITS-LAST	2X	66	199	398	0,571	0	1	0,25	0	0
ITS-LAST	10X	304,8	199	1990	0,707	0	10	0,5	0	0
ITS-LAST	100X	2938,8	199	19900	0,611	0	787	3,95	0	0
CRITS-LAST	2X	66	199	398	0,424	0	0	0	0	0
CRITS-LAST	10X	295,2	199	1990	0,499	0	4	0,2	0	0
CRITS-LAST	100X	2926,2	199	19900	0,511	0	906	4,55	0	0

Table C.3: Overall metrics for the experiments with case study 3. The table shows the generation time in seconds, the original and generated size, the correlation distance, the number of null values, the number and the percentage of duplicates, and the number and the percentage of repetitions from the original data.

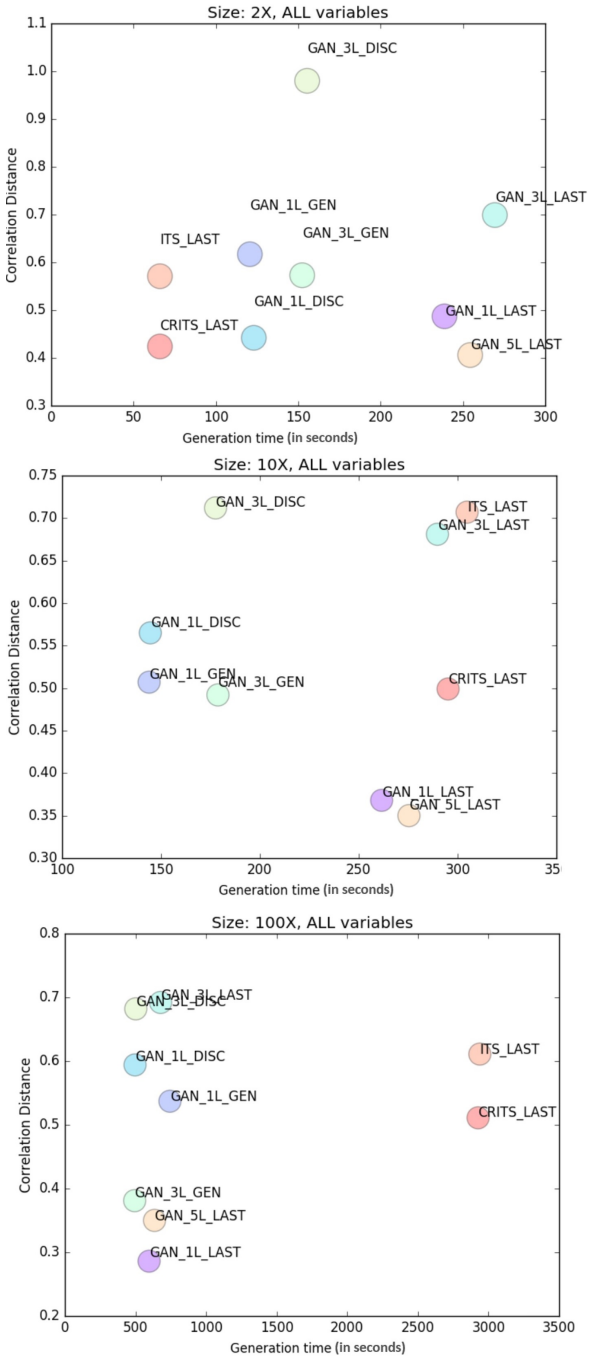


Figure C.3: Correlation distance and data generation times (in seconds). The best models in terms of preserving the correlation patterns of their original datasets and being efficient (data generation time), are the ones shown at the bottom left corner of each plot.

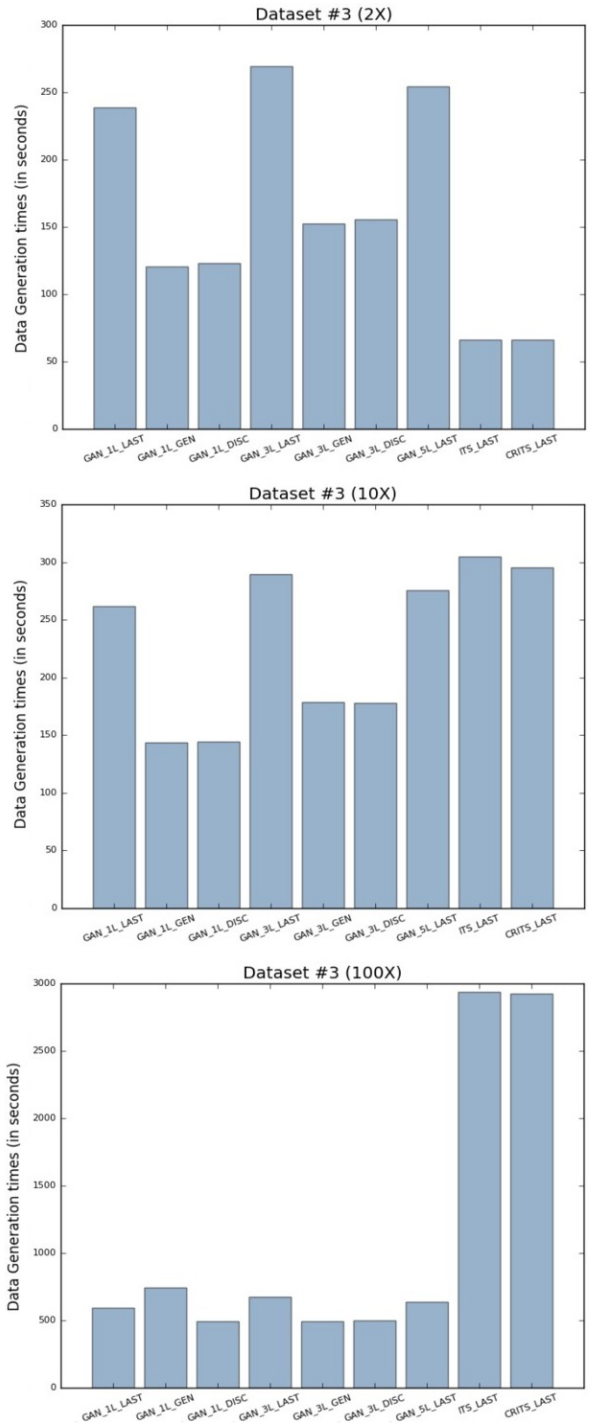


Figure C.4: Average data generation times (in seconds). Each bar correspond to an experiment variant and the data generation time.

TRITA TRITA-EECS-EX-2018:650