Evaluation of a Fintech Sales Synthetic Data Generation Model Using a Generative Adversarial Network

Felipe A. Lopez[1(✉)], Marcia Duran-Riveros[2], Sebastian Maldonado-Duran[2], David Ruete[2], Giannina Costa[2], Jairo R. Coronado-Hernandez[3], and Gustavo Gatica[2]

[1] Facultad de Ciencias Empresariales, Universidad del Bío-Bío, Avda. Collao 1202, 405023 Concepción, Chile
felipe@flopez.cl

[2] Engineering Faculty, Universidad Andres Bello, Antonio Varas 880, Providencia, 7500971 Santiago de Chile, Santiago, Chile
{mduran.riveros,smaldonado.duran,druete,giannina.costa,ggatica}@unab.cl

[3] Departamento de Productividad e Innovación, Universidad de la Costa, Cl. 58 #55 - 66, Nte. Centro Histórico, 080020 Barranquilla, Colombia
Jcoronad18@cuc.edu.co

Abstract. The need for more and better information for decision making is fundamental in modern organizations, especially in the financial industry. One type of this information is time series, which allow prediction and estimation of different scenarios, but are difficult to obtain for small and medium sized enterprises (SMEs). This research presents the design and validation of a generative adversarial network (GAN) capable of generating synthetic data for daily sales of Chilean SME. The problem that needs to be resolved is the lack of this kind of data within a Chilean fintech company called Dank. This data can be useful in developing an automatic risk evaluation model and, therefore, in reducing business process time, since risk evaluation is currently being carried out by people. The solution allows maintaining the anonymity of the data and using GAN to obtain different synthetic time series, increasing the data by 10%. It uses images from a vector of random numbers that are in temporal coherence and equal distribution. This research allows SMEs to obtain a greater amount of data, with a simple solution, to make better decisions.

Keywords: GAN · Times series · Fintech · Synthetic Data Generation

1 Introduction

The Fintech industry is based on three pillars: technological innovation, process disruption, and service transformation [1]. Although there is no consensus, the definition is "a company or subsidiaries of companies that combine financial services with modern and innovative technologies" [2].

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024
O. Gervasi et al. (Eds.): ICCSA 2024 Workshops, LNCS 14820, pp. 56–70, 2024.
https://doi.org/10.1007/978-3-031-65285-1_5

The evolution of this industry is commonly divided into three main periods [3]: Fintech 1.0: in this period one can find technologies ranging from transatlantic telegraph communication to ATMs; Fintech 2.0: one can find technologies ranging from the Internet to the Internet of Things; Fintech 3.0: data-driven technological innovations.

In all industries, although accentuated in the financial industry, the impact of digital technologies based on data, and in particular Artificial Intelligence (AI), is generating a structural change in the way business is conducted and organized [4]. This phenomenon is referred to in the literature as the Fourth Industrial Revolution [5].

In this scenario, large companies have a competitive advantage over small companies due to the amount of data they handle. For this reason, it is debated whether it affects free competition [6].

The industry is faced with privacy requirements, at least the anonymization of information before its use by internal or external teams [7, 8].

These difficulties, the lack of large volumes of data, and the need for anonymization have led to the need to generate synthetic data.

Synthetic data is defined as: "Any data used productively for a particular situation that is not obtained by direct measurement" [9].

The use of statistical techniques to share and study statistically relevant data, while preserving confidentiality, extracted from the US Census is discussed in [10]. Thus, it is proposed to use the same techniques used to impute missing data to generate a new series of statistically relevant data.

From the 1990s until 2010, the discipline of data generation was dominated by statistical models; these types of methods produce a steady increase in model complexity and require the use of complex correction methods in addition to non-linear or non-standard analysis [11, 12].

With the advent of AI, particularly computer vision, synthetic data generation is not only being refined but also extended to the use of simulated environments. The increased detail of three-dimensional models has succeeded in replacing real data for successful model training. The variability ranges from object recognition to face recognition [13, 14].

One of the main methods for data generation is Generative Adversarial Networks (GAN), a method that consists of training two models simultaneously, a generation model and a discrimination model [15]. In this case, the generation model is fed with a noise component to ensure its variability and is trained to generate a data set that follows the same distribution as a previously defined set. The discrimination model is trained to distinguish when a data set matches the ensemble and when it does not. Training ends when the discrimination model is unable to effectively distinguish whether a data set is synthetically generated or part of the originally defined set [16].

Although this type of model is successful in generating synthetic data, it shows weaknesses in generating time-dependent data [17]. This leads to the development of new GANs that allow the generation of synthetic time series [18, 19].

In the financial industry, time-dependent data or time series are highly valued and are mainly used in machine learning and forecasting models [20–22]. This information allows them to evaluate different scenarios and make better decisions. However, this

type of information is costly and difficult to obtain for SMEs and companies that have only been in business for a few years.

Our research aims to generate synthetic time series using a GAN model, this research of exploratory type, with an Action Research approach [23]. It uses a design science methodology and is associated with a fintech that supports small and medium enterprises (SMEs) by providing financing solutions. The business objective is to reduce the duration of the factoring finance delivery process, for which it is necessary to synthetically increase the amount of data by 10%.

This paper is structured with this introduction, Sect. 2 presents a background and related work. Section 3 describes the methodology, the decisions and methods used in our development, then Sect. 4 reports the results and discussions, and Sect. 5 is a summary of conclusions, limitations, and recommendations for future work.

2 Background and Related Work

GAN research in finance is still in its infancy, mainly because GAN does not naturally model time series [24]. Research that combines different techniques with GAN (neural networks, ML algorithms, etc.) has been incorporated to model time series.

In [25], they aim to generate time series focusing on reproducing the statistical properties of financial time series. It takes 6 statistical properties of this type of series: 1) linear unpredictability, 2) thick-tailed distribution, 3) volatility clustering, 4) leverage effect, 5) coarse-fine volatility correlation, and 6) profit/loss asymmetry. To generate data sets with these properties, they generate three different GANs, one with a multilayer perceptron generator with four fully connected layers and a hyperbolic tangent activation function (MPL); another with a six-layer convolutional generator with a ReLu activation function (CNN); and finally a combination of both, where both layers are used in parallel and the final results are combined with an element-by-element multiplication (MPL-CNN).

The results of the above networks are compared with each other and with previous work to verify that they satisfy these properties. Thus, it is presented how to set up a GAN to preserve the relationships between the data of the GANs.

In [26], the authors address the problem of data imbalance, which occurs when data are divided into positive and negative data and causes a bias in the learning of the GAN. To the existing solutions, which consist in modifying the data set to reduce the more frequent part and modifying the objective functions to give more importance to the false cases, the authors propose the generation of negative data by a modified GAN.

To do this, they take a type of recurrent neural network, the Long-Short Term Memory (LSTM), and transform it into a neural network layer to preserve the temporal relationship of the data to a time series. Then they penalize the objective function obtained from the real data and the generated data, and finally, they use the Kernel Maximum Mean Discrepancy (Kernel MMD) metric [27], which allows them to evaluate the performance of the model.

In [28], the database contains fraudulent transactions with a class imbalance. A synthetic generation of fraudulent transactions is presented to improve the training performance of the models. The characteristics of the user and his transactions are used.

The neural architecture is divided into two parts, a fusion module that tries to mix these two types of data to leave them in a single homogeneous set, and a balancing module that allows to generate examples of fused data of fraudulent transactions. The discriminator of the latter module distinguishes between three classes: normal, fraudulent, and synthetic transactions.

Improving the performance of a neural model to discriminate whether an e-money transaction is fraudulent is the goal of the research in [29]. Therefore, the use of autoencoders is explored to extract features from the data sets, which are then fed to a model that discriminates whether the transaction is fraudulent or not. Given the imbalance of the data, an architecture with two information generators is created to improve the balance. The architecture considers three phases, in the first one they use a Wasserstein GAN (WGAN) [16] to generate synthetic fraudulent data, with the augmented dataset they go to a second phase where they train two autoencoders: one for the augmented fraudulent data and another for the benign data; with the trained autoencoders they start a third phase where they train a second generative network that generates data equivalent to those generated by the autoencoders as input for the final model. This study concludes that the use of dataset features improves the performance of neural networks, but they require a larger number of data to overcome the class imbalance problem. In addition, it applies the novelty of having two generative networks in sequence to perform its task and this not only does not affect the performance of the final network but improves it.

In [30], GAN is used to generate synthetic data in the financial domain, in response to the imbalance of data with fraudulent transactions, in this case, fraudulent transactions represent 0.185% of the total number of records. To complement the results, they train a discriminator purely with real data to compare with another model. The results of the study are compared with data generated by the Synthetic Minority Oversampling Technique (SMOTE) method [31]. They conclude that this method can be applied in the field of synthetic generation of financial data, with results similar to other methods.

In [32], they use conditional GAN to model financial time series; since conditional GAN requires a vector to label the data according to some characteristic. An experimental estimation is performed with the S&P 500 index, the cycles are analyzed, and the convergence of the GAN is evaluated, as well as its performance according to the chosen hyperparameters.

They then build three GANs, a small one (cGAN-Small) with 5 neurons in the discriminator (D) and the generator (G), a medium one (cGAN-Medium) with 100 neurons, and a large one (cGAN-Large) with 500 neurons each. They add to the GAN two types of basic learners (M) to be combined with the GAN: Reg Tree (Reg Tree) and Multilayer Perceptron (MLP), for which they use different numbers of instances (B) of the methods. For their evaluation, they use the Sharpe index, the Calmar ratio, and the RMSE.

The mathematical support for the use of temporal convolutional networks is provided by [24], who use stochastic volatility neural networks in the time series domain, evaluated on the S&P 500 index. In addition to the neural network presented, the information preprocessing pipeline is analyzed, where, in addition to the normalizations performed, an inverse Lambert transform is performed [33].

The authors propose an architecture using layers of extended convolutional neural networks. They compare the generated neural network models and their performance with respect to several metrics, including the Wasserstein distance, the Dragulescu and Yakovenko (DY) metric, and the ACF score (which measures the historical dependence of the data). They conclude that temporal convolutional networks outperform the autoregressive model with conditional heteroscedasticity (GARCH) and its C-SVNN variant for generating synthetic time series.

In [34], they address the problem of detecting fraudulent banking transactions, to overcome the imbalance problem, they augment the dataset with synthetic data. For data generation, they combine a GAN with a denoising autoencoder (DAE), a DAE is a special case of an autoencoder that adds noise to the input data (x) to then transform it into another data set (x') through a function (f). Training a DAE consists in minimizing the error between x and $f(x')$. The results show the usefulness of mixing different techniques to generate synthetic data, as well as the need to measure the degree of similarity between the two data sets.

3 Material and Methods

The methodology used is based on the science of design, using metrics that validate the previously defined objectives [35, 36]. This research is exploratory in nature and uses an Action Research approach [23] as a meta-methodology, since the focus will be on application in the organizational context, with the specific objective of improving a business process.

The programming language used is Python 3.8.5, on Anaconda version 2020.11. The following libraries used torch, tqdm and utils.

The original dataset consists of a list of 1,495 time series of length 512, corresponding to daily totals of issued invoices, each series generated by a different company. The dataset has been anonymized to avoid bias and privacy issues.

The fields with continuous values ($) were normalized by logarithm to smooth their effect. To measure the performance of the GAN generator, the following variables are considered: Comparison of the Fréchet distance of the generated time series with the existing ones, by random sampling; Comparison of the distance generated by dynamic time distortion of the generated time series with the existing ones, by random sampling.

The objective is to design and test a model that allows the generation of synthetic company sales data to support risk assessment and thus discriminate and accelerate the decision-making of financial partners.

To achieve the objective, the following tasks are required:

- Extract an anonymized sample of the data to be enriched.
- Design a model capable of generating synthetic company data.
- Implement the data generation model.
- Implement the data discrimination model.
- Generate synthetic data that is indistinguishable to the data discriminator.
- Implement a classification that allows visual comparison of synthetic and real data.
- Implement metrics to determine the closeness of the generated data to the real data.

The process for generating synthetic sales data is shown in Fig. 1, and Sect. 3.1 explains each step of its execution.

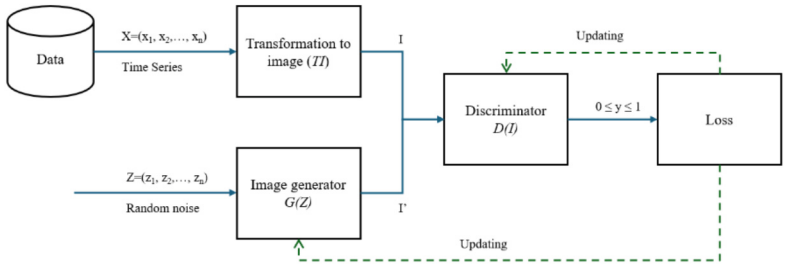

Fig. 1. Diagram of GAN components for synthetic data generation. Source: Own elaboration.

Once the tasks listed above have been completed, the direct impact on the specific objective will be measured.

3.1 Project Execution

First, a classification of the time series was made using the average sales and the number of days with sales for each company. It is observed that the average sales are less than one million CLP and less than 30% of the days with sales, the detail of the distribution is shown in Fig. 2.

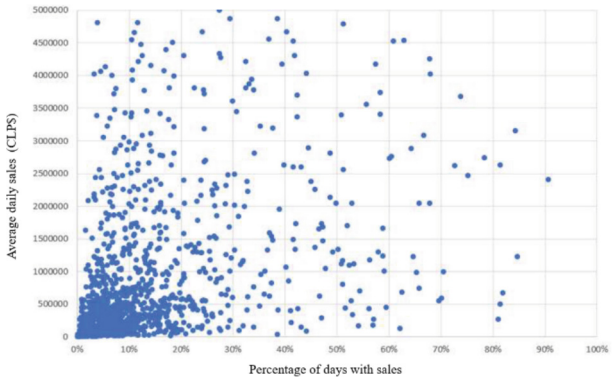

Fig. 2. Distribution of % of daily sales by company and daily average.

3.1.1 Transformation to Image

Then, each time series is transformed into an image. This procedure consists of a function (*TI*) that, after receiving a time series (*x*), returns an image (*I*). To represent the time series, an image of height and width (n = 512) is defined, where each row represents

one day of information. In addition, two channels are used: green and red. For green, the Graminian Angular Field (GAF) method [37] is used with the summation option [38, 39], the method allows to visualize of the distances that exist between each of the points of the time series, in this way the temporal relationships between the data are graphically preserved. In addition, the unnormalized logarithmically normalized value of the time series is added to each row of the image. To represent this value, the binary conversion of the number is performed and then this value is added to the red channel as 0 or 255, representing the zero or one corresponding to the binary value.

To generate the random noise, the rand function of the Python torch package [40] is used to generate random number vectors of the type $Z = (z1, z2, \ldots zk)$, where $zj \in \mathbb{R}$ and $k = 64$.

3.1.2 Image Generator

The generator (G) of the series corresponds to a function that generates images of size n x n, given an input Z, i.e. $G(Z) = I$. For the generated one, 4 linear layers are established where the ReLU activation function is normalized and applied, using the Python torch package. Finally, a final linear layer with a sigmoid activation function is added.

To represent the time series, an image of height and width $n = 512$ is defined, where each row represents one day of information. In addition, two channels are used: green and red.

For the green, the Graminian Angular Field (GAF) method is used with the sum option [38], this method allows visualizing the distances that exist between each of the points of the time series and in this way the temporal relationships between the data are graphically preserved.

3.1.3 Discriminator

A function is used as discriminator (D), which allows to indicate the degree of similarity with the real set of the synthetic time series, this is expressed in the following function $D(I) = y$ with $0 \leq y \leq 1$.

The degree of membership (y) of the discriminator allows us to update the weights of the generator and the discriminator, respectively, according to the loss function. Four sequential linear layers are designed, three of them with LeakyReLU activation function with a negative slope of 0.2 and a last linear layer. A hidden dimension size of length 128 is defined.

For the loss function, which measures the distance between the probability distributions, the cross-entropy function BCEWithLogitsLoss [40] is used. In the case of the discriminator, the average loss of this component is calculated with respect to its performance, i.e., the ability to correctly discriminate synthetically generated images and images belonging to the ensemble.

In the main execution function, the parameters that can modify the learning result, such as the number of epochs and the learning rate of the network, are received. In this function, the neural network is completed with the Adams optimization function [41, 42]. Then, for each of the training epochs, a batch of data is taken, it is determined whether the update of the discriminator occurs, the loss is calculated, and the nodes of

the neural network are updated. Later, the same is done with the generator, obtaining a generated data set, calculating its loss, and calculating the update of the nodes belonging to its network.

3.1.4 Loss

In general, generative adversarial networks attempt to replicate the probability distribution of a set in another synthetically generated set. To do this, metrics are used to measure the distance between probability distributions. This metric of the difference between the distributions is called the loss function or cost function.

Variables containing metrics are updated and their results are analyzed. In this neural network execution algorithm, depending on the input parameters, it is possible to update the discriminator and the generator in a differentiated way, which allows, if necessary, to omit the learning of the discriminator.

In addition, variations of the neural architecture are made to explore its use, integrating convolutional layers into the neural network architecture, given the high computational cost, surface convolutional layers are specifically chosen, however, these generated images of low quality and resolution, not representing time series.

To solve the above problem, the complexity of the convolutional layer is increased, it is trained and after 24 h of continuous work, it is calculated that the complete training can take between 3 and 4 months, therefore the execution of this experiment is abandoned.

In the last additional test, the loss function used in the neural network is modified by the Wasserstein loss function, but it does not converge and gives lower results than those obtained by GAN.

3.1.5 GAN Execution

The main execution function receives the parameters that can modify the learning result, such as the number of epochs and the learning rate of the network. In this function, the configuration of the neural network is completed with the Adams optimization function.

Then, for each of the training epochs, a batch of data is taken, it is determined whether the update of the discriminator occurs and, if so, the loss is calculated, and the nodes of the neural network are updated, calculating its loss and calculating the update of the nodes belonging to its network.

In this neural network execution algorithm, depending on the input parameters, it is possible to update the discriminator and the generator differentially, which allows, if necessary, to omit the learning of the discriminator.

3.1.6 Hardware Infrastructure

Two machines are used in the design phase:

Processor: 2.2 GHz Intel Core i7 6-core 8th generation 64bit Intel Core i7; RAM Memory: 16 GB 2400 MHz DDR4 RAM; Video chip: Radeon Pro 555X 4 GB.

Processor: 1.6 GHz Intel Core i5-8250U 4 cores 64 bits; RAM Memory: 8 GHz 1866 MHz DDR4; Video chip 1: Intel UHD Graphics 620 and NVIDIA GeForce MX150 2 GB.

Plus a third AWS g4dn.16xlarge machine to run the experiment.
Processor: 64 vCPU's; RAM memory: 256GB; Video chip: 16GB.

4 Preliminary Results and Discussion

Twenty simulations were performed to generate the synthetic time series and to evaluate the different metrics obtained. Information from the other simulations has been omitted as they are the variations that showed the lowest indicators. The hyperparameters used in the most representative tests are shown in Table 1.

Table 1. Hyper-parameters for GAN execution and synthetic time series generation.

Hiper-parameters	First	Second	Third (*)	Fourth
Epochs (numbers)	250	500	125	75
Batch size	20	20	20	20
Noise vector size	64	64	64	64
Device	CPU	CPU	CPU	CPU
Learning rate	0.00001	0.00001	0.00001	0.00001

Note: (*) Represents hyperparameters with best results

The execution time was at least 5 h for each simulation, the memory usage varied between 98%–100% RAM usage. Due to the architecture used, it was not possible to use the available video card, only the CPU of the machine was used.

The third simulation was chosen because it gives the best distribution of results. From these cases, a sample of generated and real time series is taken and compared to establish their similarity and thus verify the validity of the generated data. The metrics chosen for this purpose are the Dynamic Time Distortion (DTW) [43, 44] and the Fréchet Distance [45].

The distribution of the time series is within the expected range in terms of the distribution of days with sales, as shown in Fig. 3.

When using the Dynamic Time Distortion method, distances are first calculated between a sample of the real time series to establish an expected base of diversity among the synthetic series. Similarly, the distances between the synthetic time series are calculated to establish the diversity in a generated universe.

Next, the exercise of calculating the distance between the real and synthetic time series samples is performed. In this way, the behavior of both can be compared. Table 2 shows the DTW values for the real and synthetic data. It can be seen that, although there is a difference between the observed distance ranges between the synthetic and real time series, the gap decreases when they are compared, and they reach values similar to those obtained when measuring the distance between real time series.

Similarly, the difference in the standard deviation between the distances collected between the real series and the standard deviations of the synthetic time series compared

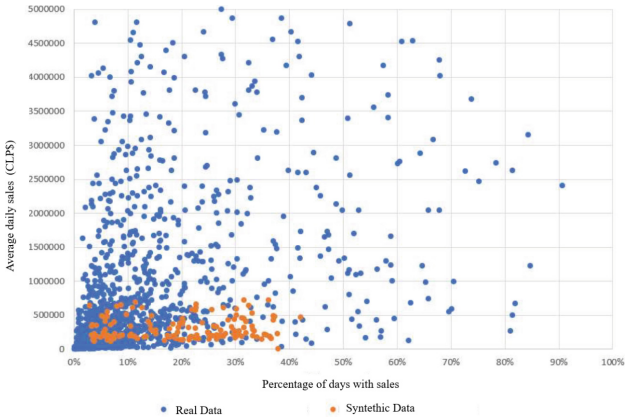

Fig. 3. Original and synthetic average sales comparison.

Table 2. DTW Data Trend Measure Summary.

	Min	Max	Average	SD
Real Data (RD)	56,588,519	329,227,381	204,915,746	55,488,325
Synthetic Data (SD)	39,133,762	276,994,524	158,807,390	69,230,463
RD v/s SD	66,032,324	339,255,910	203,836,510	65,819,000

to the real ones shows a gap evidenced by time series concentrations that account for little variability.

Understanding that the comparisons are relative, a normalization of the distance data is performed, with respect to the maximum of the set. With the normalized data, Table 3 is created, which groups and counts the distances at regular intervals in order to analyze the distribution of the occurrences of the normalized distances.

The distribution in the normalized intervals, and the synthetic time series tend to have a lower variability distributed in all the intervals, causing some concentrations of these. Most of the distances are in the interval 0.4 to 0.9, concentrating the distances in a higher interval than observed in the behavior between real time series distances, which are more distributed throughout the interval. This shows a low variability in the generated series.

The distribution between both series has similarities in its distribution, despite not being aligned, they are not different distributions, but the same type of distribution in a different interval.

When evaluating the Fréchet distance, it is observed that the distances of the generated series are within the real ones. Table 4 shows the minimum, maximum, average, and standard deviation values. From these data, it can be concluded that the distances of the generated series are within the real ones for the synthetic data set and for the synthetic data set versus the real data.

Table 3. DTW frequencies relative to maximum.

Interval	Real Data (RD)	Synthetic data (SD)	RD - SD
0–0.1	0	0	0
0.1–0.2	1	25	1
0.2–0.3	16	24	27
0.3–0.4	8	27	45
0.4–0.5	34	30	84
0.5–0.6	45	17	58
0.6–0.7	54	36	76
0.7–0.8	51	26	65
0.8–0.9	19	5	60
0.9–1	3	0	24

Similarly, the standard deviation shows that the synthetic time series have less variability between them, a result of the previous analysis regarding their classification. In addition, it is important to highlight that the maximum distance between two series is in the real data set, so no time series escapes the domain of the real series.

Table 4. Summary of trend measures of Fréchet data.

Conjunto	Min	Max	Average	SD
RD	941,033	30,277,974	10,235,362	6,475,263
SD	2,097,152	19,386,512	8,852,097	2,893,125
RD - SD	2,097,152	26,166,589	9,919,285	4,389,311

Since the comparison between the different distances is relative, and to facilitate the visualization of the results, the entire table is normalized by dividing the distances by the maximum distance. Table 5 is generated with the observed data and shows the frequencies at which the relative distances occur for each data set.

It is verified that most of the distances are accumulated between 20% and 60% of the maximum distance, another result of the low variability between the generated series, but they are reasonably distant from the real series and from each other.

From the evaluation of the best case, it is possible to state that the model allows to increase in the data by the expected 10%: from the original 1495 cases, 150 time series are generated, and distributed analogously to the original series.

It is important to note that the time series are not distributed over the entire available space; the values are strongly clustered in the range of less than $500,000 per day.

One possible cause of this is the way the time series are encoded; it is possible that the higher numbers are constrained by the approximation of the encoding of real numbers

Table 5. Frequencies of Fréchet distances relative to maximum distance.

Interval	Real Data (RD)	Synthetic data (SD)	RD - SD
0–0.1	14	4	22
0.1–0.2	56	25	66
0.2–0.3	51	86	116
0.3–0.4	38	49	121
0.4–0.5	27	21	65
0.5–0.6	13	3	33
0.6–0.7	13	2	7
0.7–0.8	9	0	8
0.8–0.9	6	0	2
0.9–1	3	0	0

between 0 and 1 to the discrete interval between 0 and 1 in order to obtain a binary encoding of the magnitudes.

This can be avoided by setting these values to one, but this distorts the time series towards higher magnitudes without having a valid basis for doing so.

On the other hand, analysis of the DTW and Fréchet distances confirms that the time series are distributed similarly to a subset of the original time series.

5 Conclusions

This research has focused on exploring a method to generate synthetic time series for the financial industry. A model capable of generating time series that respect the original distribution of the data is designed.

The strengths of the solution obtained are to maintain the anonymity of the data, i.e. it is not possible to deduce the original data from the generated data. Thus, it is possible to maintain the exportability of the data within the legal framework and to reduce the risk of information leakage.

The solution is easy to implement since the various GANs that solve the time series generation problem correspond to complicated amalgamations that use complex algorithms and optimizations to maintain the temporal coherence of the time series. This allows SME companies to have more and more information for decision-making.

Finally, since the model generates images from a vector of random numbers, it is possible to generate an unlimited number of time series that must be within the same distribution already discovered, so despite its limitations around the variability of the series, it provides a substantial amount of data to begin the development of a credit evaluation model. As a result, SME companies can have more and better tools to make them competitive.

This work is not free of limitations, the main one related to the high consumption of computational resources, which meant limiting the number of experiments to be

performed by modifying the hyperparameters to increase the variability of the time series generated.

On the other hand, the limitation related to the nature of the time series and its variability, so that the general shape of the series and its distribution may change over time so that in order to continue obtaining valid series, it is necessary to constantly train the neural network to adapt to changing market conditions.

Finally, the limitation related to the amount of data (real time series) with which we have worked should be extended to verify if there are changes in the results.

As a future development, it is proposed to perform tests with other time series on image encodings that are less dependent on the resolution of the generated image. In this way, modifications can be made using a recurrent neural network model with memory (RNNM model).

References

1. Gomber, P., Kauffman, R.J., Parker, C., Weber, B.W.: On the Fintech revolution: interpreting the forces of innovation, disruption, and transformation in financial services. J. Manag. Inf. Syst. **35**, 220–265 (2018). https://doi.org/10.1080/07421222.2018.1440766
2. Dorfleitner, G., Hornuf, L., Schmitt, M., Weber, M.: The Fintech market in Germany. SSRN Electron. J. (2016). https://doi.org/10.2139/ssrn.2885931
3. Leong, K.: FinTech (Financial Technology): what is it and how to use technologies to create business value in Fintech way? Int. J. Innov. Manag. Technol. 74–78 (2018). https://doi.org/10.18178/ijimt.2018.9.2.791
4. Makridakis, S.: The forthcoming Artificial Intelligence (AI) revolution: its impact on society and firms. Futures **90**, 46–60 (2017). https://doi.org/10.1016/j.futures.2017.03.006
5. Xu, M., David, J.M., Kim, S.H.: The fourth industrial revolution: opportunities and challenges. Int. J. Financ. Res. **9**, 90 (2018). https://doi.org/10.5430/ijfr.v9n2p90
6. Stucke, M.E., Grunes, A.P.: Introduction: big data and competition policy. In: Big Data and Competition Policy. Oxford University Press (2016)
7. Heaton, J.B., Witte, J.: Synthetic financial data: an application to regulatory compliance for broker-dealers. J. Financ. Transform. (2019)
8. Couldry, N., Mejias, U.A.: Data colonialism: rethinking big data's relation to the contemporary subject. Telev. New Media **20**, 336–349 (2019). https://doi.org/10.1177/1527476418796632
9. Dictionary of Engineering. McGraw Hill LLC (2003). https://doi.org/10.1036/0071417990
10. Rubin, D.B.: Discussion: statistical disclosure limitation. J. Off. Stat. **9**, 461–468 (1993)
11. Reiter, J.P., Drechsler, J.: Releasing multiply-imputed synthetic data generated in two stages to protect confidentiality. Stat. Sin. **20**, 405–421 (2010)
12. Drechsler, J.: Synthetic Datasets for Statistical Disclosure Control. Springer, New York (2011). https://doi.org/10.1007/978-1-4614-0326-5
13. Handa, A., Patraucean, V., Badrinarayanan, V., Stent, S., Cipolla, R.: Understanding Real-World indoor scenes with synthetic data. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4077–4085. IEEE (2016). https://doi.org/10.1109/CVPR.2016.442
14. Abbasnejad, I., Sridharan, S., Nguyen, D., Denman, S., Fookes, C., Lucey, S.: Using synthetic data to improve facial expression analysis with 3D convolutional networks. In: 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), pp. 1609–1618. IEEE (2017). https://doi.org/10.1109/ICCVW.2017.189

15. Goodfellow, I.J., et al.: Generative adversarial nets. In: Proceedings of the 27th International Conference on Neural Information Processing Systems, vol. 2, pp. 2672–2680 (2014). https://doi.org/10.5555/2969033.2969125
16. Adler, J., Lunz, S.: Banach Wasserstein GAN. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. Curran Associates, Inc. (2018)
17. Liao, S., Ni, H., Szpruch, L., Wiese, M., Sabate-Vidales, M., Xiao, B.: Conditional Sig-Wasserstein GANs for Time Series Generation (2020). https://doi.org/10.48550/arXiv.2006.05421
18. Choi, K., Yi, J., Park, C., Yoon, S.: Deep learning for anomaly detection in time-series data: review, analysis, and guidelines. IEEE Access **9**, 120043–120065 (2021). https://doi.org/10.1109/ACCESS.2021.3107975
19. Raturi, R., Kumar, A., Vyas, N., Dutt, V.: A novel approach for anomaly detection in time-series data using generative adversarial networks. In: 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS), pp. 1352–1357. IEEE (2023). https://doi.org/10.1109/ICSCSS57650.2023.10169365
20. Sezer, O.B., Gudelek, M.U., Ozbayoglu, A.M.: Financial time series forecasting with deep learning: a systematic literature review: 2005–2019. Appl. Soft Comput. **90**, 106181 (2020). https://doi.org/10.1016/j.asoc.2020.106181
21. Cryer, J.D., Chan, K.-S.: Time Series Analysis With Applications in R. Springer, New York (2008)
22. Makridakis, S., Wheelwright, S.C., Hyndman, R.J.: Forecasting. Methods and Applications. Wiley, (1998)
23. Erro-Garcés, A., Alfaro-Tanco, J.A.: Action research as a meta-methodology in the management field. Int. J. Qual. Methods **19**, 160940692091748 (2020). https://doi.org/10.1177/1609406920917489
24. Wiese, M., Knobloch, R., Korn, R., Kretschmer, P.: Quant GANs: deep generation of financial time series. Quant. Financ. **20**, 1419–1440 (2020). https://doi.org/10.1080/14697688.2020.1730426
25. Takahashi, S., Chen, Y., Tanaka-Ishii, K.: Modeling financial time-series with generative adversarial networks. Phys. A Stat. Mech. its Appl. **527**, 121261 (2019). https://doi.org/10.1016/j.physa.2019.121261
26. Zhang, Z., et al.: A generative adversarial network–based method for generating negative financial samples. Int. J. Distrib. Sens. Netw. **16**, 155014772090705 (2020). https://doi.org/10.1177/1550147720907053
27. Gretton, A., Borgwardt, K., Rasch, M., Scholkopf, B., Smola, A.: A kernel method for the two-sample-problem. Adv. Neural Inf. Process. Syst. **19** (2006)
28. Lei, K., Xie, Y., Zhong, S., Dai, J., Yang, M., Shen, Y.: Generative adversarial fusion network for class imbalance credit scoring. Neural Comput. Appl. **32**, 8451–8462 (2020). https://doi.org/10.1007/s00521-019-04335-1
29. Wu, E., Cui, H., Welsch, R.E.: Dual autoencoders generative adversarial network for imbalanced classification problem. IEEE Access **8**, 91265–91275 (2020). https://doi.org/10.1109/ACCESS.2020.2994327
30. Fiore, U., De Santis, A., Perla, F., Zanetti, P., Palmieri, F.: Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. Inf. Sci. (NY) **479**, 448–455 (2019). https://doi.org/10.1016/j.ins.2017.12.030
31. Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W.: SMOTEBoost: improving prediction of the minority class in boosting. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) Knowledge Discovery in Databases: PKDD 2003. PKDD 2003. LNCS, pp. 107–119 (2003). https://doi.org/10.1007/978-3-540-39804-2_12

32. Koshiyama, A., Firoozye, N., Treleaven, P.: Generative adversarial networks for financial trading strategies fine-tuning and combination. Quant. Financ. **21**, 797–813 (2021). https://doi.org/10.1080/14697688.2020.1790635
33. Goerg, G.M.: The Lambert Way to Gaussianize Heavy Tailed Data with the Inverse of Tukey's has a Special Case (2010)
34. Zheng, Y.-J., Zhou, X.-H., Sheng, W.-G., Xue, Y., Chen, S.-Y.: Generative adversarial network based telecom fraud detection at the receiving bank. Neural Netw. **102**, 78–86 (2018). https://doi.org/10.1016/j.neunet.2018.02.015
35. Hevner, A., Chatterjee, S.: Design Research in Information Systems. Springer, Boston (2010). https://doi.org/10.1007/978-1-4419-5653-8
36. Araya, S., et al.: Design of a system to support certification management with an adaptive architecture. In: 2021 16th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1–6 (2021). https://doi.org/10.23919/CISTI52073.2021.9476390
37. Sreeram, V., Agathoklis, P.: On the properties of Gram matrix. IEEE Trans. Circuits Syst. I Fundam. Theory Appl. **41**, 234–237 (1994). https://doi.org/10.1109/81.273922
38. Faouzi, J.: Time series classification: a review of algorithms and implementations. In: Kotecha, K. (ed.) Machine Learning (Emerging Trends and Applications), p. In Press. Proud Pen (2022)
39. Wang, Z., Oates, T.: Imaging Time-Series to Improve Classification and Imputation (2015). https://doi.org/10.5555/2832747.2832798
40. Karasan, A.: Machine Learning for Financial Risk Management with Python. O'Reilly Books (2022)
41. Rojano, A., Salazar, R., Miranda, L., Ojeda, W.: Algoritmo Adam en la Inteligencia Artificial. In: COMEII-21005. Sexto Congreso Nacional de Riesgo, drenaje y Biosistemas, pp. 1–8 (2021)
42. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization (2014)
43. Dahmen, J., Cook, D.: SynSys: a synthetic data generation system for healthcare applications. Sensors **19**, 1181 (2019). https://doi.org/10.3390/s19051181
44. Esling, P., Agon, C.: Time-series data mining. ACM Comput. Surv. **45**, 1–34 (2012). https://doi.org/10.1145/2379776.2379788
45. Alt, H., Godau, M.: Computing the Frechet distance between two polygonal curves. Int. J. Comput. Geom. Appl. **05**, 75–91 (1995). https://doi.org/10.1142/S0218195995000064