UNIVERSIDADE D
COIMBRA

Beatriz Rodrigues Moreira da Silva

SYNTHETIC DATA AUGMENTATION FOR BIOLOGICAL DATASETS

Dissertation in the context of the Master's in Data Science and Engineering, advised by Professor Nuno Lourenço and Professor Francisco Pereira to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September of 2022

Faculty of Science and Technology
Department of Computer Engineering

SYNTHETIC DATA AUGMENTATION FOR BIOLOGICAL DATASETS

Beatriz Rodrigues Moreira da Silva

Dissertation within the scope of the Master in Engineering and Data Science, supervised by Professor Nuno Lourenço and Professor Francisco Pereira and presented to the Faculty of Science and Technology / Department of Informatics Engineering.

September 2022

UNIVERSIDADE D
COIMBRA

Resumo

A busca pela compreensão dos sistemas biológicos, juntamente com seu papel na saúde e na doença, impulsionou a pesquisa das ciências da vida nos últimos dois séculos. No entanto, a pesquisa biológica e bioquímica é muito desafiadora devido à alta complexidade dos sistemas biológicos, onde milhares de moléculas interagem de maneira não linear para orquestrar todas as comunicações intercelulares e intracelulares que ocorrem em cada indivíduo.

Felizmente, com o sucesso comprovado da Inteligência Artificial em muitas outras áreas (por exemplo, sistemas de recomendação, geração de música, tradução de texto e condução automática), os modelos de tomada de decisão baseados em *Machine Learning* começaram a tornar-se omnipresentes no domínio das ciências da vida também. No entanto, existe uma grande diferença entre as aplicações onde a Inteligência Artificial é normalmente aplicada e a área das ciências da vida - a quantidade de dados. Dependendo do domínio-alvo e do desenho experimental, a aquisição de dados biológicos pode ter várias restrições (económicas, falta de amostras, falta de tempo ou questões éticas) que acabam por levar uma situação de poucos dados, tornando difícil a utilização de modelos de Machine Learning.

Embora o número de amostras biológicas possa ser imutável, a geração de dados sintéticos pode compensar o pequeno tamanho dos dados. Neste trabalho propomos um *framework* inspirada na literatura recente, para desenvolver novas técnicas para aumentar vários conjuntos de dados biológicos. O TVAE, o *Variational Autoencoder* explorado na nossa *framework*, conseguiu capturar característica mais importantes dos nossos dados e produzir *datasets* inteiros que exibiam as mesmas propriedades dos dados originais. Os modelos generativos foram avaliados em 2 fases, na primeira fase foi feita a seleção dos melhores modelos generativos possíveis, e na segunda fase estes foram aplicados a *datasets* biológicos do mundo real.

Na primeira fase os modelos alcançaram bons resultados, tais como similaridades de 92% para *datasets* que continham apenas 100 amostras, e valores de utilidade que superaram as performances originais em 4%. Quando aplicados a *datasets* biológicos os modelos apresentaram resultados igualmente satisfatórios, com similaridades de 92% até 100% e utilidades que conseguiram ultrapassar os valores originais também até 4% mais que os valores originais.

Palavras-Chave

Machine Learning, Biologia, Generative Adversarial Networks, Variational Autoencoders, Dados Tabulares

Abstract

The pursuit of understanding biological systems, along with their role in health and disease, has driven the life sciences research in the last two centuries. However biological and biochemical research is very challenging due to the high complexity of biological systems, where thousands of molecules interplay in non-linear ways to orchestrate all the intercellular and intracellular communications occurring in each individual.

Fortunately, with the proven success of Artificial Intelligence (AI) in many other areas e.g., recommender systems, music generation, text translation, and automatic driving, Machine Learning (ML) based decision-making models started to become ubiquitous in the life-science domain as well. However, there is a big difference between applications in biological data and the first ones - the amount of data. Depending on the target domain and the experimental design, the acquisition of biological data may have several reservations (economic, lack of samples, lack of time, or ethical issues), which ultimately leads to a small data size situation, complicating the extraction of viable information.

Although the number of biological samples may be immutable, the generation of synthetic data can compensate for the small size of the data. In this work we propose a framework inspired by recent literature, to develop new techniques to augment various biological datasets. TVAE, the Variational Autoencoder exploited in our framework, managed to capture the most important features of our data and produce entire datasets that exhibited the same properties as the original data. The generative models were evaluated in 2 phases, in the first phase the selection of the best possible generative models was made, and in the second phase these were applied to real-world biological datasets.

In the first phase, the models achieved good results, such as similarities of 92% for datasets that contained only 100 samples, and utility values that surpassed the original performances by 4%. When applied to biological datasets, the models presented equally satisfactory results, with similarities from 92% to 100% and utilities that managed to exceed the original values also up to 4% more than the original values.

Keywords

Machine Learning, Biology, Generative Adversarial Networks, Variational Autoencoders, Tabular Data

Acknowledgements

Changing fields is not an easy thing to do, and neither is completing a dissertation. However, with the contribution of some people I knew, and others that I came to know, it all became possible, for which I would like to thank them.

The first person I would like to thank is Beatriz, who not only was the first friend I made in this adventure but who also became one of my best friends, sharing with me all the ups and downs of these last 2 years, so for that I can't thank her enough.

Next, I would like to thank everyone from the "Mesa mais fixe do bar": Guilherme, Miguel, João, Leonor, Tiago, José e Gonçalo for being the best friends I could not even ask for, welcoming to a place where I did not know anyone or anything, while they knew everyone and everything so I will cherish them forever.

I would also like to thank my best friend and partner Miguel, who since almost my 1st day in this academy became my biggest supporter and stayed that way for the 5 years that followed. To his parents Antonio and Piedade, I would also like to thank them for being the borrowed family from Coimbra and making me feel like this was my home too.

To my friends from "Saltyshore": Mariana, Beatriz, Rodolfo, João, Diana and Rafaela I thank them for believing in me more than I did in myself and for sharing every day with me despite the different paths that each of us has taken.

I would like to also thank my parents for giving me the opportunity to study and become who I wanted to be.

Finally, I would also like to thank my professors Dr. Nuno Lourenço and Dr. Francisco B. Pereira for being great mentors since almost 3 years ago, when I still didn't know how to write a single line of code.

This work was funded by the FCT - Foundation for Science and Technology, I.P./MCTES through national funds (PIDDAC), within the scope of CISUC R&D Unit - UIDB/00326/2020 or project code UIDP/00326/2020.

Contents

Acronyms

List of Figures

List of Tables

Chapter 1
Introduction

The pursuit of understanding biological systems, along with their role in health and disease, has driven the life sciences research in the last two centuries [1]. Back in 1957, Francis Crick gave a lecture at the symposium on the Biological Replication of Macromolecules, where he proposed a ground-breaking hypothesis: "Once information has got into a protein it cannot get out again. Information here means the sequence of the amino acid residues, or other sequences related to it.". Crick's hypothesis outlined key ideas about gene function and generated a lot of stir in the scientific community of that time, so much so that they came to name it "the Biology Central Dogma"[1]. However, biological systems are far more intricate than what this Crick's Dogma may shine through, and what appeared to be a fundamental 'biological law' became the controversial topic that it still is today, 60 years later. Although this was considered a dogma, i.e, a principle laid down as incontrovertibly true, it received a lot of scrutiny, which was not an uncommon phenomenon, since in biology there are few firm principles and exceptions can be found to every 'fundamental' principle when they are bound to such complex systems as biological systems.

In biological systems, tens of thousands of molecules interplay in a non-linear way to orchestrate all the intercellular and intracellular communications occurring in each individual. Understanding all these interactions is not easy, which is why biological and biochemical research is so challenging. To counter this, it was necessary to develop new cutting-edge tools that could model the biological activities of systems in a holistic way, but that could also offer insights into the molecular details of organisms.

Fortunately enough, with the increasing technological sophistication and the blossoming of Machine Learning algorithms, Artificial Intelligence presented the life-sciences with new data-based tools that had their opportunity in DNA sequencing, gene expression analysis, biological imaging, brain imaging and human-machine interfaces [2]. Looking at the gene expression analysis for example, we can understand the urgency of applying ML models to biological datasets. When we talk about gene expression analysis, the data aims to quantify and characterize specific ensembles of biological molecules such as DNA, RNA, proteins, and metabolites, with the ultimate purpose of understanding how these molecular aggregates interact to determine the structure, function, and dynamics of biological systems which, in turn, can be cells, tissues, organs or organisms. As we can see, this is a task of remarkable complexity, but for that very reason, Machine Learning algorithms can be very useful, since they are known for their ability to detect hidden patterns and extract knowledge in a time-efficient manner [3].

It is easy to see why ML-based decision-making models are becoming ubiquitous in the life-science domain. After all, its success is already known in many other areas: in recommender systems [4], music generation [5], text translation [6], and automatic driving [7]. However, there is a big difference between applications in biological data and the last ones - the amount of data. While the latest solutions use large amounts of data, collecting the same volume in the biological context is often impossible. Traditionally, biological data is acquired from

experiments developed and performed in the laboratory. The need to have controlled experimental protocols is due to having to ensure that what is being measured and the conclusions that are being linked are in fact the result of what is happening in the biological system and not the result of other variables. However, given the empirical nature of biological data, their acquisition may have several downstream restrictions that make the application of data analysis tools quite challenging. Depending on the target domain and the experimental design, the acquisition of biological data may have several constraints such as economic (e.g., high costs) [8], lack of samples (e.g., material losses, low number of organisms to collect from) [9], lack of time (e.g., time-consuming processes of collection, long reaction times of compounds) [10] or ethical issues (e.g., animal sacrifice) [11].

These upstream restrictions often lead to consequences later, notably on data quantity. Very often, the data is collected from several different sources or is subject to experimental errors (systematic or random) but given the experimental restrictions, even if the quality of the data is impaired, the procedures cannot be repeated. Because of this, biological data often includes missing values, extreme data (outliers), or erroneous/inconsistent samples. However, the most dramatic consequence on data quality is the small data size. In many real-world datasets there are problems with data size related to a specific class, which is called class imbalance. Although this is a common problem with data, it is not so common in the field of biology as the class balancing is already covered *à priori* by the experimental design. In biology there is another very common problem, the small data size relative to the high-dimensional feature vectors, i.e., we have a large amount of collected variables on a small number of samples. A small data size has high chances of not being able to represent the population that gave rise to it, creating biases in the results and compromising reproducibility.

Furthermore, when we try to apply more sophisticated models such as Machine Learning models, a small data size limits the extraction of knowledge from these datasets. When a small data size is combined with many variables, it becomes an even bigger problem, even being called "the curse of dimensionality" [12] which, in Machine Learning, refers to the case where the dimensionality is so high in relation to the number of samples, that it ends up increasing the risk of overfitting and making it very difficult to learn any valuable thing.

While the number of biological samples may be unchangeable, the generation of synthetic data points could accommodate for the small data size. Variational Autoencoders (VAEs) have been proving to be quality assets in the synthetic data generation panorama, being involved in the generation of "deep fakes" [13], image colorization [14], and background subtraction [15], among others . These achievements make strong suggestions that there may then be a possible window of opportunity for generative models to solve the small data size problem characteristic of biological data. That is what we will be focusing on in this work.

1.1 Objectives

In this work, we propose a framework to develop new techniques to augment biological datasets. The main objective of our system is to generate synthetic data that exhibits the same properties as the original biological datasets and therefore overcome the limitations of small data in biological contexts. To meet those goals, our work was structured according to the following sub-goals:

- Produce a comprehensive literature review where previous works in the field of synthetic data generation are presented and discussed to identify which are currently the best architectures and practices to respond to the small data size problem.
- Develop VAE approaches to artificially augment biological datasets, using knowledge gathered from the literature review, and from exploratory results obtained with generic datasets.
- Validate the results obtained with the assessments developed by analysing the generated data through the lens of several data quality assessments that test both the precision and representativeness of the synthetic data.

1.2 Contributions

Our work resulted in a number of contributions which we list bellow:

- A comprehensive literature review of the state of the art procedures in the Generative Modeling domain, focused in the Synthetic Augmentation of tabular data.
- Develop of a framework that integrates a Variational Autoencoder (VAE) in a generative pipeline for the synthetic augmentation of biological data.
- Obtained positive results for the generation of generic tabular data and biological data, expanding the knowledge in those two fields.
- Presented the oral communication "Synthetic Data Augmentation for Biological Datasets" in the 56th Annual Scientific Meeting of the European Society for Clinical Investigation held in Bari, Italy, between 8th-10th of June 2022.

1.3 Document Structure

The present document is structured as follows: Chapter 2 is dedicated to the background necessary to the comprehension of this work, focusing on the relevant notions about Machine Learning and biological data and on the literature review of the contributions that already exist in data generation panorama; Chapter 3 presents the methodology used for our experiments and the datasets and generative models explored; In Chapter 4 the results obtained with our system are displayed and discussed; and finally the last section, Chapter 5, presents the conclusions and our perspectives for the future.

Chapter 2
Background

1.4 Machine Learning

Back in the 1950s, scientists wondered if computers could be automated to perform intelligent tasks like humans[16]. The first efforts to make this happen focused on defining explicit rules to guide the intended actions, which was called Symbolic AI [17]. But this strategy can only solve a short set of problems. Symbolic AI is based on the principle of deduction, being necessary to include the maximum of premises/knowledge in the system, when the problem was not explicitly programmable by clear and straightforward rules or the rules where not correctly programmed, Symbolic AI failed to provide solutions to the user, which led to its replacement by Machine Learning.

Machine Learning is a sub-field of Artificial Intelligence that involves the development and application of algorithms that allows the extraction of knowledge from large quantities of data automatically [9]. By depending only on the data, computers can learn the patterns underlying that data and extract themselves the "rules" that in Symbolic AI would be introduced by a programmer. Its applicability is vast because after learning with the data, the models can proceed to generate new knowledge about data sets not yet seen. Machine Learning combines elements from diverse disciplines such as statistics, information theory and probability.

Threshold Logic Unit

In 1943 McCullock and Pitts invented a computational model called the Threshold Logic Unit (TLU) that was intended to mimic a biological neuron. As biological neurons receive information from afferent neurons and may or may not trigger an impulse, artificial neurons do this as well. Depending on whether the inputs have enough weight to reach a threshold θ, the TLU decides whether to trigger a pulse (1) or not (0). TLUs can thus be used for binary classification behaving like a Logistic Regression or linear Support Vector Machine (SVM) classifier. To produce an output, the TLU calculates a weighted sum of the individual weights of each input ($z = w_1x_1 + w_2x_2+...+w_nx_n = x^Tw$) and applies a step function to it such as the Heaviside function or the sign function. So, the outputs are the result $h_w(x) = step(z)$, where $z = x^Tw$.

Perceptron

Later, in 1957, Frank Rosenblatt proposed the Perceptron, a simple Artificial Neural Network (ANN) architecture based on TLUs. The proposed Perceptron consisted only of a layer of TLUs, with connection to all inputs, being therefore called "fully connected layer" or "dense layer". The information that the TLUs of the dense layer process comes from the input layer, which is made up of input neurons, which only pass through the information they receive, but also from a bias neuron that, like the weights, is updated during network training. Thus, computing the outputs of a fully connected layer can be given by the Equation 1.

$$h_{W,b}(X) = \phi(XW + b) \quad (1)$$

Where X is the matrix of input features, with one row per sample and one column per feature, W is the matrix of weights, with one row per input neuron and one column per artificial neuron in the layer but excluding connection weights between the bias neuron, b is the vector that contains the connection weights between the bias neuron and the artificial neurons, with a bias term per artificial neuron, and ϕ is the activation function, which in the case of artificial neurons being TLUs is a step function.

As we mentioned earlier, the weights and biases are updated during network training, which, similarly to what happens in biology, is strongly influenced by the triggering of several neurons in a chain, something known as "the Hebb's rule". Second, Hebb, the connection between two neurons is stronger the more frequently the chained triggering of these same neurons is. Later, Rosenblatt summarized the Hebb's rule with the phrase "Cells that fire together, wire together" and it was based on this assumption that the perceptron training was instituted, allied clearly to the error made by the network when making a prediction. Thus, the Perceptron receives one training instance at a time, makes a prediction, and for each wrong prediction, the connection weights between the neurons that contributed to the correct prediction of the inputs are reinforced because these neurons will reduce the error in the network. This process can be summarized by the Equation 2.

$$w_{i,j}^{(nextstep)} = w_{i,j} + \eta(y_j - \hat{y}_j)x_i \quad (2)$$

In this equation, $w_{i,j}$ is the weight between the i^{th} input neuron and the j^{th} output neuron, x_i is the i^{th} input value, $\hat{y}_j$ is the jth output value, y_j is the jth target output and η is the learning rate.

Multilayer Perceptron

Although Perceptrons showed good results in converging to a solution, this only occurred when the training instances were linearly separable. Due to the decision boundary of each neuron output being linear, perceptrons were incapable of learning more complex patterns that were needed to solve other types of classification problems, such as the XOR problem. This led to the need to develop new alternatives, which happened when instead of using a single Perceptron, scientists realized that they could use multiple stacked Perceptrons, and thus formed the Multilayer Perceptron (MLP). The MLP comprises an input layer, one or more layers of TLUs, called hidden layers, and an output layer also formed by TLUs. All layers are fully connected layers and with the exception of the output layer, they all contain a bias neuron. In order to train, MLPs use backpropagation, a very effective algorithm that in just two steps - a forward and a backward - allows you to tweak all the weights and bias terms in the network in order to reduce the network's error. Thanks to the Gradient Descent algorithm, the backpropagation manages to obtain the optimal values of the parameters as it automatically computes the gradient of the network's error with regard to every model parameter and then uses the Gradient Descent algorithm to find the optimal value of each parameter, the result of the its convergence.

Deep Learning

Deep Learning is a specific domain of Machine Learning that relies on neural networks for its learning process. The word "deep" comes from these neural networks being stacked in

successive layers in order to mimic biological neural networks. Obviously this is a simplified model of a real brain but its success has been proven by several achievements such as image recognition[18], language translation [19], and generation of human faces and voices, known in the media as "deep fakes" [20].

1.5 Generative and Discriminative Models

Machine Learning models can be further classified into two types of models: Discriminative and Generative models. In discriminative models, the predictions made for the target label of unseen samples are made based on conditional probabilities because the focus of these models is on learning the decision boundary that separates the classes of a dataset. On the other hand, generative models learn the individual distribution of each class and it is based on the greater or lesser probability that a sample x belongs to a class y that the target label is determined. Furthermore, since the focus of generative models is on modelling the individual distributions of each dataset class, it is possible, based on the concept of joint distribution, to generate new instances where the desired features and labels coexist.

1.6 Generative Architectures

Long Short Term Memory

Long Short Term Memory Networks are a specific type of Recurrent Neural Networks (RNN), capable of Learning long-term dependencies. Unlike RNNs, LSTM manages to avoid the Vanishing Gradient Problem because it has in its architecture a hidden state h to carry information across the learning process to counter the loss of signals. Furthermore, the information that is passed along the LSTM is controlled by the presence of structures called "gates" that assign the value that each piece of information has.

Variational Autoencoders

Autoencoders [21] are neural networks with an architecture similar to that of a Multi-Layer Perceptron (MLP), with the peculiarity of having the number of neurons in the output layer equal to the number of inputs. In an autoencoder there are always two components: an encoder, responsible for converting the inputs to a latent representation and a decoder that converts this latent representation (codings) into the outputs (Figure 1). The objective is to be able to reconstruct the inputs that are fed to it so that the outputs (reconstructions) are as similar as possible to the inputs. To ensure that, autoencoders use a cost function that penalizes the reconstructions the more different they are from the original data. But can the autoencoder just copy your inputs? The answer is no. The internal representation of the inputs has lower dimensionality than the input data creating a space called "bottleneck", and because of that, the autoencoder is forced to learn only the most important features of the input data and drop the more irrelevant ones. The autoencoder is thus said to be undercomplete.

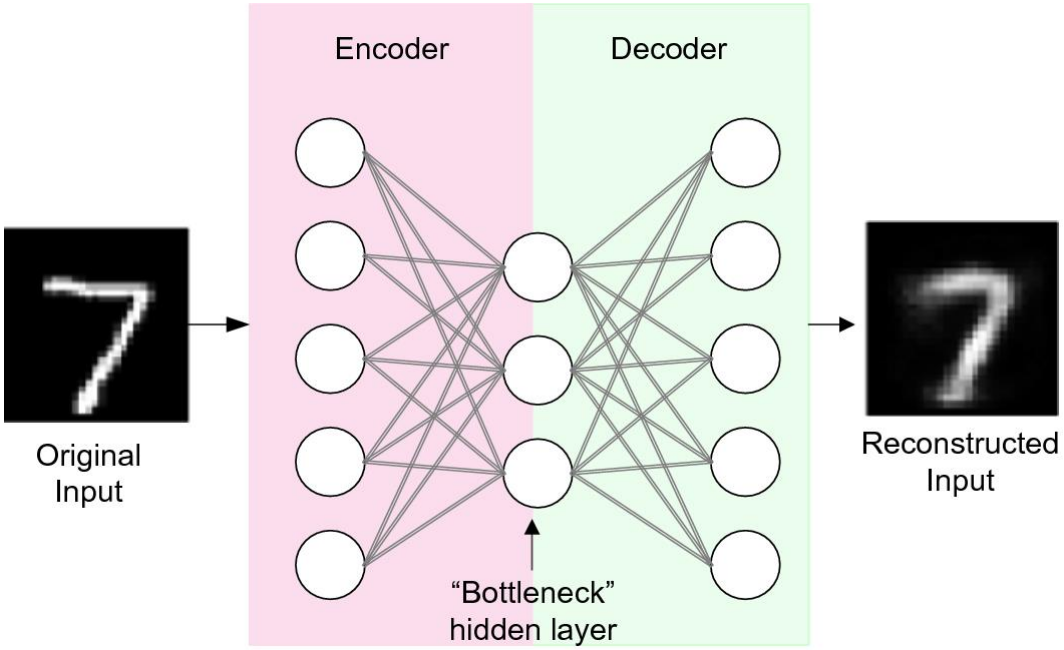

Figure 1 - Illustration of an Autoencoder.

One important variant of autoencoders are Variational Autoencoders (VAE) [22], which differ a little from other variants since they are probabilistic and they are generative. The twist in these autoencoders is that, unlike other autoencoders, VAEs do not output a coding directly from the input. Instead, the encoder first outputs a mean μ and standard deviation σ from the codings and then each coding is sampled from a Gaussian distribution with mean μ and standard deviation σ. Only then does the decoder reconstruct it in output (Figure 2).

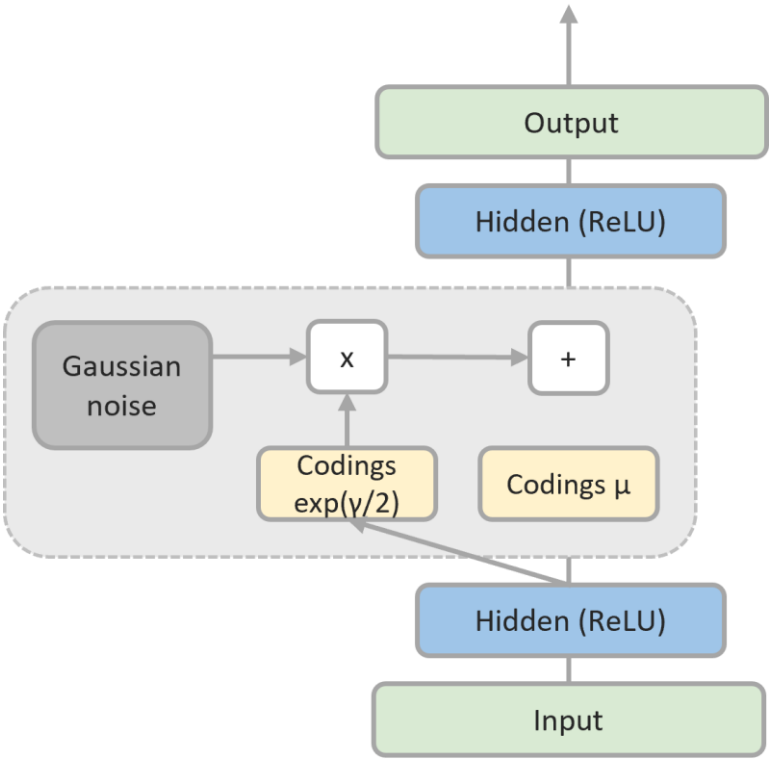

Figure 2 - Illustration of a Variational Autoencoder (VAE).

Looking at the outputted codings, they all seem to have come from a Gaussian distribution, even though the inputs may have had very different distributions. The VAEs achieve this

from the cost function they use during training, which pushes the codings to look like a cloud of Gaussian points. In this cost function there are two parts: the construction loss and the latent loss. The reconstruction loss is the metric responsible for penalising the model as much as the reconstructions differ from the inputs. As seen in many autoencoders, this is a standard loss and one of the most popular examples for this is the cross entropy. The second part of the VAE cost function is the latent loss, which is responsible for pushing the autoencoder to produce codings that appear to have come from a Gaussian distribution. This last loss is the Kullback–Leibler divergence between the target distribution and the initial distribution of the codings and can be written according to Equation 3.

$$L = -\frac{1}{2}\sum_{i=1}^{K} 1 + \gamma_i - exp(\gamma_i) - \mu_i^2 \qquad (3)$$

Where K is the codings dimensionality, μ_i and σ^2_i are respectively the mean and standard deviation of the ith component of the codings, and γ is equal to $\log(\sigma^2_i)$. Since in the VAE case the target distribution is a Gaussian distribution, the codings will gradually migrate during training and in the end they will all appear to have been taken from a simple Gaussian distribution.

The great characteristic about VAE is that after training it, one can easily sample a random coding from the Gaussian distribution, decode it, and have this process repeated many times to generate new data points that can be used to synthetically produce entire datasets, making this autoencoders generative models!

Generative Adversarial Networks

GANs consist of two separate neural networks: a generator G, and a discriminator D, which, as the name suggests, are intended to generate or discriminate data. To generate data, the generator G works very similarly to the decoder of an autoencoder: it receives as input a random distribution p_z (e.g. Gaussian) and then data that intends to resemble real/training data is generated. On the other hand, the discriminator D aims to determine if the data fed to it were generated by G or if they are real data belonging to the training set. To do so, the D discriminator acts like a binary classifier and assigns the label "real" or "fake" according to an estimate of the probability that such data has been sampled from the ground truth distribution p_{data} or not (Figure 3).

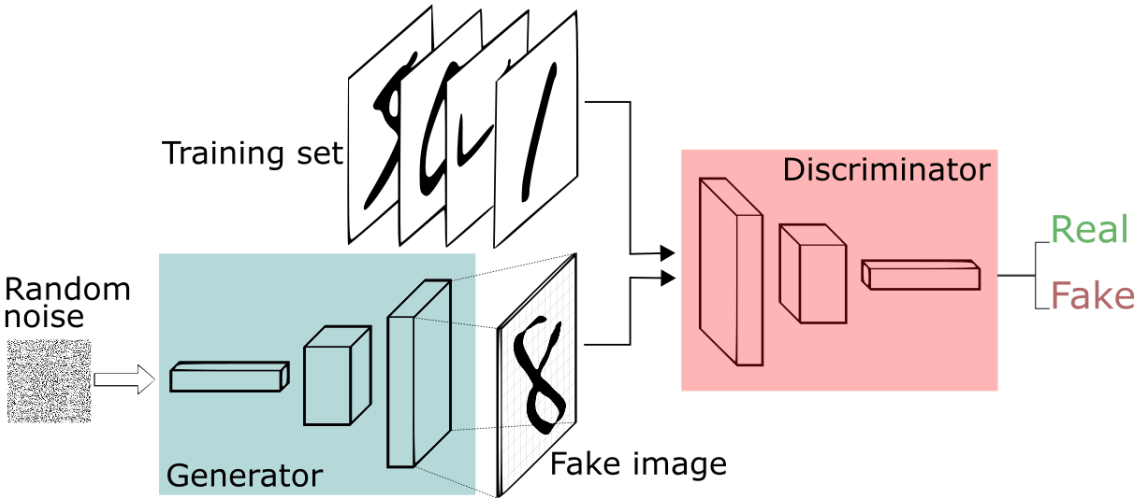

Figure 3 - Illustration of an Adversarial Generative Network (GAN).

Both networks D and G are trained simultaneously, but they have opposite goals since for a generator the performance is better the more fake data is classified as "real" but for a discriminator the performance is better the more fake data is classified as "fake" and vice versa, hence the name "Adversarial". As it is an adversarial training, although the optimization of the weights of one and the other network has to be done separately, the adjustment is made in view of the overall performance of the GAN. Thus, the weights for G are optimized to minimize log(1-D(G(z)), but at the same time, the weights for D are optimized to minimize log D(x) as if the GAN training were a game min-max with value function V(G, D):

$$\min_{\mathrm{G}} \max_{\mathrm{D}} V(G,D) = \max_{G} \max_{D} E_{x\sim p_{\mathrm{data}}(x)}[\log \mathrm{D}(\mathrm{x})] + E_{z\sim p_{z}(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right] \quad (4)$$

1.7 Discriminative Architectures

Decision Trees

Decision Trees are algorithms that follow a hirerchical data structure that contains a series of forks that correspond to individual tests on a given feature. As in real trees,

Decision Trees have nodes, branches and leaves but in this case each node corresponds to a test on an attribute, the branches are the possible outcomes of a test, and the leaves contain the classes to which the samples can belong.

Random Forests

Random Forests are an ensemble learning method based on Decision Trees. In ensemble methods, several models are aggregated to obtain a better predictive performance than what we would get if we used each model in isolation. In the specific case of Random Forests, the way the models are aggregated is by bootstrap aggregating (abbreviated bagging) and that is why Random Forests are voting ensembles, since in bagging all individual models, in this case Decision Trees, have the same weight in the final decision. That is, for a given input x, the classification result of the Random Forest is the average of all the classification results made by the Decision Trees that make up the ensemble.

Extreme Gradient Boosting

Gradient Boosting is a model, which like Random Forests is an ensemble of Decision Trees. However, unlike Random Forests, Gradient Boosting, as the name implies, is built by boosting and not by bagging. The main difference is that, while in bagging the final result is obtained by voting, ie, by averaging the outcomes of all learners, in boosting, each learner has a weight and based on this weight the final result is calculated by computing a weighted average. Another inference that can be drawn from the name "Gradient Boosting" is that it probably uses the Gradient Descent optimization algorithm, which is true, as the models are trained by minimizing the arbitrary differentiable loss function using Gradient Descent. Extreme Gradient Boosting, or XGBoost for short, is an efficient open-source implementation of the Gradient Boosting algorithm.

Support Vector Machines

Support Vector Machines are algorithms that can separate data points into their classes by building a hyperplane - a subspace with N-1 dimensions where N is the total number of dimensions where the data is distributed. In the hyperplane, the distance from the decision boundary to the closest data point is called "the margin" and what SVM tries to do is maximize this margin so that the distance between data points of different classes is also maximum. SVMs can be used for linear or nonlinear classification as they use the kernel trick. Presented by Vapnik, the kernel trick is to increase the dimensionality of the data in such a way that the non-linear data can be separated in the same way that the linear data can, however this trick is computationally effective because although SVMs increase the dimensionality, the number of dataset attributes is not explicitly changed.

Logistic Regression

The Logistic Regression model is a regressor that is used for classification problems because, instead of predicting continuous values, it estimates the probability of a given instance belonging to a class or not. To assign a class, the Logistic Regression model computes the weighted sum of the inputs, plus a bias term, and outputs the logistic result (Equation 5) which has the shape of an S and varies between 0 and 1.

$$\sigma(t) = \frac{1}{1 + exp(-t)} \tag{5}$$

After estimating the probability $p = h_\theta(x)$ of an instance x belonging to the positive class, the model makes its prediction $\hat{y}$ so, imagining for example a binary classification problem, if the estimated probability is greater than 50%, the model classifies the data instance as belonging to the positive class, otherwise it belongs to the negative class (Equation 6).

$$\hat{y} = \begin{cases} 0, & p < 0.5 \\ 1, & p \geq 0.5 \end{cases} \tag{6}$$

1.8 Validation of the Models

When dealing with classification problems, the most popular way of evaluating the performance of a classifier is by analyzing a confusion matrix (Figure 3). The matrix is a compilation of all possible combinations of predicted class and actual class made by the model. So, imagining a binary classification case, in the confusion matrix we will have the true positives (TP) and false negatives (FN) related to the positive class and true negative (TN) and false positives (FP) related to the negative class. Although the matrix is not a metric by itself, from the values it contains we can derive various performance metrics, e.g., accuracy, recall, precision and F1-score.

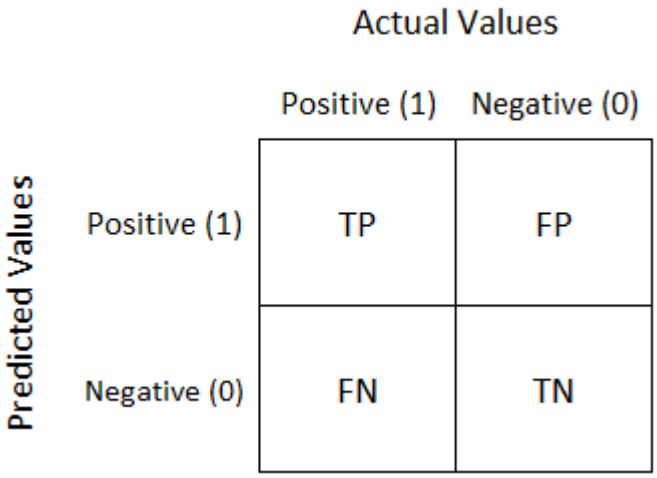

Figure 4 - Representation of a Confusion Matrix.

Accuracy

Accuracy is one of the most used metrics as it gives us a measure of how many labels we are assigning correctly (Equation 7). But it is not advisable if we have unbalanced datasets, because if we have, for example, 95% of samples of the positive class and 5% of the negative class, even if the model classifies all negative samples badly, the accuracy will be 95%.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

Recall

The recall is a metric that tells us how many samples of the positive class were correctly classified (Equation 8). This is a good metric for when we want to penalize false negatives, for example in a medical context, where identifying malignant tumors in benign ones is more serious than identifying malignant tumors in benign ones.

$$recall = \frac{TP}{TP + FN} \tag{8}$$

Precision

Precision is a metric that tells us from the samples that were classified as positive, which ones are actually positive (Equation 9). This is the metric to use when we want to penalize false positives, for example in credit scoring, where it is more serious to assign credit to someone who will not be able to pay it than not to give credit to someone who could.

$$precision = \frac{TP}{TP + FN} \tag{9}$$

F1-score

Precision and recall are both more informative metrics than accuracy in the context of imbalanced data because they are not affected by the proportions of positive and negative samples. For this reason, there is the F1-score metric that combines the two (Equation 10).

$$F1 - score = \frac{2 * precision * recall}{precision + recall} \quad (10)$$

Furthermore, the two metrics precision and recall are contradictory, so by combining the two, the F1-scores ends up being a very informative metric for when we want the best trade-off between the two.

1.9 Data Format

The perspective for 21st century biology is that it will continue to be a data-intensive enterprise, with laboratory data continuing to underpin biology's tradition of being empirical and descriptive [23]. With digitalization, the collection of text, sound, and images became possible and the spectre of biological data formats became wider than the spectre that existed at the beginning of biological research, where data types were reduced to tabular data. However, even with digitalization, biological research continues to be largely based on measurements that are made repeated times to obtain data that is often stored in tables and, if digitalization has made any changes to this acquisition framework, it was by making it faster and more structured. For this reason, biological data of tabular nature continue to represent a large share of biological datasets and keeping this in mind, we will be focusing our work on biological data that is presented in this form.

Tabular Data

Nowadays heterogeneous data is ubiquitous in many important applications like medical diagnosis [24], financial analysis [25], recommender systems [26], cybersecurity [27], psychology [28], among others. Unlike homogeneous data, such as text, sound and image, data from heterogeneous modalities contain a mixture of continuous and discrete attributes belonging to different distributions (e.g. ordinal values, high-cardinality categorical values and binary values).

Within heterogeneous data, the tabular data is the oldest and most common. Recently, several works have tried to model this data with supervised [29], semi-supervised [30] and self-supervised [31] deep-learning frameworks. However, when dealing with tabular data we do not have previous knowledge about the structure and relationships between its attributes. For that reason, learning with tabular data is not a trivial task, raising several challenges that have to be taken into account, as we list in Section 1.9.

Notation

Usually tabular data is presented in a table T like Table 1, which contains n_c continuous columns $\{C_1,...,C_{nc}\}$ and n_d discrete columns $\{D_1,...,D_{nd}\}$(Figure 4). Each column represents a random variables that follows an unknown probability distribution $P(C_{1:nc}, D_{1:nc})$. Each row in the table is a datapoint that, from a probabilistic view, can also be seen as a sample from the joint distribution.

Table 1 – Example of a table.

Product ID	Color	Price
1	red	15.99
2	yellow	23.99
3	green	17.50
4	yellow	9.99
5	red	29.99

1.10 Challenges of Deep Learning in Biological Data

Generating synthetic tabular data is not a trivial task, as we have to account for the mix of attributes both continuous (usually with multi-modal distributions) and discrete (e.g., binary values, ordinal values, high-cardinality categorical values). However, it is oftentimes unclear why the performance on this type of data is not as good as expected. Based on the literature, we then enumerate some possible causes for this phenomenon and discuss them in light of our specific problem of generating synthetic biological data.

Data Quality

As we saw earlier, biological datasets often have a very small data size relative to the high number of features. Although the small data size is also the motivation for our work, it is itself a bottleneck of the solutions we are proposing, since the literature of generative models usually require a considerable amount of data to produce acceptable results. The datasets that are presented in the works where these models are proposed have typically at least few hundred data points but, in the case of biological datasets we usually do not have that, since normally there are no more than 7/8 samples per class. With that being said, the data quality in this study is the motivation for our work but it will also be one of the major problems that we will have to face while searching for a solution.

Missing or Complex Irregular Spatial Dependencies

Within the high-dimensionality of biological datasets, it is not clear whether or not these features are correlated with each other and what is the degree of correlation. Oftentimes, there is no correlation in tabular datasets but in other situations the dependencies between variables are highly complex and irregular. For this reason, the modeling of these data is very complicated as the inductive bias that is normally used to deal with other types of inputs is not adequate (e.g. image processing with convolutional neural networks) [32].

Extensive Preprocessing

As a result of the heterogeneity of biological data, one of the biggest challenges when working with this data is how to handle categorical variables. The obvious approach to dealing with these variables is usually to convert their categorical values to numeric values via one-hot-encoding or ordinal-encoding. However, the application of a one-hot-encoding scheme can result in the increase the number of features, further contributing to the curse of dimensionality and, on the other hand, the application of an ordinal encoding scheme is not the best solution either, because it can lead to the synthetic alignment of unordered values.

Another big challenge lies in the pre-processing of multi-modal continuous features. In most cases, the first step of preprocessing continuous variables is normalization with min-max scaling, so that the values are centered around -1 and 1. However, when the features are multi-modal, it was verified that it can lead to the vanishing gradient problem, which makes neural networks untrainable after successive layers [33].

Model sensitivity

Deep neural networks are characterized for being extremely sensitive to any disturbances in the input data, especially in heterogeneous datasets, such as biological ones[34].

Machine Learning architectures that are not deep, such as decision trees, are much more robust than neural-networks, managing to deal exceptionally well with disturbances[35]. Due to selecting a threshold value for each feature and filtering the data according to that threshold, tree-based architectures have decision-boundaries that look like a hyper-plane. In contrast, deep neural networks have boundaries with high curvature, making them extremely sensitive and less robust architectures[36].

To minimize the sensitivity of neural networks to input data disturbances, it is typically necessary to apply regularizers to the algorithms (e.g. dropout), but the choice of these same regularizers and the combinations to be made with them to obtain the best results is also not trivial.

Furthermore, deep neural networks can have an excessive number of hyperparameters to be optimized (e.g. learning rate, activation function, etc.), whereas other ML architectures such as tree-based ones and support-vector machines have significantly less hyperparameters. The smaller number of hyperparameters in the last models allows them to be optimized in less time, and so, they can reach the best possible results in useful time. In the case of neural networks this is much more difficult to achieve, as it takes too long trying to optimize all the hyperparameters.

1.11 Related Work

Since our main objective is to automatically generate synthetic data, we highlight and discuss some works that are considered state of the art.

From a classical point of view, some researchers propose models based on Copulas and Bayesian Networks. Patki et al. developed the Synthetic Data Vault (SDV) [37], a system based on copulas that builds generative models from relational databases. To do so, SDV iterates through all possible relationships and builds an inter-database model. After training the model, SDV samples from the model and is able to generate synthetic data. In turn, Li et al. introduces Synthetic Data Generation via Gaussian Copula (SYNC) for generating high-resolution data from low-resolution sources (downscaling) [38]. To do so, SYNC trains a Gaussian to copulate on several low-resolution datasets and subsequently scales the sampled datasets, according to the low-resolution marginal constraints of the original data.

Still within the more traditional approaches, Bayesian Networks are also a popular solution, especially studies such as the one by Zhang et al., which uses the discrete probability discriminators approach proposed by Chow and Liu (1968) [39]. Zhang et al. presented a differentially private method for releasing high-dimensional data with the name of PrivBayes that builds a Bayesian Network N that models the correlations between the

variables of a dataset D [40]. The distribution of data is approximated using a set P of low-dimensional margins of D. Subsequently, noise is injected into each marginal in P to ensure differential privacy and an approximation of the data distribution in D is constructed together with the Bayesian network. Finally, the approximated distributions are sampled to construct a synthetic dataset.

However, the more traditional approaches, whether based on copulas or on Bayesian networks, are too dependent on the type of data they want to synthesize, and therefore they were outdated by Deep Learning algorithms, which offer better flexibility and greater performance.

With the blossoming of Deep Learning algorithms, Generative Adversarial Networks (GANs) gained ground, especially in the generation of images, and over time, the synthesis of discrete data was also presented with works using GANs in its favour. Currently, GANs are the state of the art tools for the synthesis of discrete data, especially those derived from Vanilla GANs such as Wasserstein GAN (WGAN) [41], WGAN with Gradient Penalty (WGAN-GP) [42] and Boundary Seeking GAN (BGAN) [43].

The GAN objective is shown to be equivalent to the minimization of the Jensen-Shannon Divergence between the true data distribution and the generated data distribution obtained. However, Arjovky et al. showed that if the differences that are being minimized are not continuous, this can cause difficulties, so he presented the WGAN. For the WGAN, the authors rather proposed the use of the Wasserstein distance, also known as Earth-Mover distance, since this distance can be intuitively seen as the minimum work needed to transform one distribution to another, and the work as the product of mass of the distribution that has to be moved and the distance to be moved. This distance has advantages because it is continuous everywhere and differentiable almost everywhere, which allows to train the model to optimality. In WGAN we then have a different loss, which means that in practice we have a critic instead of a discriminator. The difference that makes a critic instead of a discriminator is that it returns real values, instead of binary values, and its parameters are limited to the range [-c; c], where c is a small constant.

Gulrajani et al. also presented a GAN architecture where instead of a discriminator we have a critic but, in addition, they added a gradient penalty to the critical loss, which is why they named the model WGAN with Gradient Penalty (WGAN-GP). The loss function in this case includes a new hyperparameter called penalty coefficient and depends on the distribution defined by sampling uniformly along straight lines pairs of samples from the true data distribution and the generated data distribution. Furthermore WGAN-GP uses softmax without sampling. The WGAN-GP performed better than standard GAN and enabled stable training of a wide variety of GAN architectures with almost no hyperparameter tuning.

Hjelm et al. contributed to the architectures of the GANs with the Boundary-Seeking GAN (BGAN). The goal was to provide a GAN capable of handling discrete data. In BGAN the generator trains on discrete data and uses the optimized discriminator to estimate f-divergences (e.g., Jensen-Shannon and Kullback-Leibler). The discriminator is then used to formulate importance weights which provide policy gradients for the generator. The authors showed that the model performed well in the generation of discrete images and character-based natural language and that, theoretically, it could also be extended to continuous cases. BGAN showed some better results than WGAN-GP but only in the simple discrete setting.

Side by side with GANs, Autoencoders are also quite popular, particularly Variational Autoencoders (VAE). Next, the most influential works in the field of tabular data generation and that helped to shape this domain are presented in chronological order.

Choi et al. presented in 2017 the medGAN[44], based on Autoencoders and GANs, for the synthesis of medical records. MedGAN is an algorithm capable of generating high-dimensional multi-label discrete samples (e.g., binary and count features) efficiently in such a way that not even experts in the field, in this case doctors, were able to distinguish real data from artificially generated ones. Until then, GANs were used to generate real-valued continuous data with high performance, but they failed to generate discrete data, so the authors incorporated an autoencoder with the original GAN to learn the distribution of discrete data, thus solving this problem. One of the other big problems that can happen in the training of GANs is the mode collapse, which happens when some of the training samples are overfitted and the GAN generator starts to produce only the same type of outputs or the same set of outputs. In order to avoid mode collapse, the authors proposed a method called minibatch averaging that yielded even better results than other methods like minibatch discrimination. To increase the learning efficiency, the authors also applied batch normalization to shortcut the connection. This work was one of the first to provide the generation of synthetic patient data, however, its applicability was limited to discrete data, so it is not easy to generalize medGAN to tabular data.

Inspired by this work, Camino et. al carried out in 2018 a comparative study of 4 architectures similar to those of medGAN and Adversarially Regularized Autoencoders (ARAE) to try to continue the previous work and propose a GAN capable of training in multivariate feature vectors representing multiple categorical values[45]. However, although at work they managed to improve performance in some datasets, this improvement comes at the cost of requiring adding extra information, which in some cases may not always be available.

Abandoning the medGAN idea, Park et al. proposed the table-GAN, which adapts a Deep Convolutional GAN, first introduced by Radford et al., to tabular data, as the name implies[46]. Table-GAN was developed for data anonymization to allow sharing of private data. To this end, the table-GAN converts the attributes of each record into a matrix, so that it is later fed to the deep convolutional network and processed as an image. However, it is unclear to what extent the inductive bias that is used for imaging is suitable for generating tabular data. When the main concern is privacy, models like the Principal Aggregation of Teacher Ensembles GAN (PATE-GAN) can be interesting since they were developed to generate synthetic data with differential privacy[47]. However, as these privacy concerns are outside the scope of this work, we will not talk about them.

Later that year, Xu and Veeramachaneni proposed, similarly to the table-GAN, a GAN for generating tabular data, which they called TGAN[48]. However, there are some differences between both works. The main difference is that while table-GAN uses convolutional neural networks, TGAN uses recurrent neural networks. As a generator, TGAN uses a long-short-term memory (LSTM) network and as a discriminator a multi-layer-perceptron (MLP). Furthermore, while table-GAN explicitly optimizes the performance of the model by minimizing the cross entropy, TGAN focuses on the marginal distribution of each column, which is learned by minimizing the KL divergence. Regarding training the generator, the authors propose an interesting pre-processing step. In previous works, numerical features were normalized with min-max normalization to obtain values centred over (-1,1). This worked because neural networks could efficiently generate values with a distribution between -1 and 1 using the tanh activation function. However, in features that follow

multimodal distributions this does not work, as if there is a mode close to 1 or -1 the gradient will saturate when back-propagating through the tanh leading to the vanishing gradient problem. For this reason, Xu and Veeramachaneni presented the "mode-specific normalization" for numerical variables, which transform columns that follow a non-Gaussian probability function. For this, the values of each numerical variable C_i are clustered using a Gaussian Mixture model (GMM) with m components. Subsequently, the means and standard deviations of the m Gaussian distributions are used to represent each ci value. Thus, each value $c_{i,j}$ of a numerical feature is represented by the normalized probability distribution over the m Gaussian distributions ($u_{i,j}$) and by a scalar $v_{i,j}$ to represent the value within the mode. Categorical variables were transformed to one-hot-encoding representation and noise was added to binary variables.

In early 2019, Baowaly et al. proposed two new models, with the objective of generating more realistic synthetic Electronic Health Records (EHRs) than those generated by medGAN, proposed by Choi et al.[49]. As a starting point, the medGAN architecture was modified and the medical Wassertein GAN was generated with gradient penalty (medWGAN) and the medical boundary seekind GAN (medBGAN). Both models were tested on 2 databases and the results were compared using the 3 models (medGAN, medWGAN and medBGAN). Both proposed models outperform medGAN in all cases, and among the three models, boundary-seeking GAN (medBGAN) performed the best. However, as with medGAN, the study was conducted in order to generate discrete data, and it was not easy to generalize neither medWGAN nor medBGAN to tabular data.

Later, Chen et al. proposed the ITS-GAN, with the objective of generating data that satisfies 2 conditions: i) it has similar statistics as the entire table, and ii) it preserves the functional dependencies of the released sub-table[50]. For this, the authors were inspired by the architecture of the GANs and the AEs for the constraints and evaluated the augmentation performance on two representative datasets, the US Census Bureau data, and US Bureau of Transportation Statistics (BTS) data. The authors compare the performance of ITS-GAN with other models, but some were already quite outdated, and the number of datasets tested was also reduced compared to other studies. For these reasons, it is unclear to what extent ITS-GAN performance outperforms state-of-the-art approaches.

With the success demonstrated by TGAN in the previous year, Xu et al. later proposed the CTGAN, a conditional GAN for synthetic data generation[33]. Once again, CTGAN uses "mode-specific normalization" to transform the non-Gaussian continuous columns, but this time with some changes. For the mode-specific normalization of each numeric variable, each column C_i is processed individually but this time each value $c_{i,j}$ is represented as a one-hot vector-indicating the mode, and a scalar indicating the value within the mode. To do so, the method contains three steps. First, a Variational Gaussian Mixture (VGM) is used to estimate the number of modes and fit a Gaussian mixture. Then, for each value $c_{i,j}$ of C_i, the probability of that value coming from each of the modes is calculated, thus obtaining the probability densities that are computed by the Equation 11, where μ_k and ϕ_k are the weight and standard deviation of mode respectively.

$$\beta_{i,j} = 1 + \frac{c_{i,j} - \eta_3}{4\phi_3} \tag{11}$$

Finally, given the given probability densities, a mode is sampled and the value $c_{i,j}$ is normalized according to that same mode. In this way, j can be represented as a one-hot vector $\beta_{i,j}$, where the position of the "1" indicates that it was chosen, and a mode scalar $\alpha_{i,j}$, indicating the value within the mode. Furthermore, categorical features are also pre-

processed differently. In their previous work, Xu and Veeramachaneni represented the categorical data in the form of one-hot vectors and the generator was trained to generate probability distributions over all categories using softmax. However, this way the trivial discriminator can simply distinguish real and fake data just by checking the distribution's sparseness. For this reason, for CTGAN to obtain discrete variables and allow backpropagation the categorical values are pre-processed with the gumbel-softmax trick.

With the pre-processing steps the model can handle multi-modal variables, something that Vanilla GANs normally cannot handle. After pre-processing, CTGAN uses a conditional generator and training by sampling to address data imbalance, avoiding mode-collapse and insufficient training opportunities for minor classes. To compare the performance of CTGAN with a VAE, the authors also developed TVAE, a VAE for the generation of tabular data that still outperformed CTGAN in some of the datasets tested. Both models are currently state-of-the-art.

Some recent works such as those by Wen et al. and Kamthe et al. also propose causal models (Causal-TGAN)[51] and invertible flows (Copula-Flow)[52], respectively, for the generation of synthetic data. However, despite the good results obtained, models based on GANs and VAEs continue to prevail over all other options. However, despite the good results obtained, models based on GANs and VAEs continue to prevail over all other options.

Table 2 - State of the art summary.

Model	Authors
SDV (2016)	Patki et al.
medGAN (2017)	Choi et al.
Table-GAN (2018)	Park et al.
TGAN (2018)	Xu and Veeramachaneni
Not named (2018)	Camino et al.
medBGAN, medWGAN (2019)	Baowaly et al.
ITS-GAN (2019)	Chen et al.
CTGAN, TVAE (2019)	Xu et al.
SYNC (2020)	Li et al.
Causal-TGAN (2021)	Wen et al.
Copula-Flow (2021)	Kamthe et al.

1.12 Synthetic Data Validation

Evaluating generative models is not a trivial task, as different metrics can produce very different results[53]. The most common approach is to define a benchmarking problem and see if the same performance is maintained from the original data to the synthetic data. For this, a classifier or a regressor is trained on the original and synthetic data and the model's performance is tested in the same test dataset. For performance assessment, metrics commonly used to evaluate these models can be used (F1-score, accuracy, recall, precision, r2). This approach is referred to as "machine learning efficacy" or "utility" and is greater, the

greater the generalizability of the generative model. For situations where there is no obvious label, another feature can be used for that purpose.

Another approach is to check if the generated attributes follow the same probability distribution as the original data. From a more quantitative point of view, it is very common to assess the distributional difference by comparing the output distribution to the ground truth. This comparison can be done through metrics such as likelihood, that represent this distribution. However, likelihood can only be calculated on simulated data because it comes from a known distribution, on real datasets we can't compute these metrics. For this reason, statistical tests such as the Kolmogorov-Smirnov test for continuous features or the Chi-Square test for discrete features are used. These tests make it possible to measure the similarity between variables of the two datasets, assessing whether the two distributions are different and whether this difference is statistically significant. Basically, we want to know if the synthetically generated samples could come from the same distribution as the original data, and for that what is expected is that the p-value of the test result is high enough to fail to reject the null hypothesis "The two datasets are from the same distribution". Taking a more visual approach, the distribution of each feature can also be compared by inspecting graphs where the cumulative distribution functions per feature are plotted or scatterplots showing the expected and obtained values.

Finally, to assess whether the relationships between the attributes observed in the real data are maintained in the generated data, Pearson's, or Spearman's coefficients and heatmap tracings of the two datasets can be compared to visually support the correlations.

1.13 Discussion

Many of the challenges of learning from tabular data that we enumerated earlier have been addressed in previous work and with good results. However, when it comes to augmenting data from small datasets, there is still a lot of work to be done. When we talk about small data size referring to a specific class, i.e., class imbalance, there are already several contributions that seem to work. However, when the small data size is in comparison to the high number of features, with balanced classes, the topic is still a lot to explore.

As for the generation of biological data, small data size is a big bottleneck, this study will follow an experimental design that seeks to bring insights on how to solve this problem and later use these insights to synthetically increase biological dataset. The next chapter describes how the investigation of these topics proceeded.

Chapter 3
Proposed Approach

The work conducted in the previous chapter allowed us to identify in previous works the most promising practices for our objective of generating synthetic biological data.

In this chapter we detail our approach, based on a set of generative models, different datasets, and different quality assessment metrics.

3.1 Framework

In this work we presented a framework that integrates a Variational Autoencoder (VAE) in a generative pipeline for the synthetic augmentation of biological data. As this is not a trivial task, our pipeline encompassed 2 main stages: a first stage to select the best possible generative models, and a second stage, where the models generate biological data. To further challenge the models' ability to generalize in small data situations, we conducted a complementary study where the models were forced to learn from progressively smaller datasets.

Figure 5 depictes the two main stages: i) generation of generic data and ii) generation of biological data. Within each stage there are different steps that are the fundamental pieces of our work. As the steps between the two stages are similar to each other, we can further subdivide both stages into 3 main steps:

1) Synthetic data generation
2) Statistical data validation
3) Utility data validation

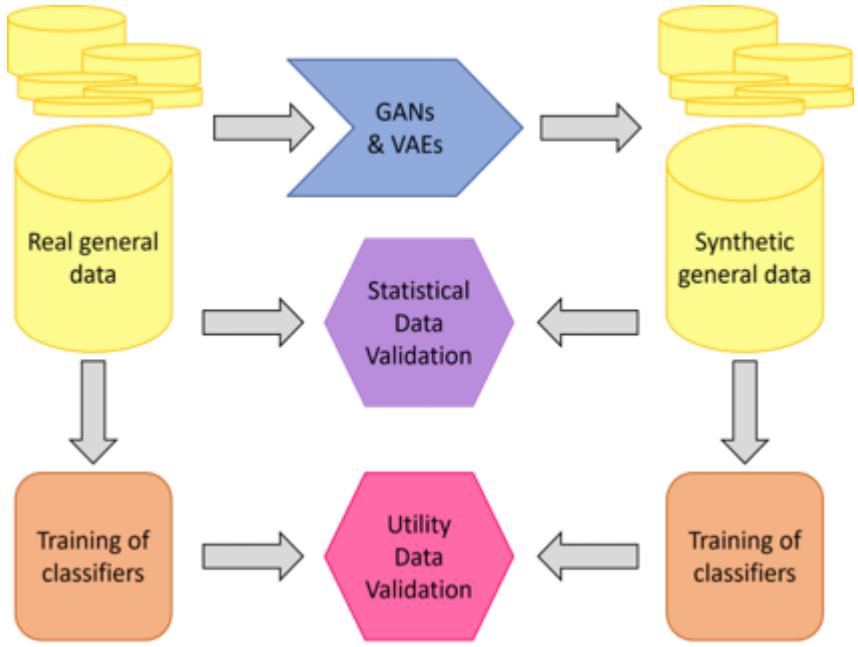

Figure 5 - Representation of stage 1 of the generative framework.

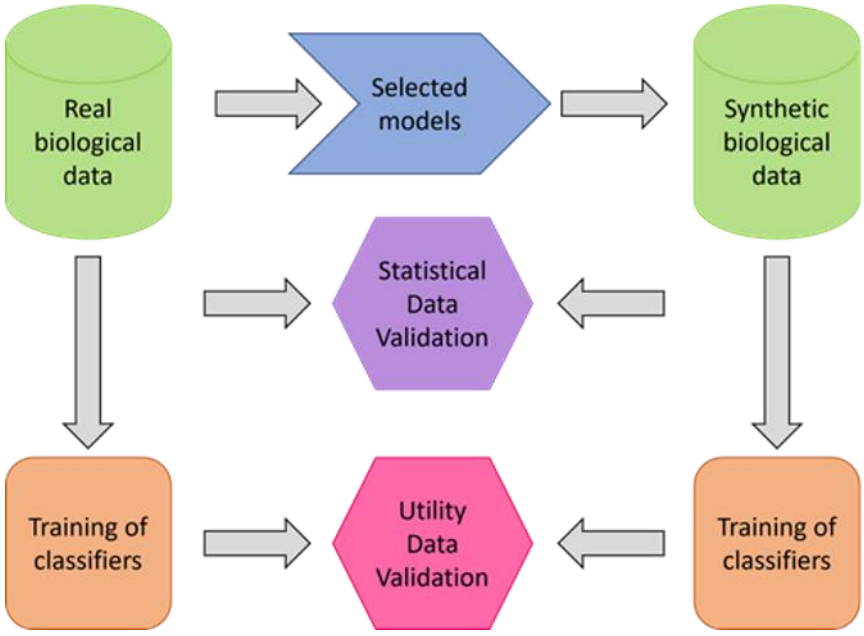

Figure 6 - Representation of stage 2 of the generative framework.

In step 1, the datasets are used to train VAE-based models which, after training, are used to generate synthetic datasets. We further explain this step in section 3.1.1.

In step 2, the newly generated data is evaluated through a statistical lens to anticipate its utility. For this, the generated datasets were evaluated with respect to the distribution of features and to the integrity of the relationships between them. More details about this step are specified in section 5.1.2.

In step 3, the utility of the newly generated datasets was evaluated using a set of several classifiers that were trained on both synthetic and real data and tested on real data to access both performances and compare them. This process is explained in section 5.1.3.

3.1.1 Synthetic Data Generation

As a first step in the data generation pipeline, the data was pre-processed. In this process, the discrete features were transformed into one-hot representations, and the continuous features were transformed by mode-specific normalization, as described in Chapter 2. Once the pre-processing was done, the training of generative models could begin. However, there were some differences in how this process was carried out depending on whether we were in stage 1 (generic data generation) or stage 2 (biological data generation). For this reason, we will explain this step individually for each stage.

Stage 1 – Generation of generic data

In this stage, the data used for training the models did not come from real biological datasets, but from 3 simulated datasets of different fields, containing several hundred samples. As each generic dataset had more records than the ones we would find in biological datasets, we decreased the number of records by randomly sampling 200 samples, equally distributed by class, from each generic dataset. This under-sampling allowed us to approximate our study to a real-life scenario and by doing it equitably between classes we also guaranteed that the datasets were balanced. This is important in this study since dataset balancing is also a common concern when acquiring real biological data.

To further challenge the models' ability to generalize in small data situations, we conducted complementary experiments where the models were forced to learn from progressively smaller datasets. For this part of the study, each dataset previously created (1:1 setup) was successively under-sampled to create smaller data size setups: 1:2, 1:3, and 1:4, corresponding to datasets with 100, 66, and 50 samples respectively. All dataset-setup combinations amounted to a total of 16 training datasets (T_{orig}) or, in other words, 16 different study conditions.

After pre-processing the data, the training of the generative models could begin, for which we used the VAE architecture combined with stratified k-fold cross validation, with k=5, as a training strategy due to the bantam size of the data.

In this stage specifically, the models had to be fine-tuned to search for the optimal values of each model's hyperparameters. For any generative model typologies, several hyperparameters govern the training process and the topology of the model. All these hyperparameters are variables that remain constant during the training process, and they have a direct effect on the performance of the model, so they must be fine-tuned. To search for the best hyperparameters we used Optuna [54], a hyperparameter optimization software that can be combined with Pytorch [55].

All these steps were repeated for each dataset-setup combination. After training on N rows of each T_{orig}, the same number N of rows were synthesized, creating a new dataset of

synthetic samples - T_{syn}. The number of T_{syn} generated was equal to the number of T_{orig} tables used for training and the class balancing was maintained.

Stage 2 – Generation of biological data

The models that proved to be the most promising in stage 1 were then selected to proceed to stage 2 to generate new synthetic biological samples. To choose which model would fit the best biological datasets a decision was made based on not only the results from the first stage but also the characteristics of the real biological data, such as data size and data types.

As these were pre-selected models, data generation in stage 2 comprised fewer steps than this same process in stage 1 since it did not require the additional pre-processing for the different data size setups and there was no need for hyperparameter search.

All models were trained using stratified k-fold cross validation, with k=5, and using the specific training conditions determined in stage 1. After the training was done, each model generated the same number N of synthetic rows as the ones present in T_{orig} and the class balancing was also maintained.

3.1.2 Statistical Data Validation

To assess the quality of the generated data we first evaluated the generated datasets using statistical tools. At this point, our main concerns were to guarantee that the synthetic data resembled the original data in such a way that the synthetic data seemed to have been sampled from the same probability distribution. Additionally, we wanted to guarantee that the relationships that existed between the variables in the original T_{orig} datasets were also found among the features of the synthesized T_{syn} datasets.

To answer the first question, a study was made about the distributional difference between T_{syn} and T_{orig}. To this end, we assessed the Chi-Square (QS) similarity for categorical features and the Kolmogorov-Smirnov (KS) similarity or the Mann-Whitney (MW) similarity depending on whether the numerical features were continuous or discrete, respectively. The above-mentioned similarities were obtained by conducting dimension-wise QS/KS/MW tests for all dimensions and computing the total percentage of similarly distributed dimensions. For this study, we considered the significance level α = 0.05 and plotted heatmaps of the obtained p-values and feature values distribution charts to visually inspect those similarities.

To answer our second question which is concerned with the conservation of the relationships between features we computed matrixes of the correlations between the features of the datasets under study and matrices of the difference between the correlation values of T_{syn} and T_{orig}. The correlation values were obtained by computing the Spearman rank correlation for datasets that contained discrete features and the Pearson's correlation for datasets that did not. After that, we were able to plot heatmaps of those values and visually inspect if the relationships were preserved.

As a sanity test, we also quantified the number of duplicated instances present in T_{syn} to ensure that the models were not producing only the same outputs multiple times, as happens in mode collapse.

3.1.3 Utility Data Validation

The utility is a metric that lets you know if, by training Machine Learning model, it can learn from T_{syn} as they would learn from T_{orig}. For this reason, in this study, utility is an important metric to assess the quality of generative models. In stage 1, the utility was especially important as it was used to ultimately decide which models to move to stage 2 with.

For this step, we relied on supervised machine learning models, to predict the labels of a certain test set T_{test}. The classifiers, trained in their default settings, were 5 of the most commonly used classifiers in the literature: Extreme Gradient Boosting (XGBoost), Logistic Regression, SVM, Decision Tree, MLP, and KNN, and all of them, after predicting the target column, were accessed for their discrimination performance in terms of accuracy, F1-score, recall precision and AUC.

To ensure that the performances of these models were comparable, the models were trained on both the original T_{orig} dataset and the synthetic T_{syn} dataset and the class prediction was done on the same test set T_{test}. The test set and training sets are mutually exclusive, so as to ensure that the test samples were records never seen by any of the classifiers (both those trained on T_{syn} and those trained on T_{orig}) (Figure 7). The training strategy used was once again stratified k-fold cross-validation with k=5. To facilitate the analysis of the utility obtained with each generative model, the individual performance values of each classifier were averaged by the number of classifiers used, which allowed to dilute less satisfactory performances as a result of the use of these same classifiers without any fine-tuning. Thus, generative models that originated synthetic datasets yielding, on average, values similar to real values, were considered good generative models.

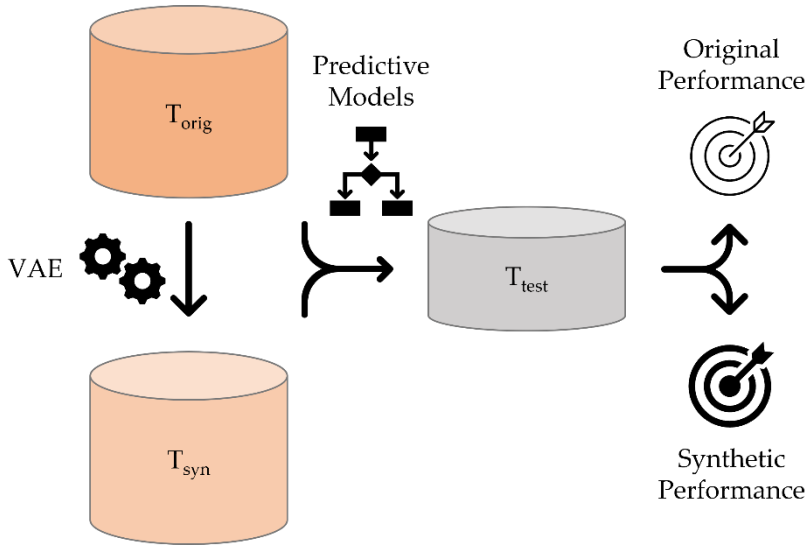

Figure 7 - Illustration of the utility data validation.

3.2 Experimental Design

In the following section, we detail the datasets used for our experimental tests, and the way the models were fine-tuned and trained to then produce the synthetic datasets.

3.2.1 Datasets

In our framework suite, we have 2 different types of datasets: i) tabular datasets typically used in research (generic datasets) and ii) biological datasets. The generic datasets were used in the first stage and the biological datasets in the second.

Generic Datasets

We picked 3 commonly used ML datasets from the UCI Machine Learning Repository (Adult/Census, Wine Quality (White), and Bank Marketing) to be used in stage 1 of our work. The datasets for our framework were chosen to be heterogeneous, like biological datasets, containing both continuous and discrete features. At the end of the pre-processing described in 3.1.1, all datasets under study had, within the same data size setup, the same number of samples regardless of the dataset that originated them, however, the original number of samples, the number of features, and the number of categories varies from dataset to dataset, so we will be describing each one individually next.

1. Adult

O Adult dataset, also known as the Census Income dataset, is a dataset originally extracted from the 1994 Census database, which contains demographic information about a population in the United States of America to determine whether a person's income exceeds $50K/yr. The dataset is composed of 48842 instances, 6 numerical features (e.g., age, capital gain, capital loss), and 8 categorical features (e.g., sex, occupation, relationship) along with a binary classification label.

2. Wine Quality

In the UCI Machine Learning Repository we can find two datasets related to red and white variants of Portuguese wine "vinho verde" from the Oporto region. In this work, we chose the white variant dataset, containing several variables of the physicochemical properties of this wine variant (e.g., fixed acidity, pH, alcohol) and a target variable labelling each data point with a quality score that ranges from 0 to 10. The physicochemical features (11) together with the target label (1) make this dataset suitable for classification tasks. The dataset is composed of 4898 instances and the classes are ordered so that the better wines have a higher quality score, and the inferior ones have lower scores however, the dataset is not balanced (e.g., there are many more normal wines than excellent or poor ones).

3. Bank Marketing

A Portuguese banking institution conducted a marketing campaign through phone calls to convince its customers to subscribe to a term deposit. To predict if other clients would be willing to subscribe to the same program, the financial institution created the Bank Marketing Dataset. This dataset contains 45211 instances, 7 discrete features (e.g., balance, duration), and 10 categorical features (e.g., education, housing), one of which is the label column containing the values "yes" for records where the clients subscribed to the term deposit and "no" otherwise.

Biological Datasets

We also picked 2 real-world biological datasets to finally test our models in real-world conditions: the ATP Rate dataset and the Mitostress dataset. The chosen datasets came from the MitoXT group (Mitochondrial Toxicology and Experimental Therapeutics) at the Center for Neuroscience and Cell Biology of the University of Coimbra. These datasets contained a lot of variables and only a few samples, perfectly exemplifying the common situation witnessed in biological research, where the number of dataset rows is very small compared to their high dimensionality. The Mitostress dataset was composed of 263 records, 7 numeric features (e.g., basal respiration, proton leak), 2 discrete features and 1 categorial target feature. The Mitostress dataset was composed for 268 records, 4 numeric features (e.g., Total ATP Production Rate (Basal) (pmol/min)), 2 discrete features and 1 categorial target feature.

In these datasets, the data mirrored the metabolic and mitochondrial characteristics of lymphoblasts from the Coriell Institute for Medical Research subjected to laboratory experiments to study the Amyotrophic Lateral Sclerosis (ALS) disease. The lymphoblasts were obtained by combining the Epstein Barr virus with lymphocytes that were collected from individuals belonging to 3 experimental groups: control/individuals without ALS (LH), ALS patients without Superoxide Dismutase 1 (SOD1) mutations (LP), and ALS patients with SOD1 mutations (LPS). The records of each experimental group were further subdivided into 3 cohorts: cohort 1 (46-year-old female individuals), cohort 3 (46-year-old male individuals), and cohort 5 (26-year-old male individuals) thus, the different experimental group-cohort combinations amounted to a total of 9 cell lines analyzed.

3.2.2 TVAE

The TVAE developed by Xu et al., receives data that, as we have seen, is pre-processed beforehand, using "mode-specific normalization" to transform the non-Gaussian distributions into continuous columns. This transformation allows us to express the values in the form of a mixture component number with the deviation from that component centre. To allow backpropagation in the presence of categorical features, the gumbel-softmax trick is used.

As for the architecture of TVAE, let r_j (Equation 12) be the representation of a row, where $d_{i,j}$ is the one-hot representation of a discrete value, and $\alpha_{i,j}$ and $\beta_{i,j}$ are respectively the scalar and the one -hot vector resulted from the pre-processing stage described in Chapter 2.

$$r_j = \alpha_{1,j} \oplus \beta_{1,j} \oplus \ \dots \ \oplus \alpha_{N_c,j} \oplus \beta_{N_c,j} \oplus d_{1,j} \oplus \ \dots \ \oplus d_{N_d,j} \qquad (12)$$

To model each $p_\theta(r_j | z_j)$ and $q_\phi(z_j | r_j)$ the model uses two neural networks, and the training is done by using the evidence lower-bound (ELBO) loss. In the design of the $p_\theta(r_j | z_j)$ the network outputs a joint distribution of $2N_c + N_d$ variables, corresponding to $2N_c + N_d$ variables r_j and it is assumed that $\alpha_{i,j}$ follows a Gaussian distribution while $\beta_{i,j}$ and $d_{i,j}$ follow a Probabilistic Mass Function (PMF). The design for the first network is as follows:

$$\begin{cases} h_1 = \text{ReLU}\left(\text{FC}_{|\mathbf{r}_j|\to 128}(\mathbf{r}_j)\right) \\ h_2 = \text{ReLU}\left(\text{FC}_{128\to 128}(h_1)\right) \\ \mu = \text{FC}_{128\to 128}(h_2) \\ \sigma = \exp\left(\frac{1}{2}\text{FC}_{128\to 128}(h_2)\right) \\ q_\phi(z_j \mid \mathbf{r}_j) \sim \mathcal{N}(\mu, \sigma\mathbf{I}) \end{cases} \tag{12}$$

Since the loss function in this case is modified to fit the tabular data it is not entirely correct to say that this is a VAE but rather an adaptation of a VAE. On the other hand, the modeling for qϕ(zj| rj) is similar to a conventional VAE:

$$\begin{cases} h_1 = \text{ReLU}\left(\text{FC}_{128\to 128}(z_j)\right) & \\ h_2 = \text{ReLU}\left(\text{FC}_{128\to 128}(h_1)\right) & 1 \le i \le N_c \\ \bar{\alpha}_{i,j} = \tanh\left(\text{FC}_{128\to 1}(h_2)\right) & 1 \le i \le N_c \\ \hat{\alpha}_{i,j} \sim \mathcal{N}\left(\bar{\alpha}_{i,j}, \delta_i\right) & 1 \le i \le N_c \\ \hat{\beta}_{i,j} \sim \text{softmax}\left(\text{FC}_{128\to m_i}(h_2)\right) & 1 \le i \le N_d \\ \hat{\mathbf{d}}_{i,j} \sim \text{softmax}\left(\text{FC}_{128\to |D_i|}(h_2)\right) & \\ p_\theta(\mathbf{r}_j \mid z_j) = \prod_{i=1}^{N_c} \mathbb{P}(\hat{\alpha}_{i,j} = \alpha_{i,j}) \prod_{i=1}^{N_c} \mathbb{P}(\hat{\beta}_{i,j} = \beta_{i,j}) \prod_{i=1}^{N_d} \mathbb{P}(\hat{\alpha}_{i,j} = \alpha_{i,j}) & \end{cases} \tag{14}$$

3.2.3 Models Fine-tuning

The hyperparameters are variables that control the behavior of algorithms and that are very important because oftentimes they can strongly determine the model's performance. To set these hyperparameters, it is not uncommon to see the programmer introduce the values manually and change them many times to achieve a better accuracy or loss. However, instead of choosing this manually intensive approach, programmers can use a tuning algorithm, which turns the process automatic and much quicker.

To return the best-performing models, we used the tuning algorithm from the Optuna package, which uses two strategies to find the best hyperparameter values within a certain number of trials. The first strategy is a sampling strategy to decide where within the search space the algorithm should look, and the second strategy is a pruning strategy to early terminate a particular trial if it is not looking very promising, providing more time for better trials. For the sampler, we used bayesian filtering to find which places within the search space to look at based on the previous iterations' best results, contrasting with, for example, random search and grid search, which treat the hyperparameter sets independently. As for the pruner, Optuna uses successive halving, an algorithm that randomly samples a set of model configurations, evaluates their performances with a small number of resources, and carries forward to the next rounds of optimization only the top-performing half of the

models. This process is repeated until one configuration remains and through the process the amount of resources allocated increases, meaning that the number of samples that the estimators are trained becomes higher and higher.

To conduct the search process, the tuner attempted to minimize the ELBO loss function and to allow for the testing of different hyperparameter combinations we set the number of trials to 20 trials. In each trial we allowed a maximum of 300 epochs so that the tuner would have a reasonable amount of time to test the models without taking too long however, we used early-stopping to quickly converge on a high-performing model without overfitting the models, since the datasets were so small. We defined patience = 5 to stop the training after 5 epochs without progress on the validation set.

By the end of the fine-tuning process, we had gathered information about the model's best validation loss and the hyperparameter values used to obtain that loss. From there we re-instantiated our model with the optimal parameters and trained it, obtaining an optimized model without overfitting problems. The hyperparameters tested and the search space for each one of them is detailed in Table 3.

Table 3 - Hyperparameters tested for the TVAE model and corresponding search space.

Parameter	Sampling method	Values
Learning Rate	Sampled in the log domain	[1e-6, 1e-3]
Batch size	Sampled from the integers	[10, 200, step=10]
Compress Dimensions	Sampled from choices	[(32, 32), (64,64), (128,128), (256,256), (512,512)]
Decompress Dimensions	Sampled from choices	[(32, 32), (64,64), (128,128), (256,256), (512,512)]
Embedding Dimensions:	Sampled from choices	[32, 64, 128, 256, 512]
Loss factor	Sampled from the integers	[1, 10, step=1]

Chapter 4
Results

In this chapter we detail the experimental results obtained with our framework, regarding all the steps and elements described in the previously.

4.1 Synthetic Data Generation

In Figures 8, 9 and 10, we can find the heatmaps of the difference in correlations existing between the features of the Marketing, Census and Wine datasets respectively. Since the values presented are the difference between correlation values present in the original (T_{orig}) dataset and the synthetic dataset (T_{syn}), the matrixes reveal the relationships that were easier to conserve through cells in white color (~ 0.00), and the ones that the model had more difficulty to mimic, through cells darker in color (~ 1.00).

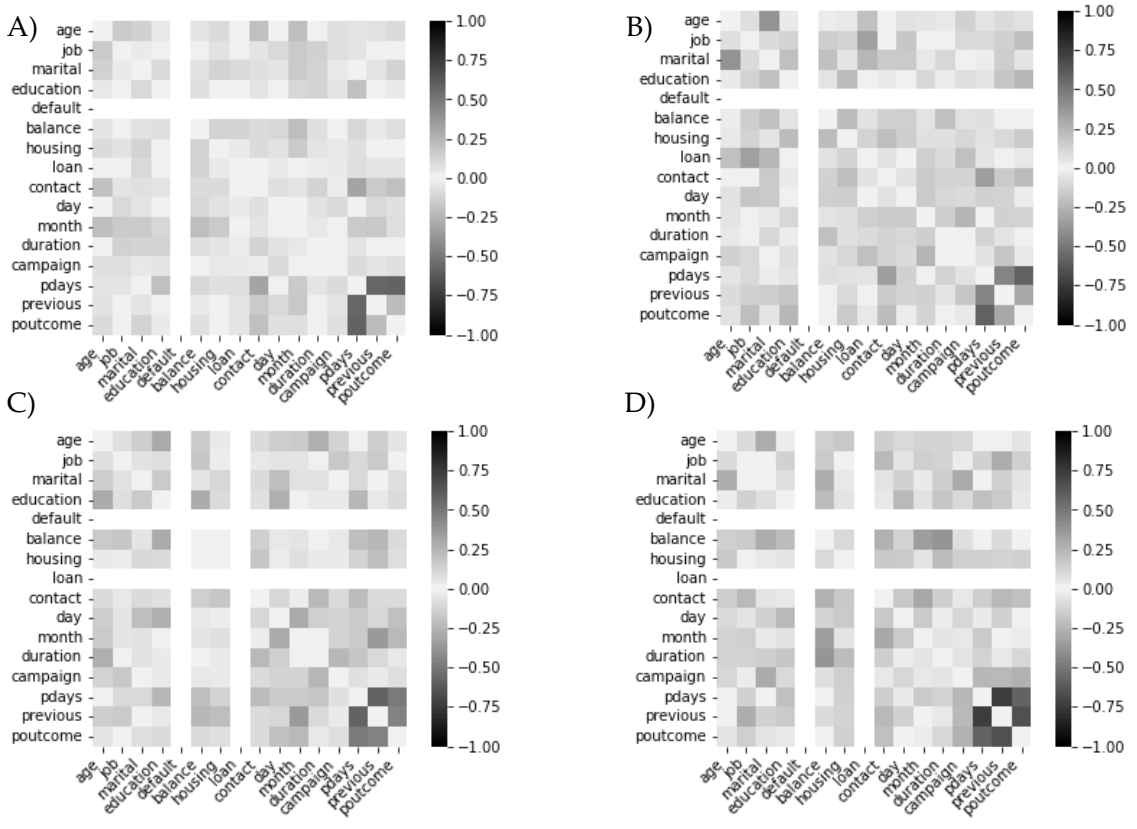

Figure 8 - Heatmaps of the difference between correlation values present in the original (Torig) dataset and the synthetic dataset (Tsyn) of the Marketing dataset. A) Heatmap of the 1:1 setup. B) Heatmap of the 1:2 setup. C) Heatmap of the 1:3 setup. D) Heatmap of the 1:4 setup.

As we can see, the relationships between the features of each one of the 3 original datasets were easily replicated for the most part of the features, as the cells that compose the matrixes are mostly in shades close to white. However, in the Marketing dataset we see that 2 relationships had trouble in being replicated, the relationship between the features "pdays" and "previous" and the relationship between "pdays" and "poutcome", which may be because these features display highly heterogeneous value distribution, where one value has a much higher frequency than the other unique values of the features.

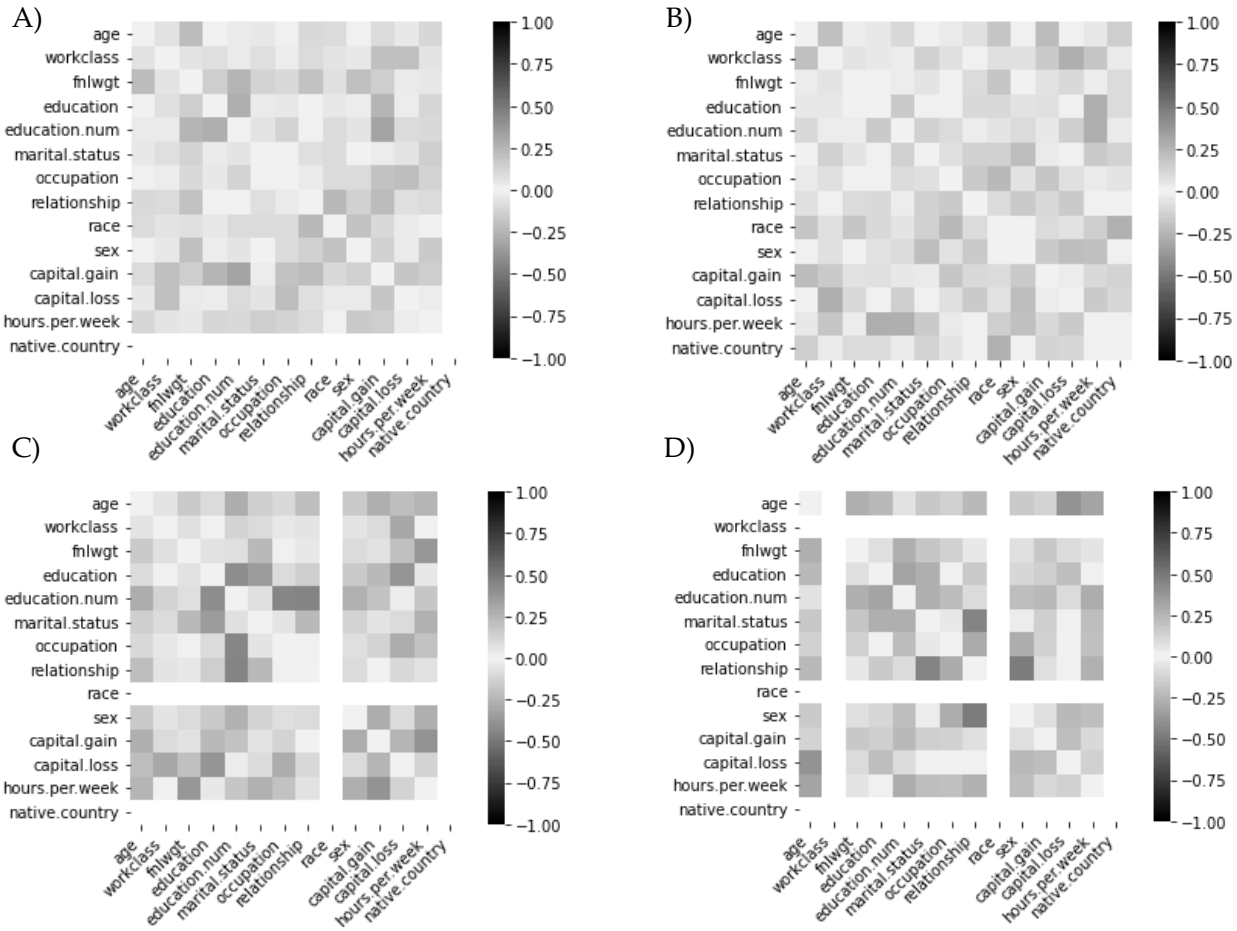

Figure 9 - Heatmaps of the difference between correlation values present in the original (Torig) dataset and the synthetic dataset (Tsyn) of the Census dataset. A) Heatmap of the 1:1 setup. B) Heatmap of the 1:2 setup. C) Heatmap of the 1:3 setup. D) Heatmap of the 1:4 setup.

In Figures 11 a) and 11 b) the histograms of the highly heterogeneous features "pdays" and "poutcome" are displayed, together with the histogram of the synthesized samples for each feature, and Figures 11 c) and 11 d), show the difference in the frequencies of each value, where bars point up are indicative that the generative model produced with more frequency samples containing that value (compared to the original distribution), and the bars pointing down indicate the opposite, i.e., there were fewer samples containing that value in the Tsyn that the ones present in Torig. As we can see from the position of the bars in Figures 11 c) and 11 d), compared to the Torig, the generative model produced more samples containing the most represented values, and fewer containing the less represented values, which may have compromised the relationship integrity of these features.

In Figure 8, we can also see a straight line for the "default" feature, due to the fact that this feature has only 1 value, which our model easily captured, resulting in a difference in

correlation values equal to 0 as both original and synthetic correlation values for pairs of features containing the "default" feature were the same (0).

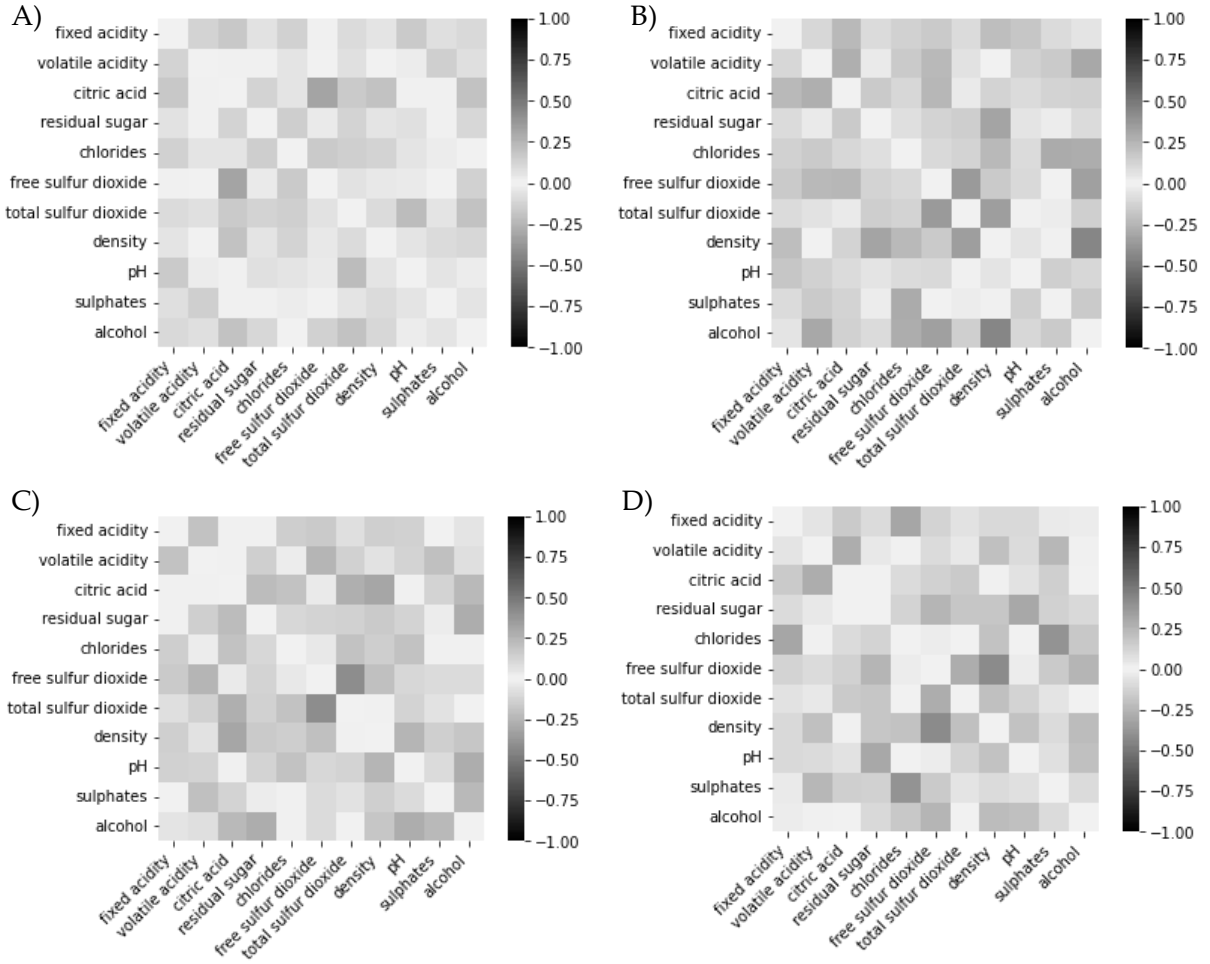

Figure 10 - Heatmaps of the difference between correlation values present in the original (Torig) dataset and the synthetic dataset (Tsyn) of the Wine dataset. A) Heatmap of the 1:1 setup. B) Heatmap of the 1:2 setup. C) Heatmap of the 1:3 setup. D) Heatmap of the 1:4 setup.

As a complementary study, we challenged the generative model to synthetically generate new samples from datasets decreasing in size as such we plotted the same correlation difference matrices and what we observed was that, as expected, the model began to have more difficulty in mimicking the relationships present in the original data, which was observable by the increase in darker cells in the matrix, meaning that the differences shifted from values closer to 0 to values closer to 1. As discussed previously, the generative model acts almost like a filter, reducing the noise caused by less frequent values, often called extreme values or outliers, synthesizing more of the modal values. As the dataset sizes become smaller, an even smaller slice of these extreme values is represented in the original datasets, and so the model produces even fewer quantities of these values. The absence of these extreme values or outliers caused the synthetic feature distribution to become different from the original distribution causing the feature relationships to also become different.

Moreover, we observed the appearance of straight white lines in the Marketing and Census datasets as they became more and more reduced, a consequence of the downsizing process since it led to datasets containing more features with only one value. The only dataset where we did not observe this phenomenon was the Wine dataset, as it is mostly composed of

continuous features. With these types of features, it is not easy to have only one exact value. Nevertheless, this phenomenon does not represent a problem or a limitation to our conclusions, it is only a consequence of the downsampling process and it does not give any information about the quality of the generative models.

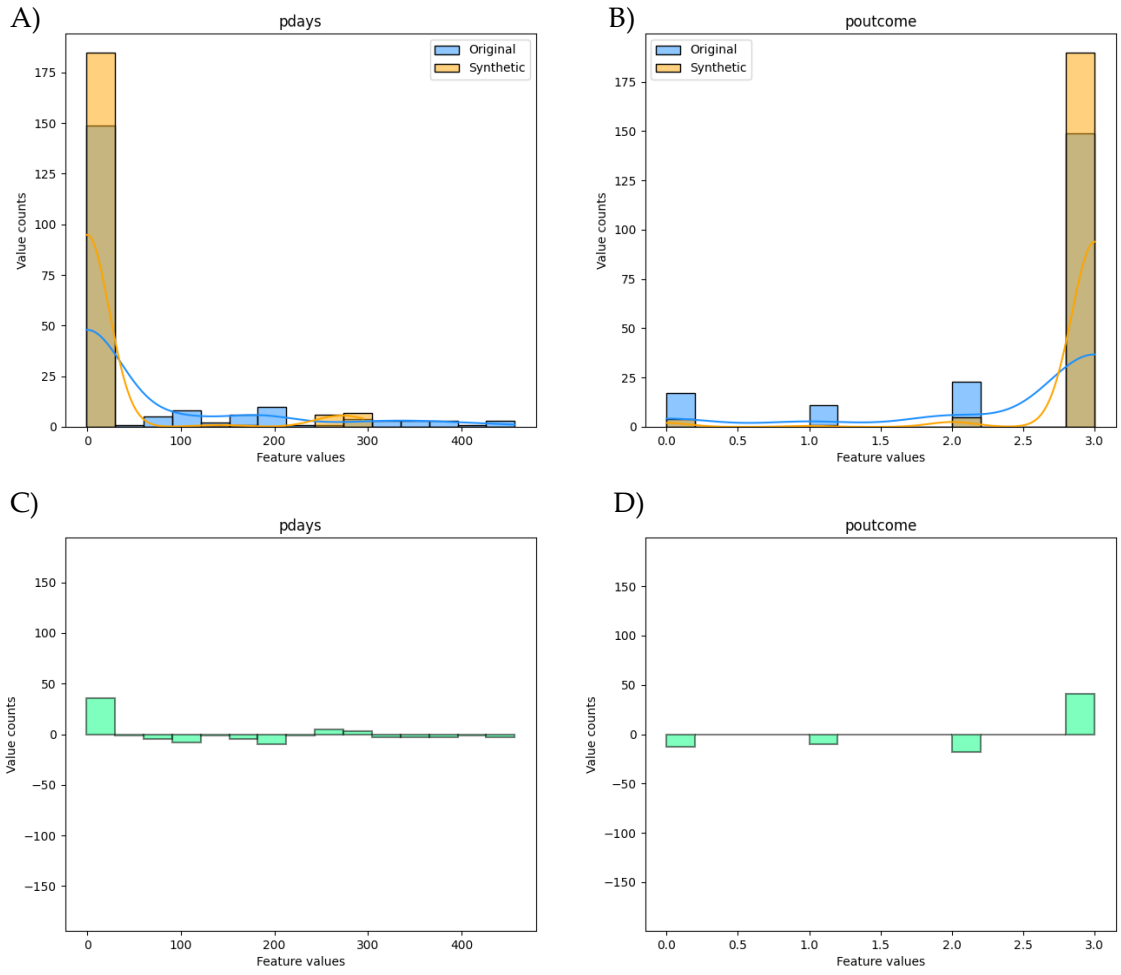

Figure 11 - Histograms of the Marketing dataset features "pdays" and "poutcome" together with the histogram of the synthesized samples for each feature, and barplots of difference in the frequencies of each value (between the original and the synthetic dataset. A) Feature distributions of the “pdays” feature. B) Feature distributions of the “poutcome” feature. C) Feature frequency difference of the “pdays” feature. D) Feature frequency difference of the “poutcome” feature.

After assessing the integrity between the features in all pairs of Tsyn and Torig, the datasets were subjected to the assessment of their similarity. The similarity results show to what degree the generative models could replicate the distribution of all the features present in the original dataset through statistical tests.

The results showed a tendency to obtain higher similarities when the composition in categorical and discrete non-continuous features was smaller, like in the Wine dataset case. This dataset is almost exclusively composed of continuous features, which results in a much more satisfactory approximation when compared to the Marketing and Census datasets, that do not have any continuous feature. Table 4 summarizes the results obtained for each dataset in each small data size setup. With these results, it becomes clear that the best similarity

values are ensured by the Wine datasets, with similarities between 58% to 92% while the Marketing and Census datasets only achieve similarity results of 35% to 41%.

Table 4 – Similarity values for the Marketing, Census and Wine datasets according to each data size setup (1:1, 1:2, 1:3, 1:4).

	1:1	1:2	1:3	1:4
Marketing	41%	35%	35%	35%
Census	40%	33%	33%	33%
Wine	73%	**92%**	83%	**58%**

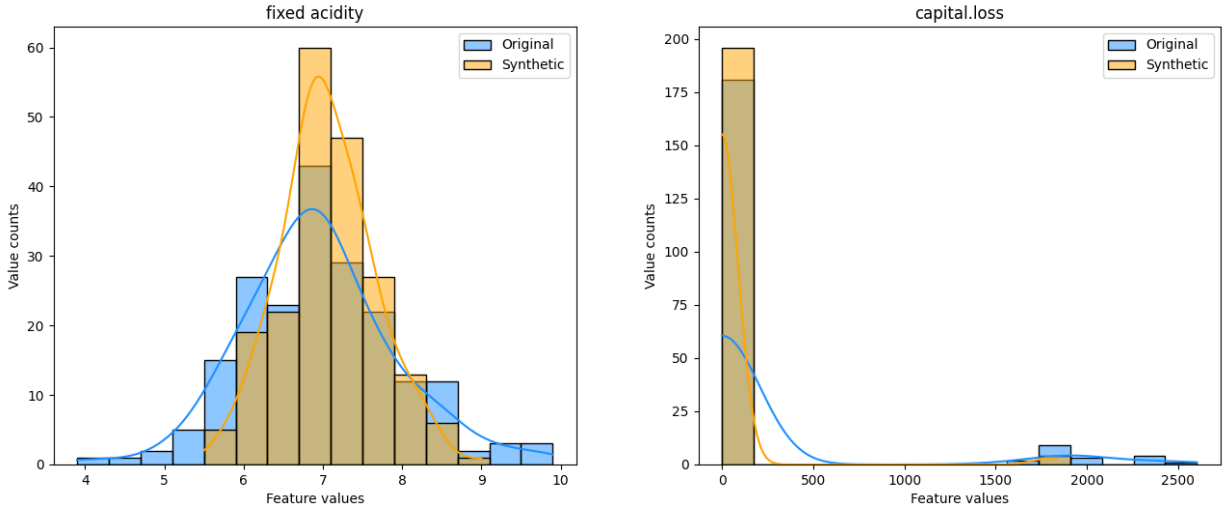

Figure 12 - Histograms of the Wine dataset feature "fixed acidity" and the Census dataset feature "capital.loss" together with the histogram of the synthesized samples for each feature.

The similarities obtained align with what we previously saw when accessing the relationship integrity. The low pair-wise similarity are observable in the same type of features that failed to accurately mimic the original relationships, i.e., features with a high discrepancy in frequency values. The features with a high number of occurrences for one specific value, or a small set of values, and a low number of occurrences for the rest are features with probability distributions that present steeper bell curves and smaller standard deviations since generative model have a filter-like behavior, producing more samples with values within the neighborhoods of the modes, and less of the extreme values, it reduces the standard deviation and steepens, even more, the distribution curves. In other words, it changes the feature distributions. Since the distributions are no longer the same, the pair wise similarity between the features becomes 0 and the total similarity is jeopardized. Figure 12 the histograms and the probability density curves of Torig and Tsyn in the 1:1 setup are shown for the "fixed acidity" feature of the Wine dataset. The distributions of the "fixed acidity" feature for Tsyn and Torig contrast with the distributions of the "capital.loss" feature of the Census dataset by displaying an original distribution curve much wider, representing a higher standard deviation. Additionally, the differences between distribution shapes between Torig and Tsyn are much lower, resulting in a pair-wise similarity of 1 for the "fixed acidity" feature, and 0 for the other two features.

Furthermore, we also saw that the generative models could better approximate the distributions of the numeric non-continuous features than the distributions of the categorical

features, as the pair-wise similarities for these types of features were between 43% to 67% while the categorical features could only achieve 11% to 40%. As the Marketing and Census datasets were both more plentiful in categorical features than in numeric non-continuous features, the total similarity values were impacted by the low pair-wise similarities obtained with categorical features.

As a sanity test, we measured the number of duplicate samples. In the synthetic dataset we observed that Tsyn had no duplicated samples while, some of the Torig did have some. This excludes the hypothesis that the model could have been "cheating", by just reproducing the Torig, also known as mode-collapsing.

Based on the statistical results, we selected two models that we considered to be promising candidates to proceed to the next phase of validation, the utility validation of the synthetic data. The first model we chose was the one that achieved the best results overall, i.e., the model which demonstrated higher relationship integrity and achieved higher similarity results in comparison with all the models in any data size setup. The second model we chose was the one that performed the best in the lowest data size setups. The model which achieved the best performance overall was named M1 and the model which achieved the best performance in the lowest data size setups was named M2.

The selected models were both models tuned for the Wine dataset but in setups of different data sizes. The M1 model was fine-tunned for the 1:2 setup (Torig with 100 samples) while the M2 model was fine-tunned for the 1:4 setup (Torig with 50 samples).

We started the utility assessment by evaluating the utility of M1 in a balanced Tsyn of 100 samples and the utility of M2 in a balanced Tsyn of 50 samples. Tables 5 and 6 show the results obtained with each classifier in means of accuracy, precision, recall, F1-score and AUC with respect to the M1 Torig and Tsyn datasets. The tables 7 and 8, show the values of those same metrics and respective classifiers for the M2 Torig and Tsyn. The performances obtained varied depending on the classifier being used and the quality metric being accessed. However, when comparing the Torig results with the Tsyn results for each model, it became clear that these models were able to learn from the original datasets and reproduce in the synthetic samples the knowledge they acquired from the original datasets since the classifiers were able to obtain performance values in the Tsyn similar to those obtained in Torig.

Table 5 - Torig utility values obtained in the Wine dataset in the setup 1:2 (100 samples).

	accuracy	precision	recall	F1-score	AUC
XGBoost	0.24	0.2144	0.24	0.2197	0.6275
Logistic Regression	0.3	0.3049	0.3	0.2742	0.6713
SVM	0.3167	0.3015	0.3167	0.2845	0.685
Decision Tree	0.3025	0.286	0.3025	0.2715	0.6482
MLP	0.312	0.3141	0.312	0.2853	0.6614
KNeighbors	0.3084	0.3092	0.3084	0.2833	0.6643

Table 6 - Tsyn utility values obtained with M1 for the Wine dataset in the setup 1:2 (100 samples).

	accuracy	precision	recall	F1-score	AUC
XGBoost	0.29	0.3034	0.29	0.2521	0.5419

Logistic Regression	0.3	0.2933	0.3	0.2521	0.6016
SVM	0.2934	0.2781	0.2934	0.2389	0.5546
Decision Tree	0.285	0.2876	0.285	0.2398	0.5504
MLP	0.3	0.3055	0.3	0.2536	0.5762
KNeighbors	0.2867	0.2927	0.2867	0.2464	0.5746

Table 7 - Torig utility values obtained in the Wine dataset in the setup 1:4 (50 samples).

	accuracy	precision	recall	F1-score	AUC
XGBoost	0.32	0.3334	0.32	0.304	0.6275
Logistic Regression	0.31	0.262	0.31	0.2659	0.645
SVM	0.3334	0.2783	0.3334	0.2865	0.5525
Decision Tree	0.335	0.2945	0.335	0.2959	0.5613
MLP	0.324	0.2831	0.324	0.2844	0.5825
KNeighbors	0.3134	0.2776	0.3134	0.2772	0.5886

Table 8 - Tsyn utility values obtained with M2 for the Wine dataset in the setup 1:4 (50 samples).

	accuracy	precision	recall	F1-score	AUC
XGBoost	0.2	0.1235	0.2	0.1362	0.52
Logistic Regression	0.26	0.1628	0.26	0.189	0.55
SVM	0.2534	0.1885	0.2534	0.1998	0.5334
Decision Tree	0.235	0.1717	0.235	0.1844	0.5219
MLP	0.256	0.1774	0.256	0.1963	0.5405
KNeighbors	0.25	0.1862	0.25	0.1982	0.5428

Particularly, with the M1 model, the classifiers were able to achieve performances slightly better in Tsyn than in Torig, which may be due to the reduction in the frequency of extreme values that we saw happening with these generative models. By reducing the noise, the classifiers were able to learn more easily the patterns inherent to the data and therefore predict more easily the label, resulting in higher performance values than those obtained with the real data. To highlight these differences, Table 9 display the results of subtracting the performance values of Torig to Tsyn of M1 and the Table 10 displays these results for Torig and Tsyn of M2. This way, negative values show that the performance with Tsyn was worse than with Torig and positive values show that the performance was better with Tsyn than with Torig. Averaging those results showed that with M1 we were able to obtain positive values for the differences between Tsyn and Torig, such as 0.0051 for precision, which meant that comparing to the precision values obtained in Torig, the precision values were improved, on average, by 4% of its true value (0.2884). Other performance metrics were not improved beyond its original values, as their differences were negative, but they still presented really good results, like accuracy and recall that achieved each 99% of the original value, f1-score, which achieved a difference of -0.005, or in other words, 92% of its original

f1-score value and AUC which was the most affected metric, achieving only 86% of its original value but which is still an impressive result.

Table 9 - Difference values between Tsyn and Torig s from the 1:2 setup obtained with M1.

	accuracy	precision	recall	F1-score	AUC
XGBoost	0.05	0.089	0.05	0.0324	-0.0856
Logistic Regression	0	-0.0116	0	-0.0221	-0.0697
SVM	-0.0233	-0.0234	-0.0233	-0.0456	-0.1304
Decision Tree	-0.0175	0.0016	-0.0175	-0.0317	-0.0978
MLP	-0.012	-0.0086	-0.012	-0.0317	-0.0852
KNeighbors	-0.0217	-0.0165	-0.0217	-0.0369	-0.0897

Table 10 - Difference values between Tsyn and Torig s from the 1:4 setup obtained with M2.

	accuracy	precision	recall	F1-score	AUC
XGBoost	-0.04	-0.0764	-0.04	-0.0742	-0.0825
Logistic Regression	-0.06	-0.0554	-0.06	-0.0717	-0.0487
SVM	-0.08	-0.0797	-0.08	-0.0872	-0.0491
Decision Tree	-0.08	-0.0858	-0.08	-0.0903	-0.0494
MLP	-0.076	-0.086	-0.076	-0.0871	-0.045
KNeighbors	-0.06	-0.0738	-0.06	-0.073	-0.0352

While with M1 some of the values obatained with Tsyn were slightly higher that those obtained with Torig, with M2 the average differences (Table 10) were all negative, since the performance values achieved in Tsyn were always inferior to those obtained in Torig. In addition, the differences, as we can see in Table 10, were with the exception of the AUC more negative differences than those that resulted from M1, i.e., the M1 was able to generate synthetic datasets that achieved higher utility than M2. As the M1 model was trained only on 50 samples, he model had to learn from only 10 samples of each one of the 5 classes, challenging the model much more. Nevertheless, the model was still able to achieve impressive results such as 75% of the original average accuracy and recall (0.3226), 90% of the original average AUC (0.5929) and 65% of the original average f1-score (0.2857).

The utility results show the potential of these generative models and the impact of extreme downsampling however, more conclusions could be drawn about M1 and M2 by evaluating the utility obtained by synthetizing samples for setups the generative models were not fine tunned for. For that reason, each model, M1 and M2, synthetized for every data size setup of the Wine dataset a Tsyn that was then subjected to utility evaluation and then models were compared.

The tables 11, 12, 13, and 14 show for each one of the 4 data size setups the average differences for each quality metric achieved with the M1 and M2 models. The tables 15, 16, 17, and 18, show the percentage of the original value each model was able to achieve on each metric

and setup. The results showed that for the setup 1:2, the model M1 achieved higher results than the model M2 for that same setup, as was expected since the model M1 was fine-tuned for this particular setup, however, the M2 model obtained very competitive results, in the 90%-97% range of utility.

Table 11 - Average differences in classifier performances between Tsyn and Torig for the 1:1 setup.

	accuracy	precision	recall	f1-score	AUC
M1	-0.0965	-0.0839	-0.0965	-0.0964	-0.0733
M2	-0.0708	-0.0716	-0.0708	-0.0786	-0.0454

Table 12 - Average differences in classifier performances between Tsyn and Torig for the 1:2 setup.

	accuracy	precision	recall	f1-score	AUC
M1	-0.0041	0.0051	-0.0041	-0.0226	-0.0931
M2	-0.0087	-0.0147	-0.0087	-0.0232	-0.0666

Table 13 - Average differences in classifier performances between Tsyn and Torig for the 1:3 setup.

	accuracy	precision	recall	f1-score	AUC
M1	-0.0236	-0.02	-0.0236	-0.0293	-0.0492
M2	-0.0427	-0.0862	-0.0427	-0.0747	-0.0602

Table 14 - Average differences in classifier performances between Tsyn and Torig for the 1:4 setup.

	accuracy	precision	recall	f1-score	AUC
M1	-0.066	-0.0762	-0.066	-0.0806	-0.0517
M2	-0.0803	-0.1198	-0.0803	-0.1017	-0.0582

For the 1:3 setup, the model M1 produced again really good results and was the best performing model, as expected since the model M1 was trained in a larger setup benefiting from the knowledge of more data to synthetically produce the desired Tsyns. However, despite the utility of M2 being lower than M1, it did not differ so much from the M1 utility. These results show that even though neither of the models was tailored for this dataset, the two models were able to produce satisfactory utility results. The M1 achieved results from 91% to 95% of the original performance values and the M2 from 76% to 91% which shows that these are robust models, especially M2 which was trained in a smaller number of samples than the ones it had to generate but still provided similar results to the M1 model.

For the 1:4 setup, we see in Table 18 a similar behaviour to what was observed in the 1:3 setup. The model M1 clearly benefited from the fact that it saw more data in the training of the generative model however, the M2 model offered again very competitive results, not differing too much from those obtained by M1.

Regarding the 1:1 setup, both models had to generate a Tsyn of sample size much larger that the Torig on which they were trained. The M1 model, not benefiting anymore from training in a larger Torig than the Tsyn, had to synthesize twice as many samples it had seen during training and the M2 model had to generate 4 times more samples than what it was fine-tunned and trained for, yet, both models were able to achieve very satisfactory values. The M1 model achieved performance values of 70% to 93% of the original performance values

and the model M2 achieved performance values of 76% to 93% which was very impressive and highlighted the robustness of models, particularly for the M2 model.

By accessing the utility in terms of comparisons between the original performances and the synthetic performances, the true values of performance were not relevant to evaluating the generative power of the models. Noisy data exists in the real world and so it is not expectable that the synthetic data modeled from original data that is difficult to classify becomes much more easily classifiable, while maintaining the same properties as the original data. This validation process was very informative and complemented well the results we obtained in the previous statistical data validation steps.

Table 15 – Percentage of utility that each model (M1 and M2) achieved compared with the original utility (on average) for the 1:1 setup.

	accuracy	precision	recall	f1-score	AUC
M1	73%	76%	73%	70%	89%
M2	80%	79%	80%	76%	93%
Real value	0.3586	0.3486	0.3586	0.3261	0.6892

Table 16 - Percentage of utility that each model (M1 and M2) achieved compared with the original utility (on average) for the 1:2 setup.

	accuracy	precision	recall	f1-score	AUC
M1	99%	104%	99%	92%	86%
M2	97%	97%	97%	92%	90%
Real value	0.2966	0.2884	0.2966	0.2698	0.6597

Table 17 - Percentage of utility that each model (M1 and M2) achieved compared with the original utility (on average) for the 1:3 setup.

	accuracy	precision	recall	f1-score	AUC
M1	94%	95%	94%	91%	93%
M2	88%	76%	88%	77%	91%
Real value	0.3414	0.355	0.3414	0.3222	0.6967

Table 18 - Percentage of utility that each model (M1 and M2) achieved compared with the original utility (on average) for the 1:4 setup.

	accuracy	precision	recall	f1-score	AUC
M1	80%	74%	80%	72%	91%
M2	75%	59%	75%	65%	90%
Real value	0.3227	0.2882	0.3227	0.2857	0.5929

4.2 Biological Data Generation

To test if the results we obtained with the generic data were reproducible in real-world biological data, the two models M1 and M2 were used to generate synthetic datasets from the two biological datasets Mitostress and ATP Rate. The results are presented next under the scope of the same statistical and utility validation methodologies.

For the Mitostress dataset, the heatmaps of the correlation differences showed for M1 and M2 satisfactory results, with most cells being in light grey colors, indicating that the differences in correlation were within the 0.00 to 0.25 range in absolute value, which show that both models were able to mimic the relationships found between the features of the original dataset (Figures 13 a) and 13 b)). Additionally, in the heatmap of the difference values obtained with M1 in Figure 13 a), we can find cells with a much lighter tonality, meaning that those relationships were mimicked almost perfectly.

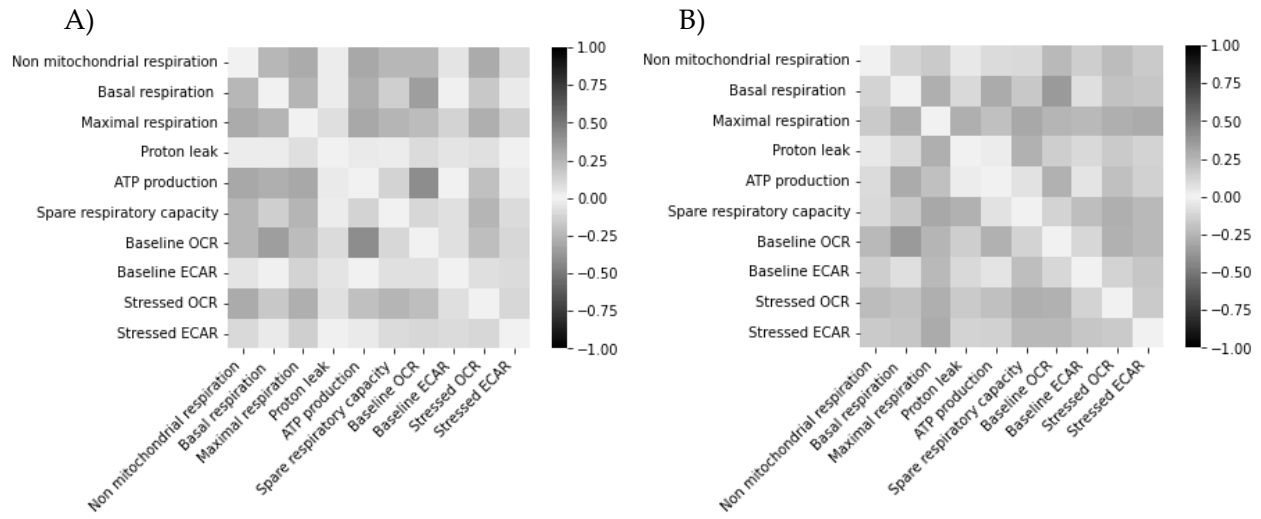

Figure 13 - Heatmaps of the difference between correlation values present in the original (Torig) dataset and the synthetic dataset (Tsyn) of the Mitostress dataset. A) Heatmap of the values obtained with M1. B) Heatmap of the values obtained with M2.

For the ATP dataset, similar results were achieved where the model M1 and the model M2 produced synthetic datasets that preserved the same relationships that were seen in the features of the original dataset, represented by the light color of the cells constituting the matrices in Figures 14 a) and 14 b). Similarly to what we witnessed in the Mitostress dataset, the differences obtained with the M2 were slightly higher than those verified with M1. However, those differences are too small to distinguish, based on that alone, if one model was better than the other.

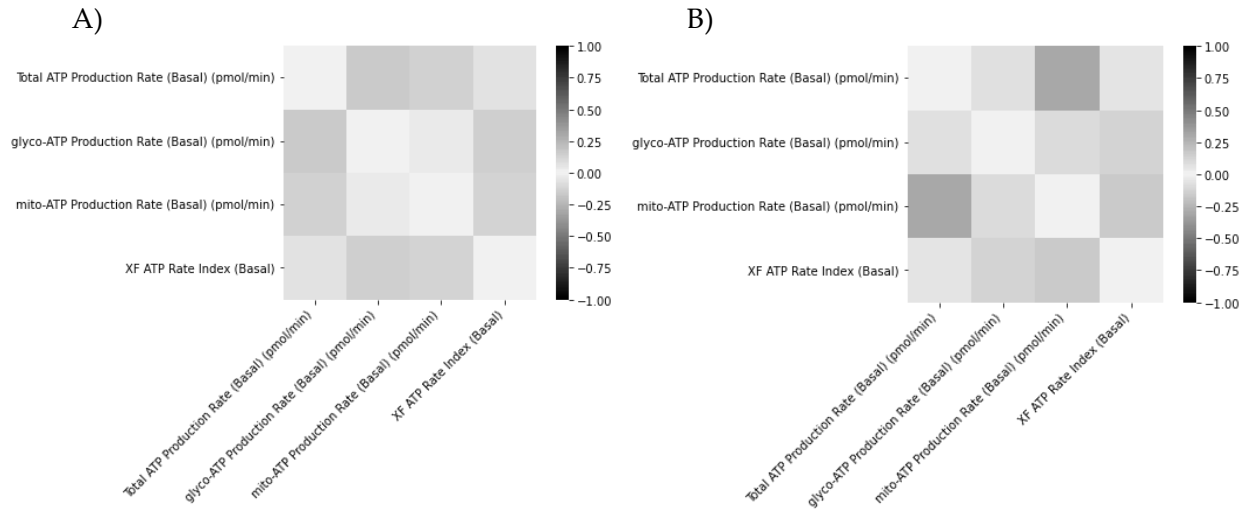

Figure 14 - Heatmaps of the difference between correlation values present in the original (Torig) dataset and the synthetic dataset (Tsyn) of the Mitostress dataset. A) Heatmap of the values obtained with M1. B) Heatmap of the values obtained with M2.

After computing the pair-wise similarities for all the features in the Mitostress dataset for model M1 and model M2, we were able to draw some more conclusions about these models. The results we obtained are summarised in Table 19 and are in line with what we saw in the heatmaps of the features correlations. Both models M1 and M2 were able to generate samples that were similar to real samples, presenting the same feature distributions and resulting in similarities of 100% for M2 and of 92% for M1. In the M1 case, the similarity was 8% less because the feature "Non mitochondrial respiration" did not follow the same probability distribution. This led to a pair-wise similarity of 0, which despite jeopardizing the total similarity of Tsyn, aligns with Figure 13, seen previously, since most of the cells involving this feature display a darker colour than most cells. Once again, the Tsyn(s) generated did not contain diuplicate samples, proving our models were not only memorizing their inputs.

The same models M1 and M2 were also evaluated in terms of similarity for the ATP Rate dataset (Table 19), and the results showed that both models achieved 100% of similarity, meaning that the synthetic datasets Tsyn generated from M1 and M2 contained synthetic samples that could have been sampled from the original dataset, as all synthetic features exhibited the same probability distributions as the original features.

Table 19 - Similarity values for the Mitostress and ATP Rate datasets obtained with M1 and M2.

	M1	M2
Mitostress	92%	100%
ATP Rate	100%	100%

The results achieved by M1 and M2 were impressive, especially considering that none of the models was fine-tuned for any of the models, demonstrating their robustness.

In the utility data validation step both M1 and M2 models were evaluated again for the Mitostress and ATP Rate datasets.

The results showed that for the Mitostress dataset both models achieved good results, with minimal reduction in performance of the classifiers in Tsyn when compared to Torig. In fact, some of the differences were positive, which meant that for some quality metrics the classifiers were able to achieve higher performances when learning from Tsyn rather than doing so from Torig. The best utility results were obtained with M2, with performance values ranging from 97% to 104% of the original average values, and with the M1 the classifiers obtained performance values just slightly lower, resulting in values that ranged from 88% to 103% of the original values.

The results observed in the ATP Rate dataset were similar to what we saw with the Mitostress dataset as well. The M1 and M2 models were able to guarantee with success the utility of the synthetic data. With the M2 model, the performance values obtained ranged from 97% to 101%, and with the model M1 it ranged from 83% to 102% which again were values slightly inferior to those obtained with M2, but still very impressive. These utility values for the Mitostress dataset are displayed in Table 20 and for the ATP Rate dataset are displayed in Table 21.

Table 20 - Percentage of utility that each model (M1 and M2) achieved compared with the original utility (on average) for the Mitostress dataset.

	accuracy	precision	recall	f1-score	AUC
M1	92%	98%	92%	88%	103%
M2	99%	103%	99%	97%	104%
Real value	0.5699	0.588	0.57	0.559	0.733

Table 21 - Percentage of utility that each model (M1 and M2) achieved compared with the original utility (on average) for the ATP Rate dataset.

	accuracy	precision	recall	f1-score	AUC
M1	90%	92%	90%	83%	102%
M2	97%	99%	97%	95%	101%
Real value	0.5585	0.587	0.559	0.553	0.694

The ability of the models to provide full datasets of synthetic samples that maintained the utility of the data, and in some cases even improved it, from real and unseen biological datasets showcased the robustness of these models and how they can be applied to other datasets to do the same.

4.3 Discussion

Our methodology, comprised of 2 stages, was able to give its users a holistic view of a great number of generic models, providing at each step important clues for what the best models would be to solve our problem. The results of each individual step alone did not provide an answer to the generative power of each model but the integration of all those procedures in

a sequential way allowed us to make an informed decision leading to high-performing models, as was shown by the results achieved in real biological data.

Regarding the generative approach chosen for this work - the TVAE - the results showed a good capacity in generating generic tabular data as well as biological data, as it was shown by the achievement of the high levels of conservation in the original statistical properties of the data – relationship integrity and similarity - and by the high values of utility achieved in the synthetic data, while also avoiding duplicates. However, some datasets were easier to recreate than others.

When generating synthetic data containing only categorical and discrete features the generative models struggled in maintaining the same properties of the original data. The results showed that as these are features with a great imbalance in the frequency distribution of values, with one specific value having a much higher number of samples than the remaining values, the models acted almost like a filter and treated those less frequent values as noise, decreasing, even more, their frequency in the synthetic samples. By doing that, the relationship integrity between features was weakened and the similarity values were low since the probability distributions of those synthetic features were not equal to the probability distributions of the original features. To counter this and to guarantee that these features would maintain the same frequency values as the original data, other measures would need to be put into practice such as introducing conditions to the sampling process. By introducing conditions during sampling, we could choose to keep or exclude the synthetic samples according to the original feature frequency values, which would be more time-consuming, but would also be a trade-off that would likely be worth exploring, given that as we are dealing with a small number of training samples, the experiments are currently relatively fast. However, another thing worth noting, is that it is not clear to what degree this filtering behaviour might limit the generative models.

In the Wine dataset, which contained only continuous features excluding the label, the downsampling of the datasets increased the difficulty of the generation of samples that maintained the original statistical properties, since the models had less samples to learn from. However, the utility validation of the data showed that even the setup with the least number of samples (50), and consequently the lower relationship integrity and similarity values (58%) exhibited high utility and when applied to biological datasets. The models were able to achieve not only high utility values as well as similarity values of 100% in both Mitostress and ATP Rate dataset. To further explore these results another study about the relationship of these two validation techniques would be necessary to know how much of statistical divergencies are too much to compromise the next stages of our framework.

With respect to the utility, our results showed that M1 and M2 behaved in very similar ways, achieving very satisfactory results both for the generic dataset Wine as well as for the biological datasets Mitostress and ATP Rate. Additionally, the utility results for model M1 on the setup 1:2 were higher than 100%, which meant an increase in performance of the classifiers trained in Tsyn, compared to those trained in Torig by 4%. This phenomenon may be related to the behaviour encountered previously, in which the model filters out the less frequent values, since those values may be the ones causing the classifiers to have smaller performance values.

The experimental design, which included the complementary study of the models under extreme situations of small sizing, was essential to study the models' behaviour with the decrease in samples. The results showed that with the decreasing in data size it became more difficult to maintain the statistical properties of the original data, however these changes

were not proportional to downsampling that we provoked and even in the extreme situation of having only 50 samples, the results observed were practically identical to the results observed in the setups containing more samples. Moreover, it allowed us to infer about e robustness of M1 and M2 models, proving that they be very robust, specially the M2 model which achieved utility values between 88% and 91% for Tsyn(s) with up to 4 times more data samples than the amount used to train the model. Furthermore, the incorporation strategies like of cross validation and early stopping in our experimental design were also pivotal to ensure the viability and reproducibility of these results.

Chapter 5
Conclusion

Biological systems are complex and can benefit from Machine Learning, however, in the biological domain the number of samples is typically small, increasing bias in the conclusions and jeopardizing the reproducibility of the results.

In this work we presented a framework that integrates a Variational Autoencoder (VAE) in a generative pipeline for the synthetic augmentation of biological data. As this is not a trivial task, our pipeline encompassed 2 main stages: a first stage to select the best possible generative models, and a second stage, where the models generate biological data. To further challenge the models' ability to generalize in small data situations, we conducted a complementary study where the models were forced to learn from progressively smaller datasets.

The experimental study conducted show that generative models were able to capture the statistical properties of the original data, mimicking in the synthetic data the original relationships and distributions. These models were capable of achieving good results such as similarities of 92% for datasets containing only 100 samples and even for the smallest datasets, which contained only 50 samples, the synthetic datasets displayed similarity values of 58%.

The statistical properties allowed us to anticipate the synthetic data utility, which was crucial to select the most promising models. The utility was then accessed on the best performing model overall (M1) and the best performing model in the smallest data size setup (M2) by means of 6 classifiers which had their original performance values compared to the performances achieved with synthetic data. The utility results showcased the robustness of the models, which reached average performances up to 104% of the original values and even when tested for datasets with 2 and 4 times more samples than the models' training samples the results were still impressive, changing very little.

When applied to the biological datasets the models achieved a good performance. In terms of the statistical validation, the models were able to generate synthetic datasets that maintained the correlations between features almost perfectly and achieved values of similarity within the range of 92% to 100% for the Mitostress dataset and 100% in the ATP Rate dataset. The utility validation also yielded very satisfactory results, as the average classification performances of the generative models were much close to the original values, even surpassing them in by as much as 4% in some cases.

The results showed the capacity of these models to generate generic tabular data as well as biological data. Despite some models performing better than others, for the most part these models achieved high levels of conservation of the original statistical properties – relationship integrity and similarity – and revealed high values of utility while also avoiding duplicates. Hence, with our experimental results we can conclude that our approach can successfully contribute to the synthetic augmentation of biological datasets, overcoming the limitations of small data.

These results were presented at the 56th Annual Scientific Meeting of the European Society for Clinical Investigation held in Bari, Italy, between 8th-10th of June 2022.

Future work

Although these models proved to be able to generate synthetic datasets that achieved impressive results, our framework lacked a higher diversity of biological datasets. As we want our finding to be the most useful to the biological domain as possible, in the future would like to incorporate more biological datasets into our framework.

Additionally, we concluded that the generation of datasets containing categorical features and features with high disparities in the frequency of values represented a weakness, as the models exhibited more restrains in preserving the statistical properties of the datasets containing these data types in high proportions. However, it is not clear to what degree one can overlook the loss in original statistical properties without compromising the next stages of the pipeline nor what measures can be put in practice to mitigate those limitations. Thereupon, in the feature we hope to explore these questions more in detail bringing more confidence to our framework.

References

1. Cobb M (2017) 60 years ago, Francis Crick changed the logic of biology. PLoS Biol 15:. https://doi.org/10.1371/JOURNAL.PBIO.2003243
2. Dixit P, Prajapati GI (2015) Machine learning in bioinformatics: A novel approach for DNA sequencing. Int Conf Adv Comput Commun Technol ACCT 2015-April:41–47. https://doi.org/10.1109/ACCT.2015.73
3. Sarker IH (2021) Machine Learning: Algorithms, Real-World Applications and Research Directions. SN Comput Sci 2:1–21. https://doi.org/10.1007/S42979-021-00592-X/FIGURES/11
4. Zhang S, Yao L, Sun A, Tay Y (2017) Deep Learning based Recommender System: A Survey and New Perspectives. ACM Comput Surv 52:. https://doi.org/10.1145/3285029
5. Dong H-W, Hsiao W-Y, Yang L-C, Yang Y-H MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment
6. Yang S, Wang Y, Chu X (2020) A Survey of Deep Learning Techniques for Neural Machine Translation. https://doi.org/10.48550/arxiv.2002.07526
7. Elallid B Ben, Benamar N, Hafid AS, et al (2022) A Comprehensive Survey on the Application of Deep and Reinforcement Learning Approaches in Autonomous Driving. J King Saud Univ - Comput Inf Sci. https://doi.org/10.1016/J.JKSUCI.2022.03.013
8. Baker M (2015) Irreproducible biology research costs put at $28 billion per year. Nature. https://doi.org/10.1038/NATURE.2015.17711
9. Pereira S (2013) Motivations and Barriers to Sharing Biological Samples: A Case Study. J Pers Med 3:102. https://doi.org/10.3390/JPM3020102
10. Six factors affecting reproducibility in life science research and how to handle them. https://www.nature.com/articles/d42473-019-00004-y. Accessed 24 Jan 2022
11. Rai J, Kaushik K (2018) Reduction of Animal Sacrifice in Biomedical Science & Research through Alternative Design of Animal Experiments. Saudi Pharm J 26:896–902. https://doi.org/10.1016/J.JSPS.2018.03.006
12. Ross KA (2009) Curse of Dimensionality. Encycl Database Syst 545–546. https://doi.org/10.1007/978-0-387-39940-9_133
13. Nguyen TT, Viet Q, Nguyen H, et al Deep Learning for Deepfakes Creation and Detection: A Survey
14. Deshpande A, Lu J, Yeh MC, et al (2016) Learning Diverse Image Colorization. Proc - 30th IEEE Conf Comput Vis Pattern Recognition, CVPR 2017 2017-January:2877–2885. https://doi.org/10.48550/arxiv.1612.01958
15. Farnoosh A, Rezaei B, Ostadabbas S DEEPPBM: DEEP PROBABILISTIC BACKGROUND MODEL ESTIMATION FROM VIDEO SEQUENCES
16. Clarence Chio & David Freeman (2017) Machine learning and Security. Mach Learn 45:40–48
17. Kolata G (1982) How can computers get common sense? Science 217:1237–1238.

https://doi.org/10.1126/SCIENCE.217.4566.1237

18. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 2016-December:770–778. https://doi.org/10.1109/CVPR.2016.90
19. Ol G Er Schwenk H (2012) Continuous Space Translation Models for Phrase-Based Statistical Machine Translation. 1071–1080
20. Nguyen TT, Viet Q, Nguyen H, et al (2019) Deep Learning for Deepfakes Creation and Detection: A Survey
21. Michelucci U (2022) An Introduction to Autoencoders
22. Kingma DP, Welling M (2019) An Introduction to Variational Autoencoders. Found Trends Mach Learn 12:307–392. https://doi.org/10.1561/2200000056
23. Wooley JC, Lin HS, Biology NRC (US) C on F at the I of C and (2005) On the Nature of Biological Data
24. Somani S, Russak AJ, Richter F, et al (2021) Deep learning and the electrocardiogram: review of the current state-of-the-art. Europace 23:1179–1191. https://doi.org/10.1093/EUROPACE/EUAA377
25. Clements JM, Xu D, Yousefi N, Efimov D (2020) Sequential Deep Learning for Credit Risk Monitoring with Tabular Financial Data
26. Zhang S, Yao L, Sun A, Tay Y (2017) Deep Learning based Recommender System: A Survey and New Perspectives. ACM Comput Surv 52:. https://doi.org/10.1145/3285029
27. Buczak AL, Guven E (2016) A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. undefined 18:1153–1176. https://doi.org/10.1109/COMST.2015.2494502
28. Urban C, Gates K Deep Learning: A Primer for Psychologists. https://doi.org/10.31234/OSF.IO/4Q8NA
29. Huang X, Khetan A, Cvitkovic M, Karnin Z (2020) TabTransformer: Tabular Data Modeling Using Contextual Embeddings
30. Yoon J, Zhang Y, Jordon J, Van Der Schaar M VIME: Extending the Success of Self-and Semi-supervised Learning to Tabular Domain
31. Khan S, Naseer M, Hayat M, et al (2021) Transformers in Vision: A Survey. https://doi.org/10.1145/3505244
32. Abutbul A, Elidan G, Katzir L, El-Yaniv R (2020) DNF-Net: A Neural Architecture for Tabular Data
33. Xu L, Skoularidou M, Cuesta-Infante A, Veeramachaneni K (2019) Modeling Tabular data using Conditional GAN. Adv Neural Inf Process Syst 32:
34. Mathov Y, Levy E, Katzir Z, et al (2020) Not All Datasets Are Born Equal: On Heterogeneous Data and Adversarial Examples
35. Achille A, Paolini G, Soatto S (2019) Where is the Information in a Deep Neural Network?
36. Szegedy C, Zaremba W, Sutskever I, et al (2013) Intriguing properties of neural networks. 2nd Int Conf Learn Represent ICLR 2014 - Conf Track Proc
37. Patki N, Wedge R, Veeramachaneni K (2016) The synthetic data vault. Proc - 3rd IEEE Int Conf Data Sci Adv Anal DSAA 2016 399–410. https://doi.org/10.1109/DSAA.2016.49

38. Li Z, Zhao Y, Fu J (2020) SYNC: A Copula based Framework for Generating Synthetic Data from Aggregated Sources. IEEE Int Conf Data Min Work ICDMW 2020-November:571–578. https://doi.org/10.1109/ICDMW51313.2020.00082

39. Chow CK, Liu CN (1968) Approximating Discrete Probability Distributions with Dependence Trees. IEEE Trans Inf Theory 14:462–467. https://doi.org/10.1109/TIT.1968.1054142

40. Zhang J, Cormode G, Procopiuc CM, et al PrivBayes: Private Data Release via Bayesian Networks

41. Arjovsky M, Chintala S, Bottou L (2017) Wasserstein GAN

42. Gulrajani I, Ahmed F, Arjovsky M, et al (2017) Improved Training of Wasserstein GANs. Adv Neural Inf Process Syst 2017-December:5768–5778

43. Hjelm RD, Jacob AP, Che T, et al (2017) Boundary-Seeking Generative Adversarial Networks. 6th Int Conf Learn Represent ICLR 2018 - Conf Track Proc

44. Choi E, Biswal S, Malin B, et al (2017) Generating Multi-label Discrete Patient Records using Generative Adversarial Networks

45. Camino RD, Hammerschmidt CA, State R (2018) Generating Multi-Categorical Samples with Generative Adversarial Networks

46. Park N, Mohammadi M, Gorde K, et al (2018) Data Synthesis based on Generative Adversarial Networks. Proc VLDB Endow 11:1071–1083. https://doi.org/10.14778/3231751.3231757

47. Jordon J, Yoon J, Schaar M van der (2018) PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees

48. Xu L, Veeramachaneni K (2018) Synthesizing Tabular Data using Generative Adversarial Networks

49. Baowaly MK, Lin CC, Liu CL, Chen KT (2019) Synthesizing electronic health records using improved generative adversarial networks. J Am Med Inform Assoc 26:228–241. https://doi.org/10.1093/JAMIA/OCY142

50. Chen H, Jajodia S, Liu J, et al (2019) FakeTables: Using GANs to Generate Functional Dependency Preserving Tables with Bounded Real Data

51. Wen B, Colon LO, Subbalakshmi KP, Chandramouli R (2021) Causal-TGAN: Generating Tabular Data Using Causal Generative Adversarial Networks

52. Kamthe S, Assefa S, Chase M, et al (2021) Copula Flows for Synthetic Data Generation

53. Borisov V, Leemann T, Seßler K, et al (2021) Deep Neural Networks and Tabular Data: A Survey

54. Optuna - A hyperparameter optimization framework. https://optuna.org/. Accessed 5 Sep 2022

55. PyTorch. https://pytorch.org/. Accessed 5 Sep 2022

Appendices

Appendix

Table 22 - Torig utility values obtained in the Wine dataset in the setup 1:1.

	accuracy	precision	recall	F1-score	AUC
XGBoost	0.36	0.3383	0.36	0.3303	0.6816
Logistic Regression	0.3625	0.3536	0.3625	0.3239	0.7035
SVM	0.3734	0.3712	0.3734	0.3387	0.7156
Decision Tree	0.3475	0.3425	0.3475	0.3169	0.6727
MLP	0.358	0.3454	0.358	0.3255	0.684
KNeighbors	0.35	0.3402	0.35	0.3213	0.6775

Table 23 - Torig utility values obtained in the Wine dataset in the setup 1:3.

	accuracy	precision	recall	F1-score	AUC
XGBoost	0.3286	0.3798	0.3286	0.3044	0.6317
Logistic Regression	0.3429	0.3693	0.3429	0.3174	0.682
SVM	0.3191	0.3305	0.3191	0.2928	0.6563
Decision Tree	0.3036	0.3119	0.3036	0.2774	0.6277
MLP	0.3058	0.3128	0.3058	0.2828	0.6465
KNeighbors	0.3048	0.3039	0.3048	0.2793	0.6386

Table 24 - Tsyn utility values obtained with M1 for the Wine dataset in the setup 1:1.

	accuracy	precision	recall	F1-score	AUC
XGBoost	0.235	0.2438	0.235	0.2133	0.57
Logistic Regression	0.2625	0.2714	0.2625	0.2282	0.6235
SVM	0.275	0.2716	0.275	0.2357	0.6429
Decision Tree	0.2613	0.2599	0.2613	0.2291	0.6103
MLP	0.269	0.2734	0.269	0.2348	0.6246
KNeighbors	0.27	0.2682	0.27	0.2372	0.6242

Table 25 - Tsyn utility values obtained with M1 for the Wine dataset in the setup 1:3.

	accuracy	precision	recall	F1-score	AUC
XGBoost	0.3286	0.3798	0.3286	0.3044	0.6317
Logistic Regression	0.3429	0.3693	0.3429	0.3174	0.682

SVM	0.3191	0.3305	0.3191	0.2928	0.6563
Decision Tree	0.3036	0.3119	0.3036	0.2774	0.6277
MLP	0.3058	0.3128	0.3058	0.2828	0.6465
KNeighbors	0.3048	0.3039	0.3048	0.2793	0.6386

Table 26 - Difference values between Tsyn and Torig s from the 1:1 setup obtained with M1.

	accuracy	precision	recall	F1-score	AUC
XGBoost	-0.125	-0.0945	-0.125	-0.117	-0.1116
Logistic Regression	-0.1	-0.0822	-0.1	-0.0957	-0.08
SVM	-0.0984	-0.0996	-0.0984	-0.103	-0.0727
Decision Tree	-0.0862	-0.0826	-0.0862	-0.0878	-0.0624
MLP	-0.089	-0.072	-0.089	-0.0907	-0.0594
KNeighbors	-0.08	-0.072	-0.08	-0.0841	-0.0533

Table 27 - Difference values between Tsyn and Torig s from the 1:3 setup obtained with M1.

	accuracy	precision	recall	F1-score	AUC
XGBoost	-0.04	-0.0764	-0.04	-0.0742	-0.0825
Logistic Regression	-0.06	-0.0554	-0.06	-0.0717	-0.0487
SVM	-0.08	-0.0797	-0.08	-0.0872	-0.0491
Decision Tree	-0.08	-0.0858	-0.08	-0.0903	-0.0494
MLP	-0.076	-0.086	-0.076	-0.0871	-0.045
KNeighbors	-0.06	-0.0738	-0.06	-0.073	-0.0352

Table 28 - Tsyn utility values obtained with M2 for the Wine dataset in the setup 1:1.

	accuracy	precision	recall	F1-score	AUC
XGBoost	0.295	0.2511	0.295	0.2437	0.6558
Logistic Regression	0.3025	0.2985	0.3025	0.2565	0.6587
SVM	0.2917	0.2955	0.2917	0.2549	0.661
Decision Tree	0.2813	0.2837	0.2813	0.2487	0.6286
MLP	0.278	0.2636	0.278	0.2376	0.6341
KNeighbors	0.2784	0.2695	0.2784	0.2439	0.6244

Table 29 - Difference values between Tsyn and Torig s from the 1:1 setup obtained with M2

	accuracy	precision	recall	F1-score	AUC
XGBoost	-0.065	-0.0872	-0.065	-0.0866	-0.0258
Logistic Regression	-0.06	-0.0551	-0.06	-0.0674	-0.0448
SVM	-0.0817	-0.0757	-0.0817	-0.0838	-0.0546
Decision Tree	-0.0662	-0.0588	-0.0662	-0.0682	-0.0441
MLP	-0.08	-0.0818	-0.08	-0.0879	-0.0499
KNeighbors	-0.0716	-0.0707	-0.0716	-0.0774	-0.0531

Table 30 - Tsyn utility values obtained with M2 for the Wine dataset in the setup 1:3.

	accuracy	precision	recall	F1-score	AUC
XGBoost	0.2858	0.2718	0.2858	0.2477	0.6317
Logistic Regression	0.3072	0.2814	0.3072	0.252	0.657
SVM	0.3096	0.2749	0.3096	0.2516	0.6462
Decision Tree	0.2965	0.2635	0.2965	0.2455	0.618
MLP	0.3	0.2662	0.3	0.2484	0.636
KNeighbors	0.2929	0.2553	0.2929	0.2394	0.6303

Table 31 - Difference values between Tsyn and Torig s from the 1:3 setup obtained with M2

	accuracy	precision	recall	F1-score	AUC
XGBoost	-0.0142	-0.0521	-0.0142	-0.0411	-0.0495
Logistic Regression	-0.0286	-0.0758	-0.0286	-0.0652	-0.0708
SVM	-0.0571	-0.1036	-0.0571	-0.0944	-0.0726
Decision Tree	-0.0428	-0.0742	-0.0428	-0.0705	-0.0544
MLP	-0.0515	-0.0975	-0.0515	-0.0813	-0.0535
KNeighbors	-0.0619	-0.1135	-0.0619	-0.0956	-0.06