*mathematics*

MDPI

*Review*

# Survey on Synthetic Data Generation, Evaluation Methods and GANs

**Alvaro Figueira [1,*,†] and Bruno Vaz [2,†]**

[1] CRACS-INESC TEC, University of Porto, 4169-007 Porto, Portugal
[2] Faculty of Sciences, University of Porto, Rua do Campo Alegre, s/n, 4169-007 Porto, Portugal; vazgbruno@gmail.com
* Correspondence: arfiguei@fc.up.pt
† These authors contributed equally to this work.

**Abstract:** Synthetic data consists of artificially generated data. When data are scarce, or of poor quality, synthetic data can be used, for example, to improve the performance of machine learning models. Generative adversarial networks (GANs) are a state-of-the-art deep generative models that can generate novel synthetic samples that follow the underlying data distribution of the original dataset. Reviews on synthetic data generation and on GANs have already been written. However, none in the relevant literature, to the best of our knowledge, has explicitly combined these two topics. This survey aims to fill this gap and provide useful material to new researchers in this field. That is, we aim to provide a survey that combines synthetic data generation and GANs, and that can act as a good and strong starting point for new researchers in the field, so that they have a general overview of the key contributions and useful references. We have conducted a review of the state-of-the-art by querying four major databases: Web of Sciences (WoS), Scopus, IEEE Xplore, and ACM Digital Library. This allowed us to gain insights into the most relevant authors, the most relevant scientific journals in the area, the most cited papers, the most significant research areas, the most important institutions, and the most relevant GAN architectures. GANs were thoroughly reviewed, as well as their most common training problems, their most important breakthroughs, and a focus on GAN architectures for tabular data. Further, the main algorithms for generating synthetic data, their applications and our thoughts on these methods are also expressed. Finally, we reviewed the main techniques for evaluating the quality of synthetic data (especially tabular data) and provided a schematic overview of the information presented in this paper.

**Keywords:** synthetic data generation; generative adversarial networks; evaluation of synthetic data

**MSC:** 68T07



**Citation:** Figueira, A.; Vaz, B. Survey on Synthetic Data Generation, Evaluation Methods and GANs. *Mathematics* **2022**, *10*, 2733. https://doi.org/10.3390/math10152733

Academic Editor: Catalin Stoean

Received: 1 July 2022
Accepted: 24 July 2022
Published: 2 August 2022



## 1. Introduction

Data are ubiquitous and can be a source of great value. However, to create such value, the data needs to be of high quality. In addition, when dealing with sensitive data (e.g., medical records or credit datasets), the privacy of the data must be ensured without sacrificing quality. The lack of high-quality data and the need for privacy-preserving data has become increasingly apparent in the last few years as companies and researchers use it more and more. Synthetic data consists of artificially generated data [1] and is a quite powerful tool to overcome the two aforementioned problems. Because synthetic data are generated rather than collected or measured, they can be of much higher quality than real data. Moreover, privacy constraints can be applied so that the synthetic data does not reveal any important information, such as patients' clinical records.

Although this is a very good idea, synthetic data must be generated properly: it must be plausible and follow the underlying distribution of the original data (Synthetic data can also be generated, for example, to create video games. In this case, it may

not need to resemble real-world data, but this is not the focus of our study). Therefore, the algorithms that generate it must be robust and capture the patterns in the real data. SMOTE [2] (Synthetic Minority Oversampling Technique) is one of the oldest algorithms that try to replicate a data distribution (it was proposed in 2002), apart from Random OverSampling (ROS) and other traditional algorithms such as rotation or scaling. The idea matured over the years, and several variants have been proposed [3–6]. However, it was not until the advent of Deep Learning that more promising ideas emerged, such as Variational AutoEncoders [7] (VAEs) in 2013 and, most importantly, Generative adversarial networks [8] (GANs) in 2014.

GANs are a powerful deep generative model trained with an adversarial procedure. Similar to SMOTE, GANs have undergone several modifications since they were first proposed to solve several different problems in different domains, e.g., physics [9] or healthcare [10]. However, the main focus has been on computer vision tasks where the domain consists generically of images. Nevertheless, tabular datasets are abundant, and the generation of synthetic tabular data is of great interest. For this reason, in this work, we have investigated different methods for generating synthetic data (with emphasis on tabular data), as well as the different GAN architectures that have been proposed over the years, with a particular emphasis on GANs that can generate synthetic tabular data.

Another important aspect we have studied is the evaluation of the quality of synthetic samples. As explained earlier, synthetic data can be of great use, but it is critical that such artificial data are plausible and can mimic the underlying data distribution of real datasets. Therefore, it is important to have methods to accurately assess the quality of the generated data. However, one problem that arises in such an assessment is the question of what to assess. One may want to generate synthetic data to improve the performance of a machine learning (ML) model, while others may need synthetic data with novel patterns without worrying too much about the performance of the model. Thus, depending on the problem and domain, some techniques are better suited than others. Clearly, there is not a one-size-fits-all evaluation method.

As such, we are combining a general overview of three main topics: synthetic data generation algorithms, GANs, and the evaluation of synthetic data. Moreover, we provide particular emphasis on GANs for tabular data generation, as we believe this is a not so well explored topic, unlike GANs for image generation. This can be quite convenient for new researchers in the field, as there is useful material and references in this survey. In turn, they can boost their research by having a general overview of the key breakthroughs in the field as well as an organizational and temporal summary of what has been reviewed throughout the document.

The remaining of this survey is organized as follows. Section 2 provides an overview of the current state-of-the-art in terms of research in the area and presents the major scientific key insights concerning the scientific journals publishing in the area, the most prominent authors, the scientific production, and the most cited works. Section 3 gives a comprehensive overview of how a GAN works, the main training drawbacks, the most important GAN breakthroughs, and GANs for tabular data (where they are explained with a fine level of detail). In Section 4, we survey the main methods for synthetic data generation, dividing them into standard and Deep Learning methods and giving our considerations to all of them. Then, in Section 5, we discuss the evaluation of the quality of synthetic samples. In Section 6, the information covered in the previous sections is condensed and schematized so that it becomes easier to see the big picture. Finally, in Section 7, we present the main conclusions of this survey.

## 2. Literature Review

To analyze the state-of-the-art in what concerns GANs used for synthetic data generation, as well as synthetic data generation methods, we collected data from four major bibliographic databases—Web of Sciences (WoS), Scopus, IEEE Xplore, and ACM Digital

Library. As such, the query used contains keywords related to both GANs (in the context of synthetic data generation) and synthetic data. The query used is shown in Listing 1.

**Listing 1.** Query used to search the WoS, Scopus, IEEE, and ACM databases.

```
((“generative adversarial network” OR “GAN” OR “adversarial neural”
OR “adversarial machine learning”) AND “synthetic” AND (“sample” OR “data”))
OR (“synthetic” AND (“sample” OR “data”))
```

At first, running the search query without any additional filters returned a considerably high number of results in all four databases. Therefore, we determined that the query should only be applied to the title field—the number of results decreased with this restriction. Because this work is not an exhaustive literature review, the dates were also constrained to be equal to or greater than 1 January 2010, and the language in which the documents were written had to be English (however, IEEE Xplore and the ACM Digital Library did not offer this filter). Table 1 shows the filtering process just described, with the exact number of documents returned at each step.

**Table 1.** Filtering process. The search query in Listing 1 was used across four different databases, and three filters were applied to decrease the initial high number of results.

| Database | No Filters | Field = Title | Date = 1 January 2010 to 31 December 2022 | Language = English |
|---|---|---|---|---|
| WoS | 167,419 | 2460 | 1699 | 1681 |
| Scopus | 1,168,662 | 3625 | 2457 | 2401 |
| IEEE | 44,887 | 731 | 562 | No filter |
| ACM | 31,463 | 65 | 57 | No filter |

Once the documents were filtered, a general analysis was conducted on the resulting dataset—2706 distinct documents were found across the four databases (see Figure 1, which shows a graphical representation of the respective databases of the filtered works).

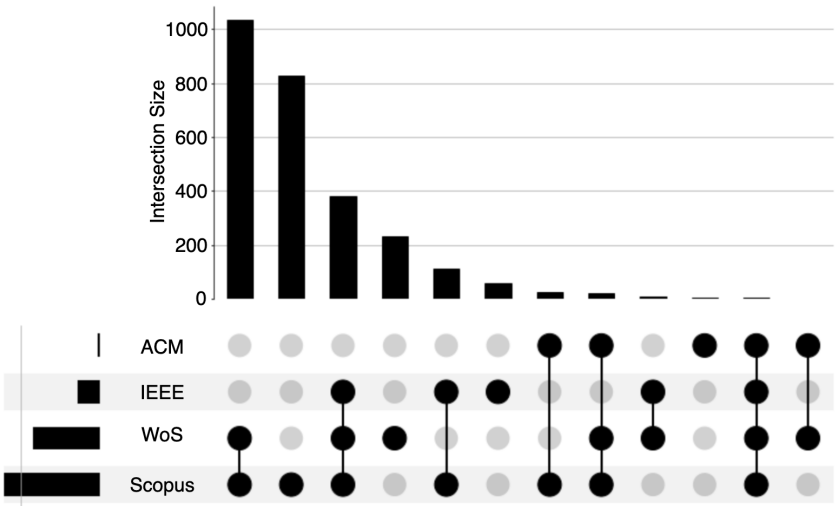
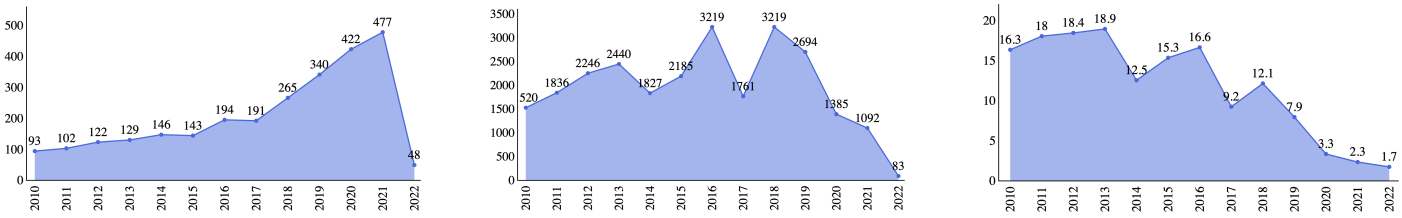



**Figure 1.** UpSet plot representing the number of documents in each database and the common works across the databases from the filtered results. The bar chart represents the number of documents in each database or in the intersection of databases. The plot immediately below the bar chart represents the intersection of works. If the same document was returned in the Web of Sciences, Scopus, and IEEE Xplore databases, it is denoted by black dots in the WoS, Scopus, and IEEE rows.

We start by looking at the annual scientific production (total number of works produced), the total number of citations, and the average number of citations (the total number

of citations in a year, divided by the respective number of documents). Figure 2 shows three line charts, each representing the annual values of the three aforementioned measures.

Annual scientific production has been increasing over the past decade, with a massive dip only in 2022, as we are at the beginning of the year at the time of writing. The same does not happen with the number of citations, as they have been steadily decreasing since 2018. To complement these two charts, we have included another chart showing the average number of citations per year, which has been decreasing in recent years.




(**a**) (**b**) (**c**)

**Figure 2.** Line charts (filtered documents). (**a**) Total number of documents per year. (**b**) Total number of citations per year. (**c**) Average number of citations per year.

To enable new researchers in this field of studies to have an overview of the subject, showcasing the main publication sources, the most relevant authors, and the highly cited works can be quite useful. As such, the main scientific journals, books, or conferences in which the filtered documents were published are first analyzed. To support this task, a treemap was created—see Figure 3. The treemap clearly shows that Lecture Notes in Computer Science is the Series with the most published documents (from the filtered documents), with two times more documents than the next two sources, the IEEE Conferences and the Journal of Applied Remote Sensing.

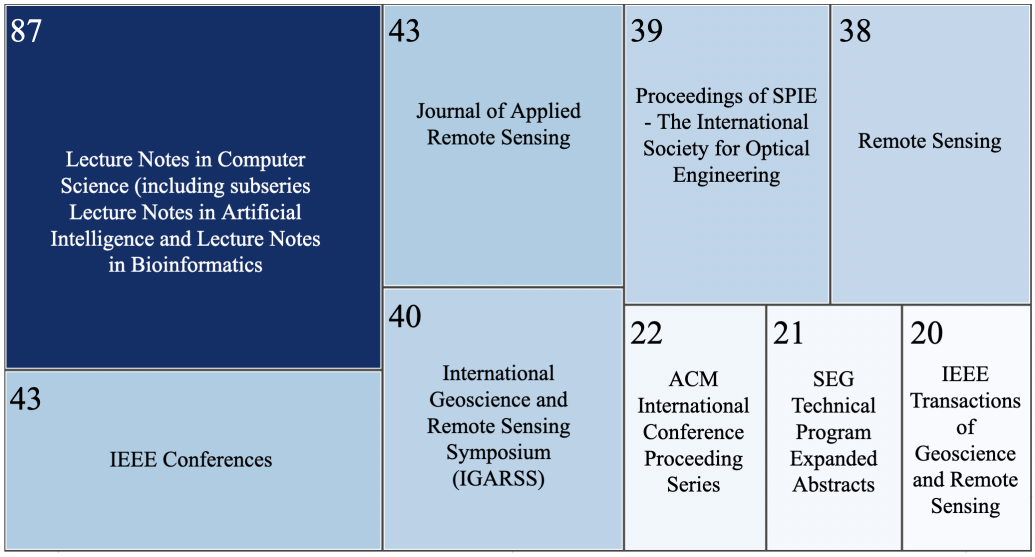



**Figure 3.** Treemap of the main publication sources from the filtered results.

Regarding the authors, there are three main insights that we have extracted: The authors with the highest number of publications, the ones that have been more productive in the years equal to or greater than 2020, and the ones with the most citations.

To identify the most productive authors, a plot was produced (see Figure 4) showcasing the ten authors with the most published works. As can be seen, the most productive author is Wang Y. (with 19 works), followed by Li X. (15 works) and Zhang Y. (14 works). Moreover, if the attention is shifted to the most productive authors in the years equal to or greater than 2020, Wang Y. remains the most published author, with eight works.

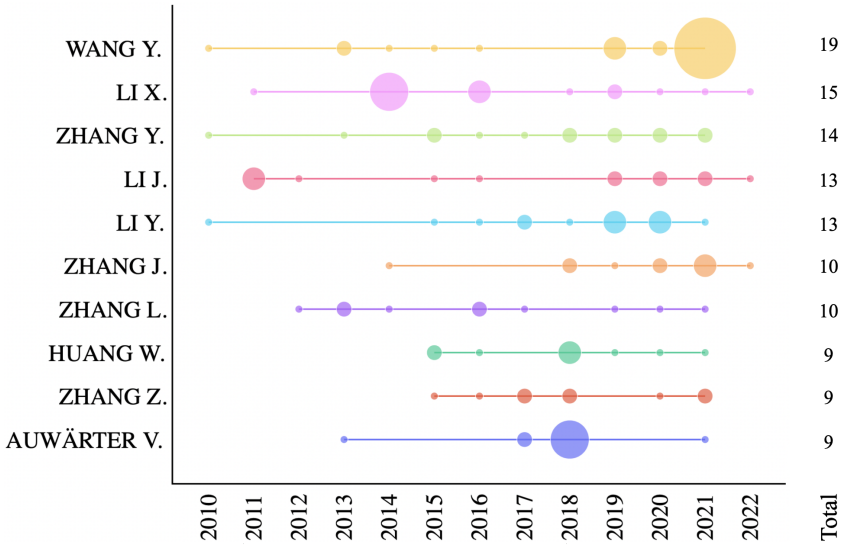



**Figure 4.** Most published authors. The *y*-axis represents the authors with the most publications by decreasing order—the total number of publications of each author is on the right. The *x*-axis represents the year of publication. The area of the circles is proportional to the number of publications of a given author in a given year.

Following this author is one of particular interest: Sergey I. Nikolenko, with six published works. His papers are of great interest as he writes about the early days of synthetic data, synthetic data for deep learning, and even where synthetic data is going. Moreover, he is the sole author of his published works and has also edited a book titled "Synthetic Data for Deep Learning", the text of which is based, to a considerable extent, on one of his published papers with the same name (this paper is cited at the beginning of Section 4, as it could not go unnoticed).

Interestingly enough, the authors mentioned previously are not the most cited ones. That place goes to Gupta A., Zisserman A., and Vedaldi A. with 676 citations each, followed by Alonso-Betanzos A., Sánchex-Maroño N., and Bolón-Canedo V., each with 392 citations. This happens because each triplet of authors has published a work that was heavily cited [11,12], respectively. Moreover, these are the most cited papers of the filtered results, and they are briefly described in a few lines.

To finish the authors' analysis, Table 2 contains information regarding the number of unique authors, the average number of authors per document, and a summary of the previous insights. It is interesting to note the high number of unique authors, 10,100, and that a typical paper has about four authors, on average.

**Table 2.** Authors' summary table.

| | |
|---|---|
| **Unique authors** | 10,100 |
| **Average number of authors per document** | 4.37 |
| **Most published works** | Wang Y. (19 works) Li X. (15 works) |
| **Most published works (≥2020)** | Wang Y. (6 works)<br>Nikolenko S. I. (4 works) |
| **Most cited** | Gupta A., Zisserman A., Vedaldi A. (676 citations)<br>Alonso A., Sánchex N., Bolón V. (392 citations) |

As for the most cited publications from the filtered works, the top five are briefly described. In [11], the most cited publication, the authors generated synthetic images with text to improve text detection in natural images. In [12], a review of feature selection methods on synthetic data is presented (synthetic datasets with an increasing number of

irrelevant features are generated to perform feature selection). Following these two works is [13], where a GAN is used to generate synthetic medical images to improve liver lesion classification (this work is at the top of Table 3). The next most cited paper, [14], concerns the field of remote sensing for forest biomass mapping and the importance of synthetic data in this field. Finally, in [15], the authors use synthetic data to improve seismic fault detection classification.

We have been looking for recent surveys on both synthetic data and GANs (since early 2019). Regarding the former topic, we found only one paper [16] in which the author (Nikolenko) explores the application of synthetic data outside of computer vision (currently the main area for synthetic data applications).

The latter topic is much more studied, as GANs have become increasingly important in recent years. Some of the papers focus on recent advances in GANs [17], others on the use of GANs in anomaly detection [18], or the challenges of GANs, solutions, and future directions [19]. These are not the only existing surveys concerning GANs, as this is not intended to be an exhaustive list. However, to the best of our knowledge, there is yet no work that explicitly combines and examines both topics. Therefore, this is intended to fill such a gap and provide helpful material to researchers interested in this area.

Graphical analysis was performed to identify the most common research areas, the institutions with the most publications, and those with the most citations. This analysis was performed using the Web of Sciences' results for two reasons. On the one hand, WoS is the only bibliographic database (out of the four used) that contains data on the research areas. On the other hand, institutions in the Scopus and IEEE Xplore databases are, in many cases, segmented by department, making the analysis quite difficult. However, the same does not happen in the returning records from the WoS database. In addition, the ACM Digital Library returned very few results compared to the other databases, so it is not representative of all the synthetic data research work.

When looking at the most common research areas (see Figure 5), Engineering and Computer Science are the most prominent, together accounting for about 45% of all publications. The fact that these two research areas are at the top is not surprising. However, it is interesting to note that, for example, Geology and Environmental Sciences and Ecology are the fifth and sixth most common research areas, which shows the widespread use of synthetic data in various fields.

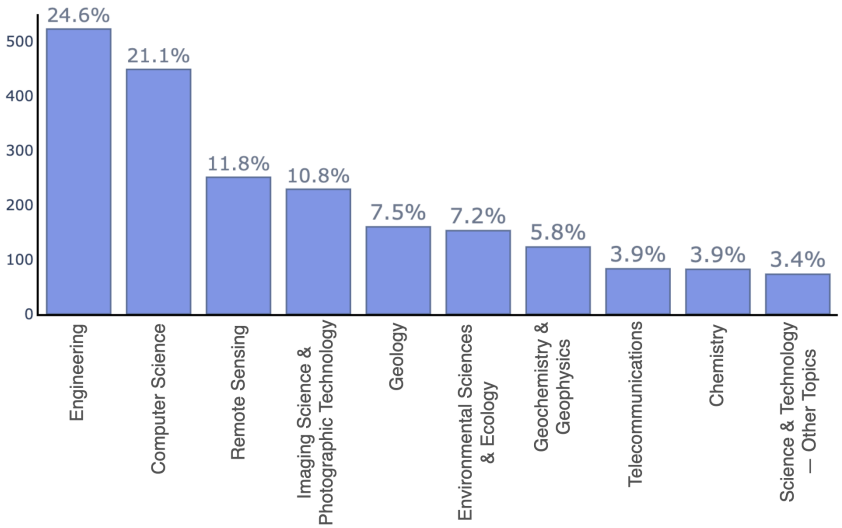



**Figure 5.** The top ten research areas of the filtered WoS results. The vertical axis represents the number of publications in a particular area. The percentages above the bars indicate the relative frequency of each area.

In what concerns the stronger institutions in terms of the number of publications and number of citations, two bar charts were constructed—see Figures 6 and 7. As can be seen

from the plots, the most frequent institutions are from the United States—e.g., University of California, University of Texas, or NASA. As there are more institutions than research areas, the relative frequencies are smaller and closer to each other, so there is not an institution (or few institutions) that stands out as much as the research areas. Nonetheless, the University of California, the Chinese Academy of Sciences, and the CNRS are leading in terms of the number of publications and citations.

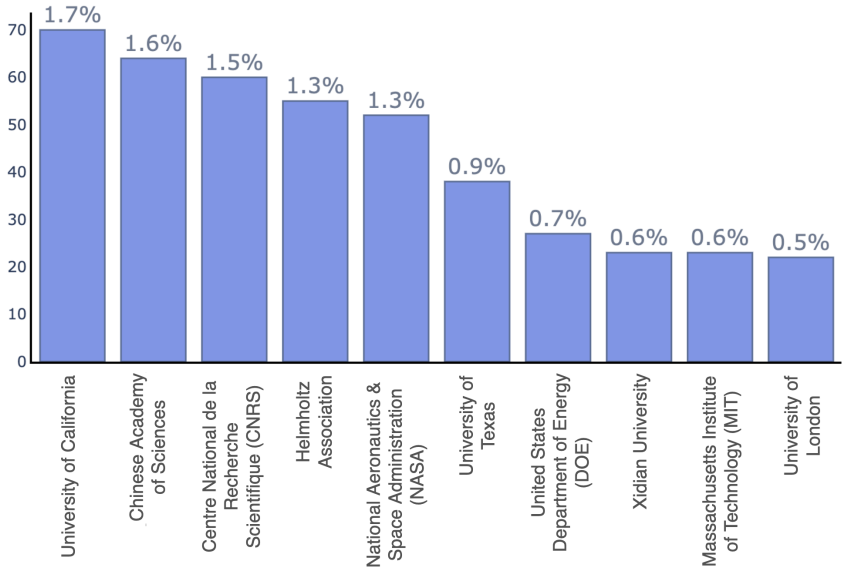



**Figure 6.** The institutions with the most publications from the WoS-filtered results. The vertical axis represents the number of publications. The percentages above the bars indicate the relative frequency of each institution in terms of publications.

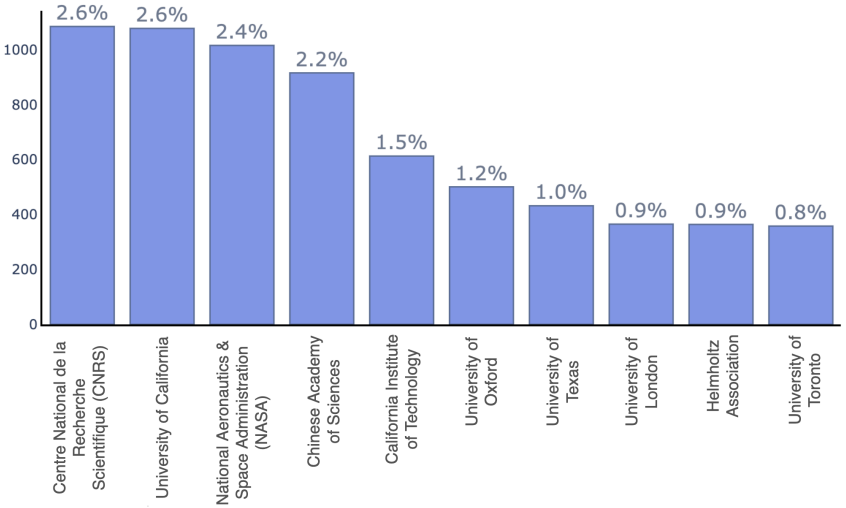



**Figure 7.** The institutions with more citations from the WoS-filtered results. The vertical axis represents the number of citations. The percentages above the bars indicate the relative frequency of each institution in terms of citations.

Since the main focus of this paper is to provide an overview of the literature concerning the generation of synthetic samples using GANs, a search of the most cited documents in the Web of Sciences bibliographic database on the above topic was carried out (the search was performed in the set of documents obtained after the filtering process—see Table 1). In Section 3, a comprehensive review of GANs and the most commonly used architectures are studied. Therefore, Table 3 can be supplemented by the next section.

It is interesting to note that most of the publications are from 2018 and 2019. We suspect that it took some time for GANs to mature and spread to different domains since they were first proposed in 2014. Thus, it is not far-fetched to imagine that GANs took about four years before a significant portion of the research community recognized their potential and began using them to generate synthetic data.

**Table 3.** Top 10 cited papers in the Web of Sciences database that use GANs.

| Title | GAN Architecture | Dataset | Year | Citations |
|---|---|---|---|---|
| Synthetic Data Augmentation using GAN for Improved Liver Lesion Classification [13] | DCGAN | Computed tomography (CT) images of 182 liver lesions | 2018 | 173 |
| Learning from Synthetic Data for Crowd Counting in the Wild [20] | SSIM embedding (SE) Cycle GAN | GCC dataset, UCF CC 50, Shanghai Tech A/B, UCF-QNRF, WorldExpo'10 | 2019 | 94 |
| Real-Time Monocular Depth Estimation using Synthetic Data with Domain Adaptation via Image Style Transfer [21] | DCGAN | KITTI, Make3D | 2018 | 53 |
| A Small-Sample Wind Turbine Fault Detection Method with Synthetic Fault Data using Generative Adversarial Nets [22] | CGAN | Wind turbine data collected from a wind farm in northern China | 2019 | 44 |
| Synthetic Data Generation for End-to-End Thermal Infrared Tracking [23] | CycleGAN, Pix2pix | KAIST, CVC-14, OSU Color Thermal, OTB, VAP Trimodal, Bilodeau, LITIV2012, VOT2016, VOT2017, ASL, Long-termInfAR | 2019 | 40 |
| Pixel-Wise Crowd Understanding via Synthetic Data [24] | SE CycleGAN | GCC dataset, UCF CC 50, Shanghai Tech A/B, UCF-QNRF | 2021 | 33 |
| Learning Semantic Segmentation from Synthetic Data: A Geometrically Guided Input-Output Adaptation Approach [25] | PatchGAN | KITTI, Virtual KITTI, SYNTHIA Cityscapes | 2019 | 25 |
| DeepSynth: Three-dimensional Nuclear Segmentation of Biological Images using Neural Networks Trained with Synthetic Data [26] | Spatially Constrained (SP) CycleGAN | 3D biological images | 2019 | 23 |
| Autoencoder-Combined Generative Adversarial Networks for Synthetic Image Data Generation and Detection of Jellyfish Swarm [27] | GAN | Jellyfish images | 2018 | 13 |
| DP-CGAN: Differentially Private Synthetic Data and Label Generation [28] | Differentially Private Conditional GAN (DP-CGAN) | MNIST | 2019 | 10 |

Unfortunately, the filtered documents lack some important works, both in terms of GAN architectures and synthetic data generation methods. Therefore, a snowballing procedure based on the references of the filtered results as well as the existing background knowledge of the authors was used to find and add relevant papers—in total, 62 extra works were found and added. These are explored in the following sections.

## 3. Generative Adversarial Networks

Generative adversarial networks (GANs)are a framework that uses an adversarial process to estimate generative deep learning models, proposed by Ian J. Goodfellow et al. [8] in 2014. These structures have been adapted and improved over the last years and are now very powerful. GANs are currently capable of painting, writing, composing, and playing, as we will see in this section.

Therefore, GANs are first analyzed in more detail in Section 3.1 to reveal how they work. In Section 3.2, the main training difficulties and their solutions are examined. Finally, in Section 3.3, the main GAN architectures are shown to demonstrate their capabilities, while in Section 3.4, GANs for tabular data are presented.

### *3.1. GANs under the Hood*

A GAN is constituted by two models: a generator model $G$ that tries to generate samples that follow the underlying distribution of the data. Nonetheless, these observations are suitably different from the ones in the dataset (i.e., they should not simply reproduce observations that already occur in the dataset). There is also a discriminator model $D$ that, given an observation (from the original dataset or synthesized by the generator), classifies it as fake (produced by the generator) (Typically, the models used for the generator and discriminator are neural networks. As such, we normally refer to $G$ and $D$ as networks. However, in theory, the models need not be a neural network. Indeed, in [8], the term "model" is used. Nonetheless, they note that the "adversarial modeling framework is most straightforward to apply when the models are both multi-layer perceptrons") or real. An important thing to consider is that $G$ and $D$ compete against each other. While $G$ generates similar data points to those in the original data, with the aim of deceiving the discriminator, $D$ attempts to distinguish the generated from the real observations.

To describe in more detail how the networks are trained, the training was split into the training of the discriminator and of the generator separately. Training the discriminator consists of creating a dataset with instances generated by $G$ and data points from the original dataset. The discriminator outputs a probability (continuous value between 0 and 1) that indicates whether a given observation came from the original data (0 means that the discriminator is 100% certain that the given example was synthesized, while 1 means the exact opposite).

The training of the generator is more complicated. $G$ is given as the input random noise (The term *latent space* is typically used to designate $G$'s input space.), commonly from a multivariate normal distribution, and the output is a data point with the same features of the original dataset. However, there is no dataset to inform whether a particular point in the latent space is mapped by $G$ into a reasonable or useful example. Therefore, the generator is only provided with a value from a loss function. This is usually the binary cross-entropy (The binary cross-entropy is mathematically defined as follows $-\frac{1}{n}\sum_{i=1}^{n} y_i log(p_i) + (1 - y_i) log(1 - p_i)$ where $y_i$ represents the label of an input sample, $p_i$ is the probability of $y_i$ coming from the original data, and $n$ is the number of examples. It is a measure of the difference between the ground truth and the computed probabilities, and it is used in the case where there are only two possible outcomes—the observation came from the original data or it was synthesized by the generator) between $D$'s output and a response vector of 1's (the instances synthesized by $G$ are all marked as coming from the original data).

Given the discriminator's feedback, i.e., the value of the loss function, the generator attempts to improve to better fool the discriminator. As training progresses, $G$ uses $D$'s output to generate better examples, i.e., examples that better resemble the real data distribution. As the data produced by $G$ becomes more realistic, $D$ also improves so that it can better determine whether a sample is real or synthetic. As such, both networks improve each other and, ideally, $G$ will be able to mimic the data distribution, and $D$ will be $\frac{1}{2}$ everywhere, i.e., the probability that $D$ distinguishes between a real observation and a generated one is as good as a random guess. In this ideal scenario, $G$ has succeeded in

recovering the distribution of the original data, completely fooling $D$. A GAN diagram is shown in Figure 8.

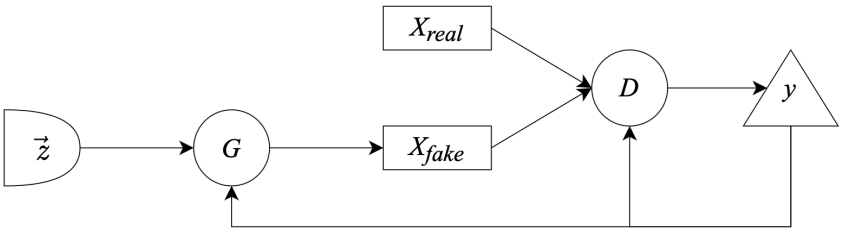



**Figure 8.** A GAN diagram. $G$ is given random noise $\vec{z}$, usually from a multivariate normal distribution, to generate a set of data points, $X_{fake}$. $D$ is provided with both the original data, $X_{real}$ and the generated data. $D$ outputs a label $y$, denoting if a given observation is fake (was produced by $G$) or real (came from the original data).

### *3.2. Main Drawbacks*

Although plain vanilla GANs—that is, the GANs in their simplest form, as we have been explaining—are quite strong ideas, they also have disadvantages. Namely, GANs are extremely difficult to train due to a number of factors that include the loss function, hyperparameters, or a generator that can easily fool the discriminator.

Oscillatory loss (instability) is a common problem that occurs during the training process. It is characterized by wild oscillations of the discriminator's and generator's loss, which should be stable over the long term. For the training process to be effective, the loss should stabilize or gradually increase/decrease over the long term. Unfortunately, in many cases, this is not what happens. Another problem with the loss function is the lack of information it usually provides (uninformative loss). For example, a commonly used generator's loss function is the binary cross entropy. This is a disadvantage because there is no correlation between the generator's loss and the quality of the output (not only in the specific case of the binary cross entropy). Hence, the training is sometimes difficult to monitor.

Another fairly common phenomenon is that the generator finds a small number of samples that fool the discriminator—this is called mode collapse. Having found such samples, the generator will focus only on them to minimize its loss function, while the discriminator remains confused during training because it cannot distinguish whether the instances are real or synthetic. Therefore, the generator is not able to produce other examples than this limited set. Figure 9 shows an example of mode collapse in a toy dataset.

Moreover, GANs have a significant number of hyperparameters. Thus, to create a well-performing GAN, a large number of hyperparameters must be tuned. It is possible to use grid search, but only for a limited subset of hyperparameters. Otherwise, the training time will be considerably long and the resource consumption extremely high.

Finally, there is the vanishing gradient problem, which may completely stop the GAN from further training, given that the gradients can be extremely small and not allow the weights to be updated further. This can occur if the discriminator is close to optimal, which allows it to accurately discern generated samples from real ones and causes the generator's train to fail.

These are the five most common problems encountered in GAN training—oscillating loss, mode collapse, uninformative loss, vanishing gradients, and hyperparameter-tuning. The above problems are broad and independent of domain and architecture. That is, they attempt to cover the range of possible GAN training drawbacks without being too specific about the loss-function or hyperparameters used (broad); they do not depend on the particular domain, whether it is live cell images or a tabular dataset of bank fraud (domain-agnostic); and finally, they do not depend on a particular GAN architecture (architecture-agnostic).

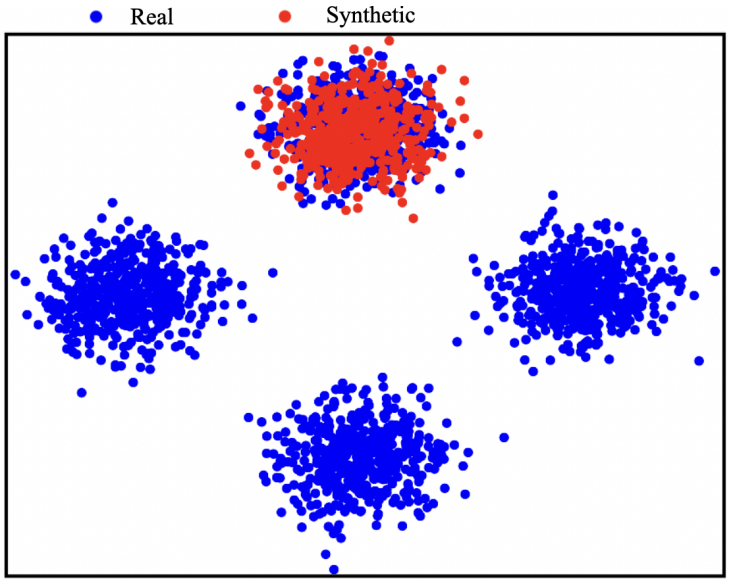



**Figure 9.** Graphical representation of mode collapse in a toy dataset consisting of random samples drawn from four Gaussian distributions with the same covariance matrix but different means (visible by the four separate clusters). The blue points correspond to real data points, whereas the red ones are synthesized by the generator. The generator has found a small number of samples (the ones in the upper cluster), so it does not learn beyond that. It will continue to produce samples in that range without seeing the overall distribution of the data, as it is enough to fool the discriminator.

### *3.3. GANs Come in a Lot of Flavours*

Since GANs were proposed, many researchers have considered them a powerful tool. As a result, they have been systematically modified and improved. The architecture of a GAN can be very problem-specific, and they are often modified or fine-tuned to serve a particular purpose. Hence, the literature on them is quite extensive, and thus, only the main highlights are shown in this paper. In the following paragraphs, the GAN architectures are arranged chronologically by year (in ascending order, i.e., earlier years are shown first), so two architectures created in the same year may not be arranged by month. Nonetheless, this can show the evolution of GANs up to the time of writing (February 2022).

**Conditional Generative Adversarial Network**, **CGAN**, is a GAN variant proposed by Mehdi Mirza and Simon Osindero in [29]. Suppose one is using a vanilla GAN on an image dataset with multiple class labels (e.g., the ImageNet dataset). The GAN has been properly trained and is ready to generate synthetic samples. However, it cannot sample an image of the desired class. For example, if one wants synthetic images of cars (assuming that images of cars were used in the training data), one cannot force a vanilla GAN to do so. This happens because there is no control over the latent space representation. That is, the GAN maps point from latent space to the original domain, but the features in the latent space are not interpretable by the user. As such, one does not know from which range of points to sample in order to produce examples of a certain class. This is an obvious disadvantage of using GANs in labeled datasets. An interesting idea is to make the GAN dependent on a class label, which allows generated data to be conditioned on class labels. That is, given a labeled dataset, the CGAN is trained using the data instances and their respective labels. Once trained, the model can generate examples that depend on a class label selected by the user. For example, if a hypothetical dataset has three classes—"low", "medium", "high"—the CGAN is trained with both the instances and their associated labels. After the learning process is complete, the user can choose to generate samples of only "low" and "high" classes by specifying the desired label. A diagram representing a CGAN is shown in Figure 10.

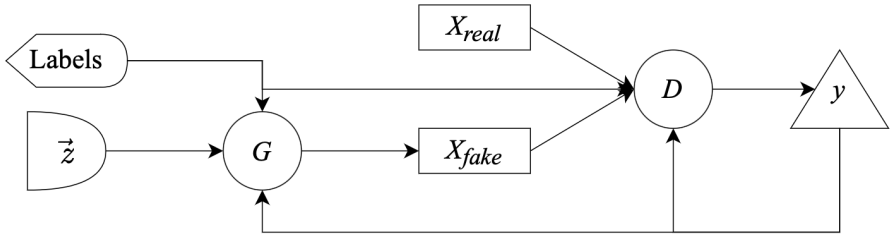



**Figure 10.** A diagram of a CGAN. The CGAN is similar to a vanilla GAN (see Figure 8), but the generator, G, and the discriminator, D, are conditioned on class labels.

Despite the importance of CGAN, with its clear advantage of being able to draw a sample from a user-selected class, back in 2014, the generation of synthetic images had a lot of room for improvement. As such, a growing number of GAN architectures focused on image generation were proposed in the following years.

**Deep Convolutional Generative Adversarial Network**, **DCGAN**, is a GAN architecture that combines convolutional layers (A convolutional layer is a layer that uses a convolution operation. A convolution, in terms of computer vision tasks, consists of a filter (represented by a matrix) that slides through the image pixels (also represented by a matrix) and performs matrix multiplication. This is useful in computer vision tasks because applying different filters to an image (by means of a convolution) can help, for example, detect edges, blur the image, or even remove noise), which are commonly used in computer vision tasks, with GANs. Radford et al., in [30], have brought together the success of Convolutional Neural Networks (CNNs) in supervised learning tasks with the then emerging GANs. Nowadays, the use of convolutional layers in GANs for image generation is quite common, but at that time, 2016, this was not the case. Therefore, the use of convolutional layers in the GAN structure is still a powerful tool for handling image data.

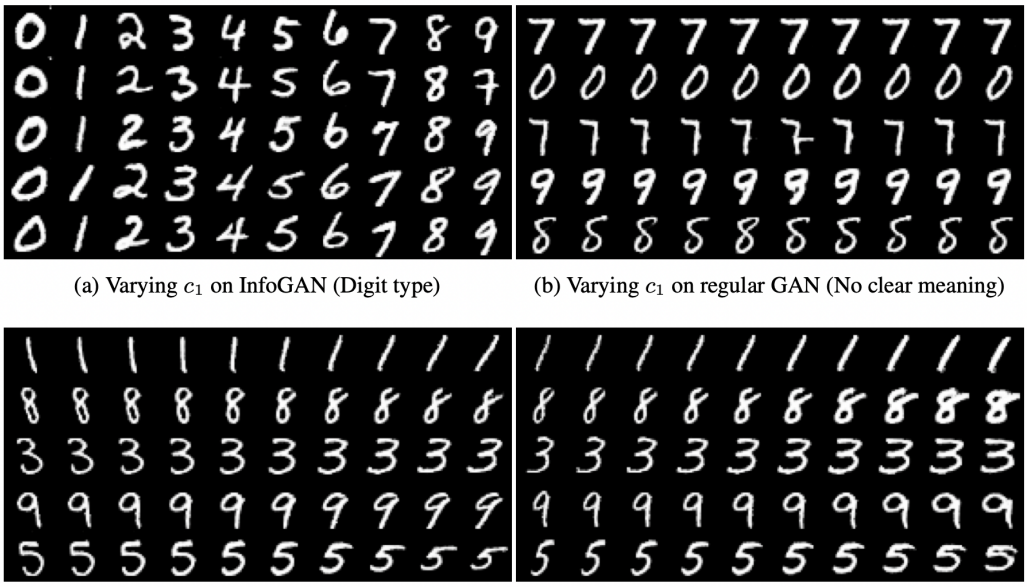
Thus, the DCGAN was able to enhance the generated images by using convolutional layers. However, the features in the latent space had no semantic meaning. That is, it was not possible to change the values of a feature in latent space and predict what that change would do to the image (e.g., rotation, widening).

**Information Maximizing Generative Adversarial Network**, **InfoGAN**, is a GAN extension proposed by Chen et al. in [31], that attempts to learn disentangled information. That is, to give semantic meaning to features in the latent space (see Figure 11). InfoGAN can successfully recognize writing styles from handwritten digits in the MNIST dataset, detect hairstyles or eyeglasses in the CelebA dataset, or even background digits from the central digit in the SVHN dataset.

The GAN architectures presented so far can be quite time-consuming and use a high amount of computing resources to train. Given a large number of hyperparameters and a large number of training samples, the training process could be prohibitively expensive due to the training time and resources required.

**Coupled Generative Adversarial Networks**, **CoGAN**, proposed in [32] by Ming-Yu Liu and Oncel Tuzel, use a pair of GANs instead of only one GAN. The CoGAN was used to learn the joint distribution of multi-domain images, which was achieved by the weight-sharing constraint between the two GANs. In addition, sharing weights requires fewer parameters than two individual GANs, which, in turn, results in less memory consumption, less computational power, and fewer resources.

The focus on image generation continued, and in 2016, the AC-GAN and the StackGAN architectures were introduced to provide improvements in synthetic image generation.



(a) Varying $c_1$ on InfoGAN (Digit type) (b) Varying $c_1$ on regular GAN (No clear meaning)

(c) Varying $c_2$ from $-2$ to 2 on InfoGAN (Rotation) (d) Varying $c_3$ from $-2$ to 2 on InfoGAN (Width)

**Figure 11.** The semantic meaning InfoGAN adds to the latent variables in the MNIST dataset. In (**a**), varying the latent variable $c_1$ leads to a digit change (from 0 to 9), while in (**b**), a regular GAN does not add meaning to its latent variables. In (**c**), the variation of $c_2$ leads to the rotation of digits. Finally, in (**d**), variation $c_3$ controls the width of the digits. Image taken from [31].

**Auxiliary Classifier Generative Adversarial Network**, **AC-GAN** [33], is a GAN extension proposed by Odena et al. that modifies the generator to be class dependent (it takes class labels into account) and adds an auxiliary model to the discriminator whose purpose is to reconstruct the class label. The results in [33] show that such an architecture can generate globally coherent samples that are comparable, in diversity, to those of the ImageNet dataset (see Figure 12).

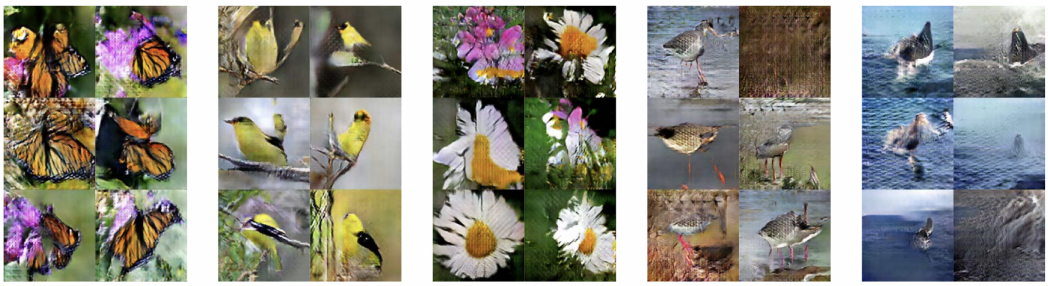



**Figure 12.** Images of five distinct classes generated by the AC-GAN. Nowadays, the detail in the images is far superior to the one provided by the AC-GAN. Image taken from [33].

**Stacked Generative Adversarial Network**, **StackGAN**, proposed in [34] by Zhang et al., is another extension of GANs that can generate images from text descriptions. This generation of photorealistic images is decomposed into two parts. First, the STAGE-I GAN sketches a primitive shape and colors based on the input text. Next, the Stage-II GAN uses the same text description as the STAGE-I GAN and its output as input and generates high-resolution images by refining the output images by STAGE-I GAN. Their work has led to significant improvements in image generation.

Despite improving the quality of the generated images, adding semantic meaning to the latent features, and reducing memory consumption and training time, the training itself was still difficult due to mode collapse and uninformative loss metrics.

**Wasserstein Generative Adversarial Networks**, **WGAN**, is an alternative to traditional GAN training. The WGAN proposed by Arjovsky et al. in [35] is a GAN extension that modifies the training phase such that the discriminator, called the critic, is updated more often than the generator at each iteration $i$, where $i$ is defined by the user. This change to GAN training avoids mode collapse and provides a meaningful loss metric that correlates with the generator's convergence and sample quality.

Returning to image generation, an interesting idea is to transfer an image from one area to another. For example, let us take a landscape image and "merge it" with an image of a Monet painting so that the landscape image has the style of a Monet painting.

**Cycle-Consistent Generative Adversarial Network**, **CycleGAN**, is a GAN extension for image-to-image translation without paired data. Zhu et al. proposed, in [36], an approach to translate an image from a domain $X$ to a domain $Y$ when no paired images are available. The CycleGAN consists of two generators, $G$ and $F$, and two discriminators, $D_X$ and $D_Y$. $G$ maps an image from $X$ to $Y$, and $D_Y$ tries to determine whether it is from the original dataset or synthesized. Similarly, $F$ maps an image from $Y$ to $X$, and $D_X$ determines whether it is real or generated by $F$. In addition to the four networks, the cycle consistency loss metric was also introduced to ensure that translating an image from $X$ to $Y$ and then from $Y$ to $X$ yields a very similar image to the original one. Figure 13 shows the image-to-image translation capabilities of CycleGAN.

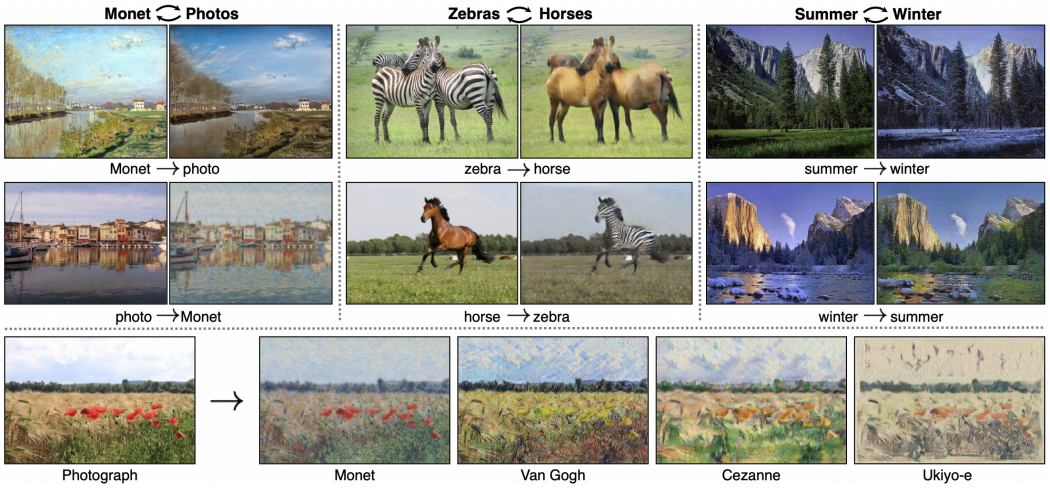



**Figure 13.** Given any two image collections, the CycleGAN learns to automatically "translate" an image from one domain into the other and vice versa. Example application (bottom): using a collection of paintings of famous artists, the CycleGAN renders a user's photograph in their style. Image taken from [36].

To date, GAN architectures have focused on image generation and translation, training stabilization, and time or have been tied to class labels. Nonetheless, there is an interesting application of GANs to music generation.

**Multi-track sequential GAN**, **MuseGAN**, proposed by Dong et al. in [37] is a GAN architecture for music generation. This is quite different from generating images or videos since music has a temporal dimension, is usually composed of multiple instruments, and musical notes are often grouped into chords. Although the music generated is not as good as that produced by professional musicians, the results were quite promising, and the MuseGAN model had some interesting properties.

In late 2017 and throughout 2018, the quality of image-generated data improved greatly with the introduction of ProGAN, SAGAN, and BigGAN architectures.

**Progressive growing of Generative Adversarial Networks**, **ProGAN**, is a technique that helps stabilize GAN training by progressively increasing the resolution of generated images. Proposed in [38] by Karras et al., the ProGAN accelerates and stabilizes training by, first, constructing a generator and a discriminator that produce images with few pixels.

Then, layers corresponding to higher resolutions are added in the training process, allowing the creation of high-quality images. Figure 14 shows images generated with the ProGAN.

**Figure 14.** Images generated using ProGAN. Notice the level of detail when compared to the ones generated from AC-GAN (Figure 12). Image taken from [38].

**Self-Attention Generative Adversarial Networks**, also known as SAGAN, improve on previous GAN structures by maintaining long-range relationships within an image rather than just local points [39]. Zhang et al. have found that using spectral normalization improves the training dynamics of the generator. In addition, the discriminator can assess whether highly detailed features in distant image regions match each other. When this architecture was proposed, the authors were able to improve both the Inception Score [40] and the Fréchet Inception Distance[41] (two widely used metrics to evaluate synthetic image data) on the ImageNet dataset.

**Big Generative Adversarial Network**, **BigGAN**, proposed by Brock et al. [42], is a type of GAN architecture that upscales existing GAN models and produces high-quality images (see Figure 15). BigGAN has also demonstrated how to train GANs at a large scale by introducing techniques that detect training instability. At the time of BigGAN's introduction, its performance was significantly better than that of other state-of-the-art structures.

**Figure 15.** Class-conditional samples generated by BigGAN. Image taken from [42].

As seen previously, the image quality has improved considerably (compare Figure 12 with Figures 14 and 15, for example). However, there were still some limitations in the images generated. Although the GAN architectures provided extremely realistic images, it was still difficult to understand various aspects of the image synthesis process [43].

**Style-based Generative Adversarial Networks**, **StyleGAN**, proposed by Karras et al. in [43], explores an alternative generator architecture based on style transfer. The focus is not on generating more realistic images but on having better control over the generated image. This new architecture is able to learn to separate high-level features and stochastic

variation. In fact, the new generator improves the quality metrics over the state-of-the-art, untangles the latent variables better, and has better interpolation properties.

Two other different ideas than those shown so far, but also very interesting, were proposed in 2019. The first is about turning a user's sketch into a realistic image. The second is about automatically completing an incomplete image in a plausible way.

**GauGAN** [44], a model proposed by NVIDIA Research that allows users to sketch an abstract scene and then turn it into a detailed image. Users can also manipulate the scene and label each element. This is achieved through the use of a spatially-adaptive normalization layer whose purpose is to aid in the generation of photorealistic images when a semantic layout is given as input.

**Pluralistic Image Inpainting GAN**, **PiiGAN**, proposed by Weiwei Cai and Zhanguo Wei [45], attempts to fill in large missing areas in an image. Unlike other Deep Learning methods that try to achieve a single optimal result, PiiGAN has a new style extractor that is able to extract the style features from the original images. As shown in [45], PiiGAN can produce images of better quality and greater variety than other state-of-the-art architectures that match the context semantics of the original image. Figure 16 shows the capabilities of PiiGAN.

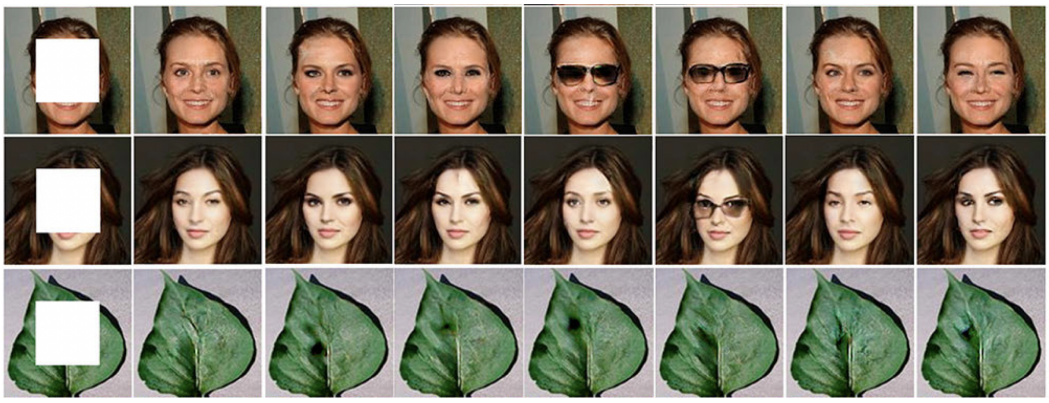


**Figure 16.** Examples of inpainting results produced by PiiGAN on two faces and a leaf. On the left column is the input image (with the center pixels removed). The images in the remaining columns are outputs of the PiiGAN. Image taken from [45].

A more recent architecture, introduced in 2021, is the Multy-StyleGAN, which highlights the capabilities of GANs in various image domains—in this case, biology.

**Multi-StyleGAN**, proposed by Prangemeier et al. [46], is a novel GAN architecture used to study the dynamic processes of life at the level of single cells. Since acquiring images to study such processes is costly and complex, the Multi-StyleGAN is a descriptive approach that simulates microscopic images of living cells. As shown by the authors, the proposed architecture is capable of capturing the underlying biophysical factors and temporal dependencies.

As shown in the previous paragraphs, the major breakthroughs of GANs are focused on imaging generation. Despite their enormous success in this area, GANs can be used in other areas as well. As can be seen in Section 3.1, there are no restrictions on whether the dataset must be an image, a video, music, or an ordinary tabular dataset. Nonetheless, different types of architectures must be considered depending on the task. Image data does not have the same characteristics as music or tabular data, so different types of layers, activation functions, or training procedures must be selected accordingly. That being said, there are some best practices that can be used depending on the data at hand, but the architecture of a GAN currently seems to be as much an art as a science. In the next subsection, we take a closer look at three GAN structures used to generate tabular data.

### *3.4. GANs for Tabular Data*

As seen in Section 3.3, GANs are widely and successfully used for image generation tasks. However, many datasets have a tabular format, and the most popular GAN architectures cannot be used in such a setting because tabular data has unique properties.

First, continuous and categorical features are present in most tabular datasets. Since image data consists solely of numerical features (the pixels), GANs used for image generation tasks cannot accommodate the different types of variables. Second, non-Gaussian and multimodal distributions are quite common in tabular datasets. Numerical features in tabular data may have multiple modes and follow a non-Gaussian distribution, which must be considered when generating synthetic data. Third, highly imbalanced categorical variables are common. This can lead to severe mode-collapse and insufficient training for the minority classes. Finally, it is easier for a trivial discriminator to distinguish between real and fake data when it learns from sparse one-hot-encoded vectors since it takes into account the sparsity of the distribution rather than checking the overall authenticity of the sample.

In the following sections, we detail three important GAN architectures used to overcome the above problems. The TGAN architecture was introduced in 2018, followed by the CTGAN architecture in 2019, which is an evolution of the TGAN architecture and was proposed by the same authors. This was followed in 2021 by the TabFairGAN, which was intended to dethrone the two aforementioned GANs in terms of the quality of synthetic tabular data generation. We believe the detailed explanations that follow can shed some light on a topic that is as not as well disseminated in the literature, as far as we are aware—the use of GANs to generate tabular data rather than image data.

#### 3.4.1. TGAN

TGAN was proposed in 2018 by Lei Xu and Kalyan Veeramachaneni [47] as a GAN architecture for synthesizing tabular data. Given a dataset, $D$, which is already split into trainset, $D_{train}$, and testset, $D_{test}$, the aim of the TGAN is twofold: given a machine learning model, its accuracy on $D_{test}$ when trained on the $D_{train}$ should be similar to its accuracy, also on $D_{test}$, but when trained using $D_{synth}$, which is the synthetic data (machine learning efficacy); the mutual information between each pair of columns in $D$ and $D_{synth}$ should be similar.

To achieve these goals, first, it is important to transform the data. A GAN usually consists of two neural networks, so it is crucial to properly represent the data before feeding it as input. This problem is addressed by applying mode-specific normalization for numerical variables and smoothing for categorical variables.

Mode-specific normalization is used to handle non-Gaussian and multimodal distributions. It fits a Gaussian mixture model (GMM), which models a distribution as a weighted sum of Gaussian distributions to each numerical variable and calculates the probability that a sample from a numerical column comes from each of the Gaussian distributions. These probabilities are then used to encode the values of the rows corresponding to the numerical features. More formally, let $\{N_1, N_2, \ldots, N_p\}$ represent the numerical columns of a tabular dataset $D$. A GMM with $m$ components is fitted to each numerical variable, $N_i$. The means and standard deviations of the $m$ Gaussian distribution are represented by $\mu_i^{(1)}, \mu_i^{(2)}, \ldots, \mu_i^{(m)}$ and $\sigma_i^{(1)}, \sigma_i^{(2)}, \ldots, \sigma_i^{(m)}$, respectively. The probability of $x_{i,j}$ (the value at row $i$ and column $j$) coming from each of the $m$ Gaussian distributions is given by a vector $u_{i,j}^{(1)}, u_{i,j}^{(2)}, \ldots, u_{i,j}^{(m)}$. Finally, $x_{i,j}$ is normalized as $v_{i,j} = \frac{x_{i,j} - \mu_i^{(k)}}{2\sigma_i^{(k)}}$, where $k = argmax_k u_{i,j}^{(k)}$ and $v_{i,j}$ is clipped to $[-0.99, 0.99]$, and $u_i, v_i$ are used to encode $x_i$.

Smoothing of the categorical variables is achieved by representing them as one-hot-encoded vectors, adding noise to each dimension (drawn from a uniform distribution), and renormalizing the vector. After applying mode-specific normalization to the numerical columns and smoothing the categorical ones, the data are ready to be fed into the TGAN. The generator is a Long-Short Term Memory (LSTM) network that generates the numeric

variables in two steps (In the first step, $v_i$ is generated, and $u_i$ is generated in the second step) and the categorical variables in one step. A fully connected neural network is used as the discriminator. A diagram of a TGAN is shown in Figure 17.

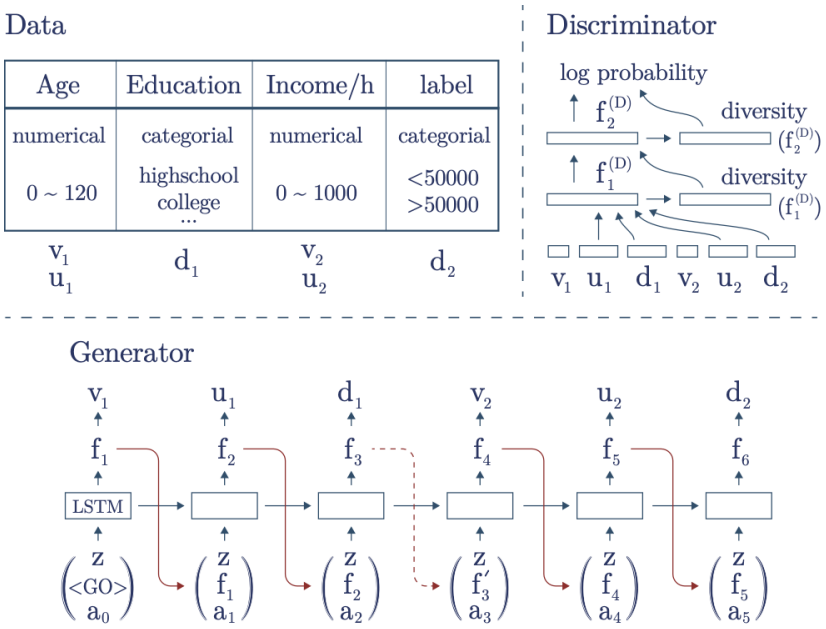



**Figure 17.** Diagram of a TGAN used in a toy example with 2 continuous and 2 discrete variables. Image taken from [47].

The TGAN was evaluated, in [47], with respect to machine learning efficacy and the preservation of correlation (the two aforementioned aims of the TGAN) and compared with other data synthesis models. Regarding machine learning efficacy, five models were evaluated in terms of accuracy and Macro-F1, namely, Decision Trees, Linear Support Vector Machines, Random Forests, AdaBoost, and Multi-Layer Perceptrons, on three different datasets. It was found that while the machine learning models generally performed better when trained on the real dataset, the average performance difference between the real and synthetic data was 5.7%. This suggests that the TGAN performs quite well (The authors compared the TGAN with a Gaussian Copula (GC) and a Bayesian Network (BN-Co), which showed a drop in performance of 24.9% and 43.3%, respectively). Moreover, the TGAN was able to maintain the ranking of the ML models. As for the preservation of correlation between any two pairs of variables, the TGAN was able to successfully capture this correlation.

#### 3.4.2. CTGAN

The CTGAN, also proposed by Lei Xu and Kaylan Veeramachaneni et al. [48] in 2019, is an improvement over TGAN. The objectives of CTGAN are almost the same as those of TGAN. The difference is that CTGAN is more ambitious, and instead of just preserving the correlation between any pair of columns in the synthetic data, it aims to preserve the joint distribution of all columns.

As for the transformations of the input data, they are similar to those presented for the TGAN model. To transform the numerical columns, a variational Gaussian mixture model (VGM) is used instead of a GMM. The difference is that the VGM estimates the number of modes for each numerical column, unlike in the TGAN, where the number of modes is predefined and is the same for each numerical column. In addition, the continuous values are represented as a one-hot vector indicating the mode and a scalar indicating the value within the mode (e.g., if the VGM has an estimated three modes and a given value $x_{i,j}$ has a greater probability of coming from mode 2, then the one-hot-encoded vector would be

$\vec{\beta} = (0, 1, 0)$ and the value within the mode would be given by $a_{i,j} = \frac{x_{i,j} - \mu_2}{4\sigma_2}$, where $\mu_2$ is the mean of the Gaussian distribution corresponding to the second mode, and $\sigma_2$ its standard deviation). The categorical features are only one-hot-encoded without adding noise.

To allow the CTGAN to deal with unbalanced discrete columns, the authors used a conditional generator that can generate synthetic rows that depend on any of the discrete columns. Further, a technique called training by sampling was proposed, allowing the CTGAN to uniformly examine all possible discrete values.

To integrate the conditional generator in the GAN architecture, it is necessary to properly prepare the input. This is accomplished by using a conditional vector, which specifies that a given categorical column must be equal to a certain value (from the set of the possible values for that particular column). Further, the generator loss is modified so that it learns to map the conditional vector into the one-hot-encoded values. The conditional vector consists of a simple transformation to the one-hot-encoded vectors. Supposing that a dataset with 3 discrete columns, $D_1 = \{0, 1, 2\}, D_2 = \{0, 1\}, D_3 = \{0, 1, 2\}$, is given, and the condition that is indicated is $D_2 = 1$, the conditional vector would be

$$\vec{cond} = (\underbrace{0, 0, 0}_{D_1}, \underbrace{0, 1}_{D_2}, \underbrace{0, 0, 0}_{D_3})$$

where the first three entries correspond to the one-hot-representation of $D_1$, the fourth and fifth entries correspond to the one-hot representation of $D_2$, and the last three entries correspond to the one-hot representation of $D_3$. The conditional generator is then forced to map the conditional vector into the one-hot-encoded ones by adding the cross entropy to its loss function.

Training by sampling is a technique that ensures that the conditional vector is properly sampled so that the CTGAN can uniformly examine all possible values in discrete columns. This is performed by randomly selecting a discrete column, constructing the probability mass function over the possible values for the selected column (the probability mass of each value is the logarithm of its frequency), and only then computing the conditional vector. A diagram of a CTGAN is shown in Figure 18 (the conditional generator and the discriminator are both fully-connected networks).

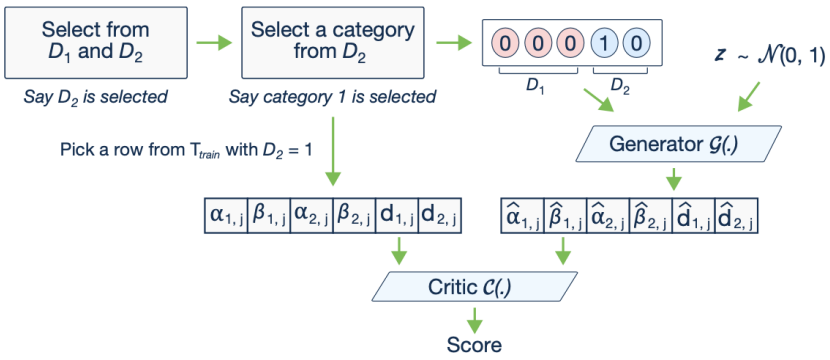



**Figure 18.** Diagram of a CTGAN. Image taken from [48].

To evaluate the CTGAN, the authors in [48] have used seven simulated datasets and eight real datasets. In the simulated datasets, the likelihood fitness metric was computed to evaluate performance, which is possible since the distribution of the data is known. In what concerns the real datasets, the machine learning efficacy was used to evaluate performance (it is not possible to compute the likelihood fitness metric in real datasets because the distribution of the data is unknown). The CTGAN was also compared with other generative models, namely CLBN [49], PrivBN [50], MedGAN [51], VeeGAN [52], and TableGAN [53]. It was found that in real datasets, the CTGAN outperformed all other models in terms of machine learning efficacy. In simulated datasets, the CTGAN performed quite well in terms of the likelihood fitness metric, although it was not able to outperform all other models.

Finally, an ablation study was conducted with the goal of evaluating the utility of mode-specific normalization, conditional generator, and training by sampling. The results showed that if the mode-specific normalization was replaced by either a Gaussian mixture model with five modes (GMM5), a GMM10 or a min-max normalization, the losses in performance (regarding machine learning efficacy) in the real datasets would be of $-4.1\%$, $-8.6\%$, and $-25.7\%$, respectively. In what concerns the training by sampling, if removed, the performance would decrease by 17.8%. If the conditional generator was removed, the performance would drop by 36.5%. Therefore, the techniques introduced in CTGAN, namely, mode-specific normalization, training by sampling, and the conditional generator, are very important for generating high-quality tabular data.

#### 3.4.3. TabFairGAN

TabFairGAN, proposed in 2021 by Amirarsalan Rajabi and Ozlem Ozmen Garibay [54], is a WGAN with a gradient penalty. As with TGAN and CTGAN, it is crucial to represent the data correctly before entering it as input to the TabFairGAN. Thus, Rajabi and Garibay used one-hot-encoding to represent the categorical features. A quantile transformation was used for the numerical features:

$$c_i' = \Phi^{-1}(F(c_i))$$

where $c_i$ is the $i^{th}$ numerical feature, $F$ is the cumulative distribution function (CDF) of the feature $c_i$, and $\Phi$ is the CDF of a uniform distribution.

In what concerns the network structure, the generator is formally described as:

$$\begin{cases} h_0 = z \\ h_1 = ReLU(FC_{l_w \to l_w}(h_0)) \\ h_2 = ReLU(FC_{l_w \to N_c}(h_1)) \oplus gumbel_{0.2}(FC_{l_w \to l_1}(h_1)) \oplus \\ gumbel_{0.2}(FC_{l_w \to l_2}(h_1)) \oplus \ldots \oplus gumbel_{0.2}(FC_{l_w \to N_d}(h_1)) \end{cases}$$

where $z$ is a latent variable drawn from a standard multivariate normal distribution, $ReLU$ is the rectified linear unit activation function, $FC_{a \to b}$ denotes a fully connected layer with input size $a$ and output size $b$, $l_w$ is the dimension of an input sample, $N_c$ is the number of numerical columns, $N_d$ is the number of categorical columns, $l_i$ is the dimension of the one-hot-encoded vector of the $i^{th}$ categorical column, $\oplus$ denotes the concatenation of vectors, and $gumbel_\tau$ is the Gumbel softmax with parameter $\tau$ (a continuous distribution that approximates samples from a categorical distribution and uses backpropagation).

In what concerns the critic (discriminator), its architecture can be formally described as follows:

$$\begin{cases} h_0 = X \\ h_1 = LeakyReLU_{0.01}(FC_{l_w \to l_w}(h_0)) \\ h_2 = LeakyReLU_{0.01}(FC_{l_w \to l_w}(h_1)) \end{cases}$$

Here $X$ denotes the output of the generator or the transformed real data, and $LeakyReLU_\tau$ represents the leaky rectified linear unit activation function with slope $\tau$. Figure 19 shows a diagram of the TabFairGAN. An initial fully connected layer (with $ReLU$ activation) constitutes the generator, followed by a second layer that uses $ReLU$ for numerical attributes and Gumbel softmax for one-hot-encoding of the categorical features. In the last layer, all the attributes are concatenated, producing the final generated data. The critic is constituted by fully connected layers with the $LeakyReLU$ activation function.

TabFairGAN was evaluated in terms of machine learning efficacy (the F1-score and accuracy metrics were used) using three machine learning models, namely, decision trees, logistic regression, and multi-layer perceptron (MLP) in the UCI Adult Income Dataset. The results were compared with two other state-of-the-art models, the TGAN and the CTGAN. TabFairGAN was found to perform better than TGAN and CTGAN on all machine learning models and metrics used, with the exception of MLP, where CTGAN performed better than TabFairGAN in terms of accuracy (but not in the F1-score). Hence, the TabFairGAN is quite effective in generating data similar to the real tabular data.

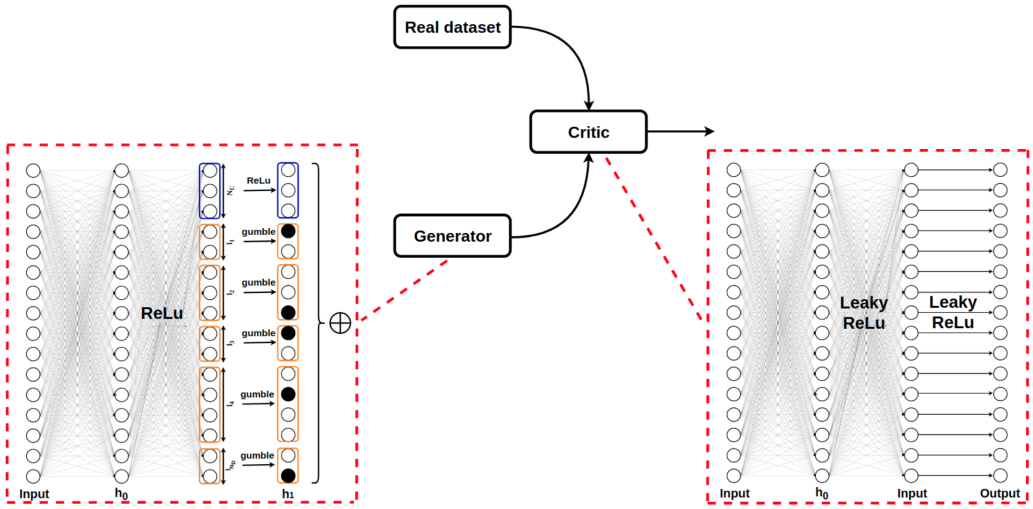



**Figure 19.** TabFairGAN architecture. Image taken from [54].

## 4. Methods for the Generation of Synthetic Samples

Synthetic data are artificially generated from real data and have the same statistical properties as real data. However, unlike real data, which are measured or collected in the real world, synthetic data are generated by computer algorithms [1,55].

According to [1], synthetic data can be generated from real data, from existing models, using expert domain knowledge, or from a mixture of these options. Synthetic samples generated from real data are obtained by creating a model that captures the properties (distribution, correlation between variables, etc.) of the real data. Once the model is created, it is used to sample synthetic data.

Synthetic data generated from existing models consist of instances generated from statistical models (mathematical models that have statistical assumptions about how the data are generated) or from simulations (e.g., game engines that create images from objects). The use of domain-specific knowledge can also be used to generate synthetic data. For example, knowledge about how the financial market behaves can be used to create an artificial dataset about stock prices. However, this requires extensive knowledge about the domain in question so that the synthetic data behaves similarly to real data.

A lot of artificial intelligence (AI) problems today arise from insufficient, poor quality, or unlabeled data. This is almost ubiquitous, as many fields of study suffer from such difficulties—e.g., physics [9,56], finance [57], health [10,58], sports [59], and agriculture [60]. As a result, there is a growing interest in the usefulness of synthetic data and the drawbacks it can overcome. An example of the usefulness of synthetic data can be found in [61], where a network trained only on synthetic data achieved competitive results when compared to a state-of-the-art network trained on real data.

In [62], the author argues that synthetic data are essential for the further development of Deep Learning and that many more potential use cases remain. He also discusses the three main directions for using synthetic data in machine learning: using synthetic data to train machine learning models and use them to make predictions in real-world data; using synthetic data to augment existing real datasets, typically used to cover underrepresented parts of the data distribution; and solving privacy or legal issues by generating anonymized data. The focus of this work is on the first two directions, with the goal of using synthetic data or augmented datasets to enhance the performance of machine learning models, so the generation of anonymized data is not addressed here.

In the following sections, several methods for generating synthetic samples are reviewed. To better organize them, they have been divided into deep learning methods and standard methods. Deep learning methods, as the name implies, use deep learning techniques to generate synthetic data. In contrast, standard methods are those that do not use deep learning.

### *4.1. Standard Methods*

In this section, we review some of the main methods for generating synthetic data (Our focus is on tabular data, so we refrain from writing about cropping, zooming, or inverting, which are used in image data). We have called them standard methods because they were the most commonly used methods before the success of generative deep learning models. The section is organized by the level of sophistication of the algorithm. Thus, random oversampling is shown as the first algorithm, followed by SMOTE and several algorithms that improve the core idea of SMOTE (e.g., by adding safe levels or clustering). Next, cluster-based oversampling is analyzed. Finally, Gaussian mixture models are reviewed as they provide a different approach to the task of generating synthetic data.

#### 4.1.1. Random Oversampling (ROS)

ROS adds additional observations to the dataset by randomly sampling from the minority class(es) with replacement. Probably, the simplest and most straightforward method for expanding a dataset is ROS. Nevertheless, this approach can change the data distribution. Thus, if a classifier is fed with such data, it may learn from an incorrect distribution. Moreover, since ROS duplicates observations, this technique does not create new synthetic samples but only replicates the existing ones. Therefore, more advanced techniques, such as SMOTE, had to be developed.

Examples of ROS can be found, for example, in [63], where the authors compared the use of random oversampling with random undersampling (Random undersampling is a technique that consists of randomly removing instances of the majority class so that minority classes are not underrepresented) (RUS). It has been shown that ROS gives better classification results than RUS since it does not affect the classification of the majority class instances as much as RUS, while it increases the classification of the minority class instances. Another example is shown in [64], where the authors also compare ROS and RUS. In that study, however, they concluded that ROS was surprisingly ineffective, producing little or no change in classification performance in most cases.

#### 4.1.2. Synthetic Minority Oversampling Technique (SMOTE)

SMOTE [2] is an oversampling approach in which synthetic observations are generated (and not duplicated, as in ROS) from the minority class(es). SMOTE was inspired by a perturbation method used to recognize handwritten digits. This was a very domain-specific problem, and so were the techniques used (e.g., rotation or skew), but the authors of SMOTE generalized them to generate synthetic samples in a less application-specific way.

The algorithm works as follows. Given a data point from a minority class and its nearest neighbor from the same class, the distance between them is determined (the distance is computed as the difference between both feature vectors, the data points). This distance is multiplied by a random number between 0 and 1 and added to the selected data point. This causes the new sample to fall in the line segment between the original sample and its neighbor. The same process is repeated until the desired number of samples is reached. Figure 20 shows a toy example of an iteration of the SMOTE algorithm.

The SMOTE algorithm is quite popular in the literature. In [65], for example, the authors evaluate the use of SMOTE for high-dimensional data. It was shown that SMOTE does not attenuate the bias toward the majority class for most classifiers. However, for k-nearest neighbor classifiers based on Euclidean Distance, SMOTE may be beneficial if the number of variables is reduced by variable selection. In [66], SMOTE is combined with decision trees and bagging to address the problem of imbalanced credit evaluation of companies. The proposed framework shows good results, outperforms the five other different approaches, and overcomes the class imbalance problem. Another example of using SMOTE can be seen in [67], where SMOTE is combined with Adaboost Support Vector Machine Ensemble with time weighting (ADASVM-TW) in two different ways and evaluated in a financial dataset. The first method uses SMOTE followed by ADASVM-

TW, while the second method embeds SMOTE into the iteration of ADASVM-TW. Both approaches greatly improved the recognition performance of the minority class.

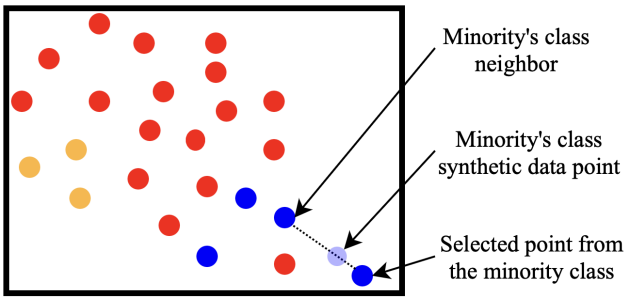



**Figure 20.** Toy example of the SMOTE algorithm for one iteration. The dataset has two minority classes (blue and orange) and one majority class (red). After selecting a minority class instance and its nearest neighbor, a synthetic data point is added somewhere in the line segment between them.

Although a more advanced technique than ROS, SMOTE still suffers from some problems—e.g., focusing on minority class instances (thus ignoring those of the majority class) or altering the true data distribution. That being said, some informed improvements can be applied. Therefore, Borderline-SMOTE, Safe-Level-SMOTE, and ADASYN have been introduced.

#### 4.1.3. Borderline-SMOTE

Han et al. [3] have proposed two algorithms that are a variation of SMOTE: Borderline-SMOTE1, which only oversamples the minority class(es) examples near the borderlines, and Borderline-SMOTE2, which also takes into account the majority class observations.

Borderline-SMOTE1 considers only the minority class data points that have a number of minority class neighbors in the range $[m/2, m]$, where $m$ is defined by the user. These are the points that can be easily misclassified (the borderline data points of the minority class). After detecting such observations, SMOTE is applied to create new synthetic samples. Borderline-SMOTE2 is quite similar, with the difference that it also considers the neighbors of the majority class. According to [3], Borderline-SMOTE offers improvements over SMOTE and ROS in terms of TP-rate and F-value.

Examples of Borderline-SMOTE can be found, for example, in [68], where the authors use this method for data augmentation and evaluate its impact on an EEG (Electroencephalography) classification dataset obtained with a brain-computer interface (BCI). Borderline-SMOTE did not improve the overall classification performance but significantly increased the performance of the classifiers that produced the worst results. Another example can be found in [69], where Borderline-SMOTE was improved by using Gabriel graphs. The authors addressed the main problems of Borderline-SMOTE and were able to improve its performance on neural networks.

#### 4.1.4. Safe-Level-SMOTE

SMOTE synthesizes minority class samples along a line connecting a minority class instance to its nearest neighbors, ignoring nearby majority class instances. Safe-Level-SMOTE [4], on the other hand, defines safe regions to prevent oversampling in overlapping or noisy regions, providing better accuracy performance than SMOTE and Borderline-SMOTE.

Each minority class example is assigned a safety level defined as the number of instances of the minority class in the $k$ nearest neighbors, where $k$ is specified by the user. Each synthetic instance is positioned closer to the largest safe level so that all synthetic instances are created only in safe regions. Intuitively, when given a data point, $p$, from the minority class and its nearest neighbor, $n$, (from that same class), the Safe-Level-SMOTE will generate a synthetic sample closer to $p$ if its safe level is higher than the one of $n$ or closer to $n$ otherwise. That is, the synthetic sample will be closer to the data point that

has more nearest neighbors from the minority class. Hence, the Safe-Level-SMOTE offers a wittier solution than the one of SMOTE, in the sense that it does not simply generate a random instance in the line segment between two minority class data points but takes into account their neighborhoods.

An example of using Safe-Level-SMOTE is shown in [70], where the authors overcome some of the difficulties of Safe-Level-SMOTE (some synthetic data points may be placed too close to nearby majority instances, which can confuse some classifiers and also the fact that it avoids using minority outcast samples for generating synthetic instances) by using two processes—moving the synthetic instances of the minority class away from the surrounding examples of the majority class and treating the outcasts of the minority class with a 1-nearest-neighbor model. Several machine learning models were evaluated with 9 UCI and 5 PROMISE datasets after using the above approach, and improvements in the F-measure were obtained.

#### 4.1.5. ADASYN

ADASYN [5] is an oversampling algorithm that improves the learning performance of the classifiers. It uses a weighted distribution for different minority class instances that takes into account their level of difficulty for a classifier to learn—the minority class samples that have fewer minority class neighbors are harder to learn than those which have more neighbors of the same class. Thus, more synthetic samples are generated for the minority class examples that are harder to learn and less for the minority class examples that are easier to learn.

ADASYN is similar to SMOTE in the sense that it generates synthetic samples in the line segments between two minority class data points. The difference is that ADASYN uses a density distribution as a criterion to automatically determine the number of synthetic samples to generate for each instance of the minority class. Hence, the extended dataset provides a balanced representation of the data distribution and forces the classifier to pay more attention to the more difficult-to-learn examples.

The ADASYN approach is used in [71] to process an unbalanced telecommunications fraud dataset. The authors concluded that ADASYN is more beneficial than SMOTE and that accuracy, recall, and F1-measure were improved when ADASYN was used. Another example can be found in [72], where ADASYN is used this time for data augmentation in an unbalanced churn dataset. A final example is retrieved from [73], where ADASYN is used in a financial dataset. The authors note that ADASYN overcame the problem of overfitting caused by SMOTE and improved the prediction of extreme financial risk.

While ADASYN, Safe-Level, and Borderline-SMOTE are variants of SMOTE, it is also possible to not modify the SMOTE algorithm but instead use an unsupervised algorithm before performing SMOTE (or random oversampling). Clustering algorithms are a type of unsupervised algorithm that can be very useful in detecting structures in the data (e.g., divide the data into classes). When applied well, clustering algorithms can reveal hidden patterns in the dataset that were previously undetectable.

#### 4.1.6. K-Means SMOTE

K-Means SMOTE was proposed by Last, Douzas, and Bacao in [6] and combines K-means [74], a popular clustering algorithm, with SMOTE, thereby avoiding the generation of noise and effectively overcoming the imbalances between and within classes.

K-Means SMOTE consists of three steps. First, observations are clustered using the K-means algorithm. This is followed by a filtering step in which the clusters with a small proportion of minority class instances are discarded. The number of synthetic samples to be created also depends on the cluster. That is, clusters with a lower proportion of minority class samples will have more synthesized instances. Finally, the SMOTE algorithm is applied to each of the clusters. Figure 21 shows the use of K-Means SMOTE in a toy dataset.

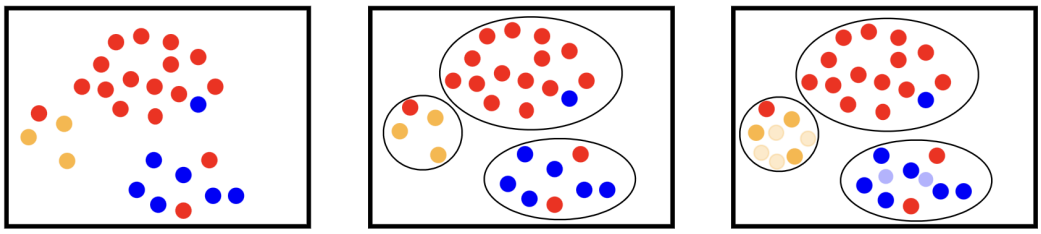


**Figure 21.** Toy example of the K-Means SMOTE algorithm. The left image shows the toy dataset, which consists of 3 classes: the blue and the orange are minority classes, and the red one is a majority class. The creation of clusters took place at the center. In the right picture, the clusters with a high proportion of samples from the minority class were populated with synthetic instances.

An example of the use of K-Means SMOTE can be found in [75], where the authors compared it with other methods of generating synthetic data, such as SMOTE or Borderline-SMOTE. It was shown that K-Means SMOTE is better at balancing datasets allowing machine learning models to perform better in terms of average recall, F1-score, and geometric mean.

#### 4.1.7. Cluster-Based Oversampling

Jo and Japkowicz, in [76], address the presence of small disjuncts in the training data. Their work has shown that the loss of performance in standard classifiers is not caused by class imbalance but that class imbalance can lead to small disjuncts, which, in turn, cause the loss of performance.

The Cluster-Based Oversampling algorithm consists of clustering the data for each class, i.e., each class is clustered separately (in [76], the authors used K-Means clustering, but theoretically, any clustering algorithm can be used), and then applying ROS to each cluster. For the majority class clusters, all clusters except the largest are randomly oversampled until they have the same number of observations as the majority class cluster with the most data points. The minority class clusters are randomly oversampled until each cluster has $m/N$ samples, where $m$ is the number of instances of the majority class (after ROS), and $N$ is the number of clusters of the minority class.

Cluster-Based Oversampling is similar to K-Means SMOTE in that both use clustering followed by oversampling, but they differ in some aspects. For instance, K-Means SMOTE uses a specific clustering algorithm, K-Means, and the classes are not clustered separately, while Cluster-Based Oversampling allows the user to freely choose the clustering algorithm, and the classes are clustered separately. Further, K-Means Clustering uses the oversampling SMOTE technique, while Cluster-Based Oversampling uses the ROS method.

The methods studied so far, with the exception of ADASYN, tend to neglect the distribution of the original data. Thus, a logical but different approach would be to model the underlying distribution of the data and draw a sample from it. However, estimating such a distribution is an extremely difficult problem, especially as the number of features in the data increase and simplifications need to be made.

#### 4.1.8. Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is a probabilistic model that assumes that the data can be modeled by a weighted sum of a finite number of Gaussian distributions [77]. Therefore, the resulting model is given by

$$p(x) = \pi_1 p_1(x) + \pi_2 p_2(x) + \ldots + \pi_n p_n(x)$$

where, in the univariate case, $p_i(x)$ is the probability density function of a univariate normal distribution with mean $\mu_i$ and standard deviation $\sigma_i$, $\pi_i$ is the weight assigned to each $p_i(x)$, and $n$ is the number of components. The number of components, $n$, is set by the user, and the parameters $\mu_1, \sigma_1, \mu_2, \sigma_2, \ldots, \mu_n, \sigma_n$, and $\pi_1, \pi_2, \ldots, \pi_{n-1}$ (The sum of all $\pi_i$ equals 1, so if $n-1$ weights are estimated, the last one is equal to 1 minus their sum. That

is, $\pi_j = \sum_{i \neq j}^{n} \pi_i$) are estimated, typically by an expectation-maximization algorithm—an iterative and well-founded statistical algorithm that calculates the probability that each point is generated by each component and then changes the parameters to maximize the likelihood of the data. For the multivariate case, $p_i(x)$ is replaced by a multivariate normal distribution, $N_k(\mu_i, \Sigma_i)$, where $k$ is the dimension of the multivariate normal distribution, $\mu_i$ is now a vector of means, and $\Sigma_i$ is the covariance matrix. Having determined the model, synthetic data are generated by drawing random samples from it.

An example of using GMM can be found in [78]. The authors address the problem of a lack of data in immersive virtual environments (IVEs) by using a Gaussian mixture model. The results have shown that the GMM is a good option to overcome the problem of a small sample size in IVE experiments.

### *4.2. Deep Learning Methods*

Deep learning methods are so named because they use deep learning techniques to create new instances. Unlike standard methods, deep learning models are more difficult to understand because they are more complex and usually cannot be interpreted. In this section, we review the three main classes of deep generative models: Bayesian networks (Even though BNs may not be considered Deep Learning, they are easy generalized to Bayesian Neural Networks, which are Deep Learning structures [79], so we have included them) (BNs), autoencoders(AEs), and GANs. There are innumerable variations of these algorithms and a whole range of domain-specific architectures. It would, therefore, not be possible to list everything in the literature, so instead, a comprehensive overview is presented.

#### 4.2.1. Bayesian Networks

A Bayesian network (also known as a belief network in the 1980s and 1990s) is a type of probabilistic graphical model that uses Bayesian inference for probability computations over a directed acyclic graph [80]. It is used to represent dependence between variables so that, essentially, any full joint probability distribution can be represented and in many cases, very succinctly [81]. In a Bayesian network, each node corresponds to a random variable (which may be discrete or continuous) and contains probability information that quantifies the effect of the parents (the nodes pointing to it) on the node. If there is a link from node $x_i$ to node $x_j$, then $x_i$ has a direct impact on $x_j$. Moreover, if there is a path from node $x_i$ to node $x_j$ (with at least one node in between), then $x_i$ also has an influence on $x_j$ (though not a direct influence).

As an example, suppose a certain person has an alarm system installed at home. The alarm is very good at detecting burglaries, but it also triggers for minor earthquakes. The person has asked his neighbors, John and Mary, to call him if the alarm goes off. On the one hand, however, John is more careful than Mary, so he almost always calls when he hears the alarm, but sometimes mistakes it for the phone ringing. On the other hand, Mary likes to listen to loud music, so she does not hear the alarm as often as John does. This is a simple toy example that can be modeled by a Bayesian network (see Figure 22).

The previous example is quite simple, but these structures can have many more layers, representing dependencies between multiple variables. As the number of layers increases, Bayesian networks become deep Bayesian networks. Although they played an important role in the history of deep learning (Bayesian networks were one of the first non-convolutional models to successfully allow training of deep architectures), they are rarely used nowadays [82].

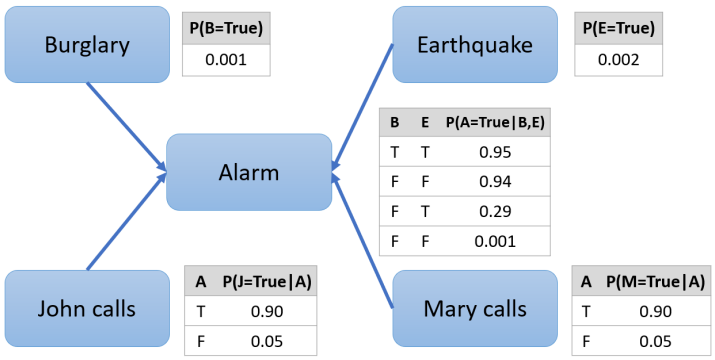



**Figure 22.** Toy example of a Bayesian network. A certain individual has an alarm installed at home, which fires in the case of minor earthquakes or burglaries (with a certain probability). The given individual also has two neighbors, Mary and John, who will call him in case they hear the alarm. Image adapted from [81].

An example of the use of Bayesian networks can be seen in [50], where the authors address the problem of sharing private data. A Bayesian network adds noise to the data to estimate an approximate distribution of the original data. The Bayesian network has been evaluated experimentally and found to significantly outperform existing solutions in terms of accuracy.

#### 4.2.2. Autoencoders

An autoencoder (AE) is a special type of feedforward neural network that consists of two parts: an encoder network that learns to compress high-dimensional data into a low-dimensional, latent spacial representation (the code), and a decoder network that decompresses the compressed representation into the original domain [83]. Figure 23 shows a diagram of an autoencoder.

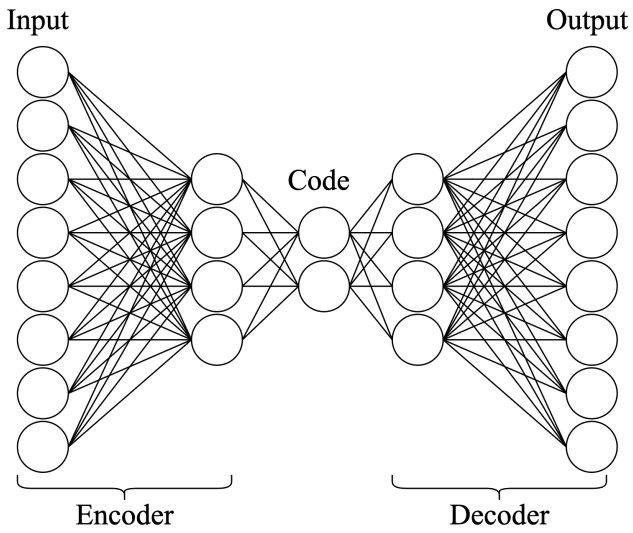



**Figure 23.** A diagram of an autoencoder. On the left, an example structure of an encoder is depicted. The input layer has more units than the middle layer, where the input is compressed into a lower-dimensional representation (the code). On the right, the decoder decompresses the code back to the original domain.

Formally, the encoder can be viewed as a function, $c = E(x)$, that produces a low-dimensional representation of the data, and the decoder as a function, $r = D(c)$, that produces a reconstruction of the code. The goal is not for the autoencoder to learn how to set $D(E(x)) = x$ for each input example $x$ but rather to learn how to copy the original

data only approximately, and only inputs that resemble the original data. By constraining it and forcing it to learn which aspects of the data should be prioritized, autoencoders can learn useful properties about the data (autoencoders have been on the deep learning landscape for decades and have typically been used for feature learning and dimensionality reduction) [84].

In terms of generating synthetic samples, autoencoders have some issues. First, the learned distribution of points in the latent space is undefined, i.e., when a point is sampled from the latent space and decoded to generate a new example, there is no guarantee that it is a plausible sample. Second, there is a lack of diversity in the generated samples. Finally, points belonging to the same class may have large gaps in the latent space, which can lead to poorly generated instances when samples are drawn from their neighborhood. To overcome these problems, variational autoencoders (VAEs) can be used.

Variational autoencoders were first proposed by Diederik P Kingma and Max Welling in [7] and are a natural extension of autoencoders aimed at solving the aforementioned problems. VAEs improve on vanilla autoencoders by making a few changes to the encoder and loss function:

**Encoder**. The encoder of a VAE maps each point in the original data to a multivariate normal distribution in the latent space, represented by the mean and variance vectors. VAEs assume that there is no correlation between two dimensions in the latent space, so the covariance matrix does not need to be calculated because it is diagonal. This small change ensures that a point, $a$, sampled from the neighborhood of another point, $b$, in the latent space, is similar to $b$. Thus, a point in the latent space that is completely new to the decoder will most likely still yield a correct sample.

**Loss function**. The VAE loss function adds the Kullback–Leibler (KL) divergence to the autoencoder reconstruction function (typically, the binary cross entropy or the root mean squared error). Formally, the KL divergence in this particular case can be written as follows.

$$KL\Big(N(\mu,\sigma) \parallel N(0,1)\Big) = \frac{1}{2}\sum_{i=1}^{k}(1 + log(\sigma_i^2) - \mu_i^2 - \sigma_i^2)$$

where $k$ is the number of dimensions in the latent space. Therefore, the loss function becomes

$$L(x,\hat{x}) = RL(x,\hat{x}) + \frac{1}{2}\sum_{i=1}^{k}(1 + log(\sigma_i^2) - \mu_i^2 - \sigma_i^2)$$

where $RL$ is the reconstruction loss, $x$ denotes the input data, and $\hat{x}$ is the predicted output. This loss function provides a well-defined distribution (the standard normal distribution) that can be used to sample points in the latent space—sampling from this distribution most likely guarantees that the sample points are in the region from which the decoder is to decompress. Further, the gaps between points in the latent space will be smaller.

Therefore, changing the encoder mapping and adding the KL divergence to the loss function leads to a better framework for generating synthetic samples—the variational autoencoder.

The use of autoencoders to generate synthetic data is widespread in the literature. For example, in [85], the authors used a multichannel autoencoder (MCAE) to assist classifiers in the learning process. They concluded that the use of MCAE provided better feature representation. In addition, the experimental results validated their methodology for generating synthetic data. In [86], a variational autoencoder was used to address the problem of imbalanced image learning. It was shown that the VAE can generate novel samples and that it produces better results compared to other methods in several distinct datasets with different evaluation metrics. A final example of the use of autoencoders can be seen in [87], where a VAE was used to generate accident data, which was then used for data augmentation. The VAE was compared to SMOTE and ADASYN. This showed its superiority as it provided a better learning process for the classifiers and thus provided better classification metrics.

4.2.3. Generative Adversarial Networks

As shown in Section 3, GANs are a type of generative deep learning consisting of two networks: the generator, G, and the discriminator, D. The details of how they operate have already been reviewed, so we will now focus on the practical applications of such structures. Due to the usefulness of GANs in generating synthetic samples, they are widely used. Hence, it would be tedious to list them all. Therefore, only some interesting results will be shown.

In [88], the authors used a GAN to generate artificial EEG (Electroencephalography) datasets. The results presented were quite good: indeed, GANs (in this case, with convolutional layers) were able to generate brain signals similar to the real ones (obtained by EEG in multiple subjects).

Patel et al. used a CGAN for data augmentation in a signal modulation dataset used for automatic modulation classification [89]. These data were then used to improve the accuracy of a CNN classifier used as a benchmark. It was concluded that CGAN-enriched data could greatly benefit CNN-based training—it has faster convergence and lower training loss. Moreover, the more data generated by the CGAN, the better the F1-score of the CNN classifier is (the authors used 1000, 2000, 3000, 4000, and 5000 synthesized samples). Figure 24 shows the F1-score for the original and the extended dataset at different signal-to-noise ratios (SNR). Clearly, the F1-score increases at each SNR level as more synthetic samples are added to the original dataset.

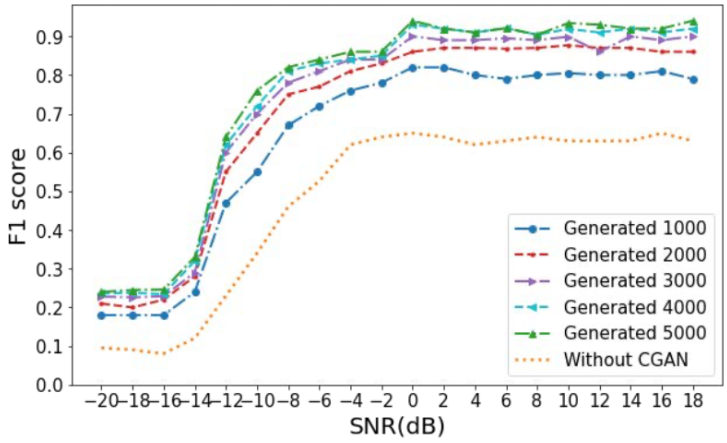



**Figure 24.** F1-score on the original data and on the augmented datasets (1000, 2000, 3000, 4000, and 5000 synthetic samples were added to the original data) at different SNR levels. The plot shows that, as the number of generated samples increases, the better the F1-score at each SNR level. Image taken from [89].

Another example of the use of GANs is the Multiple Fake Class GAN (MFC-GAN) ([90]). The MFC-GAN was used to handle datasets with multiple imbalanced classes by augmenting the original data with artificial samples. Four public datasets were used, MNIST, E-MNIST, CIFAR-10, and SVHN, and MFC-GAN was compared with FSC-GAN [91], AC-GAN [33], and SMOTE [2], both in terms of the quality of the generated samples and in a classification task (a baseline CNN classifier was used). It was found that MFC-GAN provided better quality generated samples and that the training time was significantly reduced compared to FSC-GAN (MNIST dataset). The results also showed that MFC-GAN performed better than SMOTE and AC-GAN on all SVHN and CIFAR-10 minority classes and in 7 of 10 E-MNIST and MNIST minority classes.

In [92], Sushko et al. proposed the One-Shot GAN, which given just one image (or video) as input, can generate images (or videos) that are significantly different from the original one. This type of GAN has improved the quality and variety of images (and videos) over previous works when only one image (or video) is available. When only small amounts of data are available, the One-Shot GAN mitigates the memorization problem (reproducing

the original image) and is able to generate images that are structurally different from the original. This is extremely useful for data augmentation tasks in domains where data is very scarce and collecting it may be challenging.

A quantum GAN—*entangling* quantum GAN, EQ-GAN—was proposed in [93]. By leveraging quantum circuits' entangling (quantum entanglement is a physical phenomenon that happens when, in a set of particles, an individual particle's quantum state cannot be described independently of the state of the others, no matter how far apart they are) power, it overcomes some limitations of previously proposed quantum GANs (non-convergence due to mode collapse and a non-unique Nash equilibrium). Moreover, the authors have shown that the EQ-GAN can generate an approximate quantum random access memory (QRAM), which is required by most machine learning algorithms. They have further demonstrated an application of such a QRAM, improving the performance of a quantum neural network in a classification task.

Finally, to conclude this subsection, we show one last example. In [94], the authors have proposed the Metropolitan GAN (MetroGAN), which is used for urban morphology simulations. Recent studies have shown that GANs have the potential to simulate urban morphology, despite being a challenging task. Nevertheless, the existing GAN models are limited by the instability in model training and the sparsity of urban data, compromising their application. However, when compared to other state-of-the-art urban simulation methods—XGBoost, U-NET, and CityGAN—the MetroGAN outperforms them all in the three levels used to evaluate the results: pixel level, multi-scale spatial level, and perceptual level.

### *4.3. Thoughts on the Algorithms*

In this section, eight techniques for data augmentation were reviewed. ROS is the most simple of them all and, therefore, is easier to implement than any of the others. However, it is a very naive approach that does not take into account the distribution of the data. Further, it disregards the majority class instances, as well as the difficulty of the classifiers in learning the decision boundaries. A simple yet more intelligent way to improve ROS is SMOTE. This technique does not replicate observations as ROS does but adds new synthetic data points to the dataset. This can make it easier for a classifier to learn from the data. Nonetheless, SMOTE does not care about changing the distribution of the data and does not consider majority class observations.

SMOTE brought a highly successful synthetic data generation method but also a lot of room for improvement. Therefore, new algorithms were created by borrowing the core idea of SMOTE, which is to add noise to the instances. Borderline-SMOTE oversamples near the borderlines to make the learning task easier for classifiers while also taking into account the majority class observations. Safe-Level-SMOTE has defined safe regions to generate better quality instances, which is an improvement over SMOTE and Borderline-SMOTE.

K-Means SMOTE first clusters the data using the K-Means algorithm and then oversamples the clusters using SMOTE, effectively overcoming the imbalances between and within classes. ADASYN is another variant of SMOTE. This method takes into account the learning difficulties of the classifiers and aims to not change the data distribution (one of the drawbacks of SMOTE). Cluster-Based Oversampling takes into account the presence of small disjuncts in the data. This algorithm is not a variant of SMOTE but a variant of ROS. Both the minority and majority classes are oversampled so that each class has the same number of instances.

Gaussian mixture models use a different approach to address the synthetic data generation task—modeling the data with a weighted sum of normal distributions. While this is usually an improvement over previous algorithms, it has two major drawbacks. First, not all datasets can be modeled with a weighted sum of the Gaussian distribution. Therefore, the use of GMM may not be the most appropriate method for generating plausible samples. On the other hand, some types of data may have categorical features. In

these cases, GMM cannot be applied because the normal distribution is continuous, and it cannot model discrete variables.

BNs, AEs, and GANs are more complex techniques compared to the others. Unlike the previous methods, they use a Deep Learning approach that allows them to better learn the underlying patterns in the data and, therefore, offer higher quality synthetic patterns in most cases. Bayesian networks were widely used in the past but have fallen out of favor and are rarely used today. Autoencoders, especially variational autoencoders, are powerful generative models that have evolved and are proving useful in data generation tasks.

Nevertheless, autoencoders are not as popular and usually not as powerful as GANs. Yann LeCun has even described them as "the most interesting idea in the last 10 years in machine learning" [95]. GANs have countless different architectures, and many are yet to be created. Only a few applications of GANs for the generation of samples were shown, as it would be grueling (and probably impossible) to find and summarize all the literature on GANs and data generation. They can be quite problem-specific, so a few have been selected to show their capabilities and broad application to real-world data.

## 5. Synthetic Sample Quality Evaluation

Evaluating the quality of the generated samples is critical to assessing the quality of the method used to generate synthetic data. There is a huge number of evaluation techniques, so it is tedious and almost impossible to explore and describe them all. Moreover, many of these evaluation techniques are intended for specific types of data or for very specific domains. Therefore, this section focuses on evaluation methods for tabular data.

The simplest way to evaluate the quality of synthetic data is to compare their basic statistics (e.g., mean, median, standard deviation) with those of the real data. If the values are similar, it is likely that the synthetic data are similar to the real data. However, this can be misleading, as statistician Francis Anscombe showed in 1973 [96]. The Anscombe quartet includes four datasets that are nearly identical in terms of basic descriptive statistics but whose distributions are very different.

Anscombe constructed his quartet to demonstrate the importance of plotting the data when analyzing it. Back in 1973, it may have been difficult to create graphs with data, in part because of scarce and expensive computing resources. Today, however, it is quite easy, with hundreds of graphics libraries available for various programming languages. Thus, another method to evaluate the quality of synthetic data is to use graphical representations (e.g., box plots, histograms, violin plots).

Comparing the graphs of the synthetic data with the graphs of the real data provides a visual assessment of the generated data, which can also be supplemented by descriptive statistics. The Q-Q plot is a probability plot that can be particularly useful for making comparisons between two data distributions, as it plots their quantiles against each other and can, thus, evaluate the similarity between the distributions. Given the vast amounts of data available today, with datasets containing hundreds or even thousands of variables, it can be prohibitively expensive to visually represent and analyze all of the data, so other approaches to evaluate synthetic data are required.

Machine learning efficacy is another technique for evaluating synthetic data. Since many of the uses of synthetic data are to increase the performance of ML models, machine learning efficacy is used to evaluate the quality of synthetic data with respect to the performance of ML models. It consists of, given a dataset, $D$, already split into a trainset, $D_{train}$, and testset, $D_{test}$, comparing the performance of ML models (e.g., logistic regression, decision trees, artificial neural networks) when trained in $D_{train}$, and on $D_{synth}$ (the synthetic data), and evaluated in $D_{test}$ (see Figure 25). If the performance (e.g., in terms of accuracy, recall, precision, F1-score) of the models trained using $D_{train}$ is similar to those trained using $D_{synth}$, then the synthetic data is likely to follow the underlying data distribution. In [47,48], this method was used to evaluate the performance of the TGAN and CTGAN architectures, respectively. Further, in [97], this technique was used to evaluate the forecast of emerging technologies.

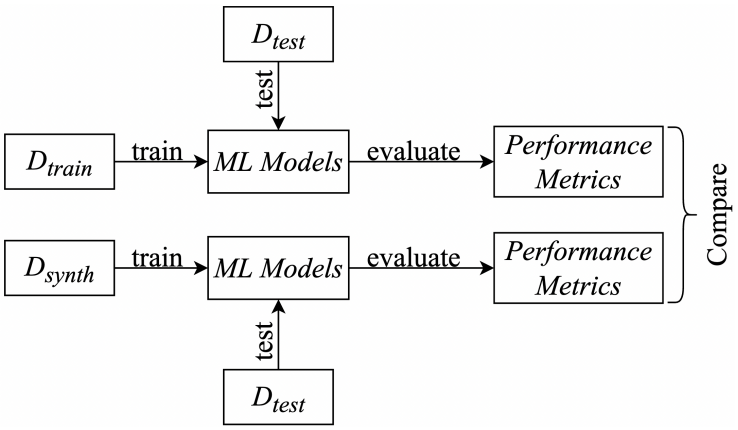



**Figure 25.** Machine learning efficacy diagram. The performance (accuracy, F1-score, etc.) of ML models (random forests, decision trees, etc.) in the test set, $D_{test}$, is compared when the models are trained on the real training data, $D_{train}$, and when they are trained using the synthetic data, $D_{synth}$.

In [98], Shmelkov et al. argue that the existing methods for evaluating synthetic samples are insufficient and need to be adapted to the task at hand. They begin by addressing two commonly used metrics, namely the Inception Score [40] and the Fréchet Inception Distance [41]. Both metrics are used for the evaluation of image-generated data and, thus, are not the focus of this work. Nonetheless, it is important to at least mention them, as they are widely used in the literature to evaluate synthetic image data.

After presenting these two metrics, the authors introduced their proposed metrics—GAN-train and GAN-test—which, although applied to image data, can also be applied to other types of data, such as tabular datasets. Moreover, despite both measures having "GAN" in their name, the synthetic samples do not need to be generated exclusively with a GAN but can also be generated with any other method. Therefore, we have slightly modified the definition of GAN-train and GAN-test given in [98] to make it more general by replacing the use of a GAN with any synthetic data generation method and the image data with any type of data (see Figure 26).

**GAN-train**. A classification network is trained with instances generated by a synthetic data generation method, and its performance is evaluated against a test set consisting of real-world data. This measure provides a measure of how far apart the generated and true distributions are.

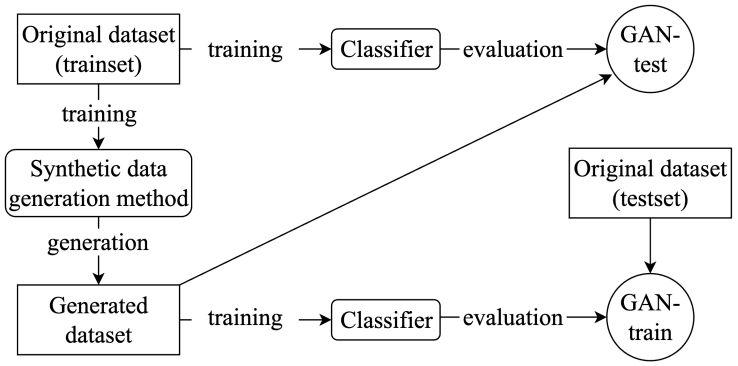



**Figure 26.** A diagram representing the GAN-train and GAN-test metrics. The GAN-train is a measure consisting of the accuracy of a classifier trained in the generated data and evaluated in the real data. GAN-test learns on the real data and is evaluated in the generated data. Image based on [98].

**GAN-test**. A classification network is trained on a real dataset and evaluated on the generated data. GAN-test provides a measure to evaluate whether the synthetic data generation method has overfitted (values significantly higher than the ones from validation

accuracy) or underfitted (values significantly lower than the ones from validation accuracy) the data.

Another technique to evaluate the quality of synthetic data was proposed in [99]. The authors address the fact that most existing evaluation metrics for generative models are focused on image data and have introduced a domain and model-independent metric. The metric is three-dimensional ($\alpha$-Precision, $\beta$-Recall, Authenticity), and it evaluates the fidelity, diversity, and generalization of each generative model and is independent of the domain (e.g., images or tabular data). Moreover, the three components correspond to interpretable probabilistic quantities, making it easier to detect a lack of synthetic data quality if such a problem occurs.

**Fidelity.** The $\alpha$-Precision component measures the similarity between the generated and the real samples. Thus, values with high-fidelity correspond to realistic samples, i.e., samples that resemble those from the original dataset.

**Diversity.** It is not enough to have samples that resemble those from the original dataset. High-quality synthetic data must also have some diversity. The $\beta$-Recall component evaluates how diverse the generated samples are. That is, whether the generated data is diverse enough to cover the existing variability in the real data.

**Generalization.** Last but not least, it is essential that the generated samples are not copies of the original data. In fact, high fidelity and diversity values do not guarantee that the synthetic samples are not just copies of the original dataset. Therefore, the authenticity component is a measure of how well the model can generalize and, therefore, not overfit the real data.

The first two components are computed by embedding the real and synthetic data in hyperspheres. That is, the original data, $X_r$, and the generated data, $X_s$, are mapped from the original domain, $\mathcal{X}$, to a hypersphere of radius $r$, $\mathcal{S}_r$. The third component is computed by evaluating the proximity of the real data to the generated data in the embedding space using a hypothesis test. Figure 27 shows a representation of the three metrics.

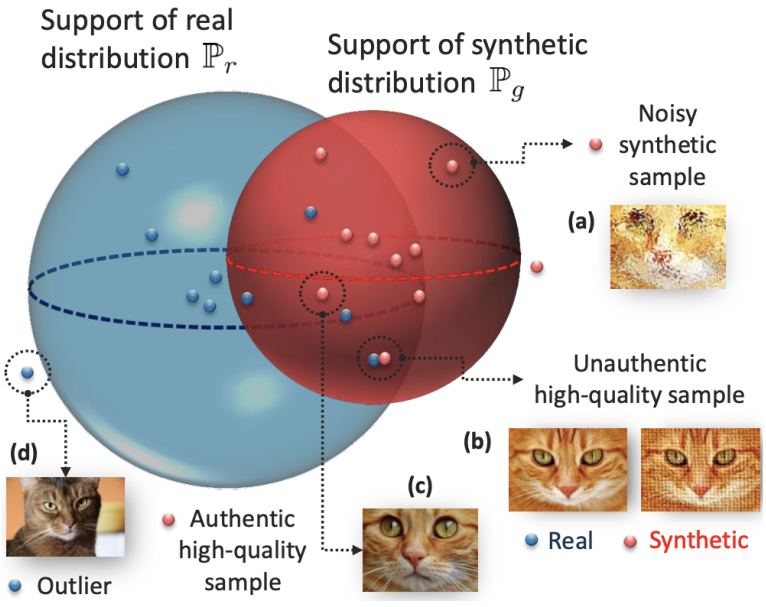



**Figure 27.** Representation of $\alpha$-Precision, $\beta$-Recall, and Authenticity. The blue and red spheres correspond to the $\alpha$- and $\beta$-supports of the real and the generated samples, respectively. Intuitively, these regions are "safe-zones" where points that lie outside the spheres are outliers, and points inside the spheres are "ordinary" samples. (**a**) Generated samples that lie outside the blue sphere are unrealistic. (**b**) Synthetic samples that are very close to real instances are inauthentic because they are almost exact copies of the real data. (**c**) Synthetic data points inside the blue sphere and without real data points near them are considered high-quality samples. (**d**) A data point outside the sphere is considered an outlier. Image retrieved from [99].

Finally, an important aspect of this metric is that, unlike the other metrics, it provides the ability to evaluate each instance. Considering this, the authors of [99] have also proposed a model-checking framework where low-quality samples (low values in some or all components) are discarded. Therefore, the final generated dataset is a "curated" version consisting only of high-quality samples.

In this section, six evaluation techniques were examined—descriptive statistics, graphical representations, machine learning efficacy, GAN-train, GAN-test, and the ($\alpha$-Precision, $\beta$-Recall, Authenticity) metric. It is always good not to use only one measure and to combine at least two of them. For example, as mentioned earlier, the descriptive statistics of a generated sample may be similar to those of the real data, but the distribution of data points may be very different. Or the efficacy of machine learning might provide similar values for the models trained in the real data and those learned in the generated data, but their descriptive statistics, or graphical representations, may be very different.

Evaluating synthetic data is challenging and depends heavily on the problem at hand. Sometimes generating synthetic data can be useful to better train a classifier when there is a lack of data. In other cases, the problem might be to create simulated realities for a video game. The definition of "high-quality samples" is likely to be different in the two cases. In the first scenario, the synthetic data must be very similar to the original data for the classifier to learn a reasonable model of the real world. Therefore, the synthetic data must be closely scrutinized, and various evaluation techniques need to be used. In the latter case, the generated data need not be plausible in the human world, and less stringent criteria can be used to evaluate the quality of the samples.

Even for problems of a similar nature, the evaluation techniques may be different. Suppose there are two different classification tasks. The first is to classify a patient with cancer as "benign" or "malignant". The second task is to classify the sex of an unborn child as "male" or "female". In the first task, it is critical to generate extremely high-quality synthetic data to improve the classifiers. The data must be highly plausible and truly represent the real world. Failing to generate trustworthy synthetic data might lead doctors not to diagnose a patient with a malignant cancer, which can have serious consequences for the patient (and also for the doctor). Therefore, multiple evaluation techniques must be used to be sure that the generated data will help in the classification task and not jeopardize it.

In the second scenario, evaluation techniques to assess the quality of the generated sample may not need to be as rigorous. Improving the performance of the classifier might be useful even if it is with samples of intermediate quality so it is not necessary to analyze the synthetic data in detail. Whether the unborn child is classified as "female" or "male" does not have as much impact as a tumor being "benign" or "malignant".

## 6. Discussion

Given the amount of information covered in this document, it is important to schematize everything in order to have a clear overview of what was shown. For this purpose, an organizational chart (Figure 28) was created. It has been divided into two main topics, namely the methods used to generate synthetic data and the evaluation of the synthetic samples.

As for the synthetic data generation methods, they were further divided into standard and deep learning methods, as in Section 4. The standard methods include ROS, SMOTE, Borderline-SMOTE, Safe-Level-SMOTE, ADASYN, K-Means SMOTE, Cluster-Based Oversampling, and GMM. Deep learning methods consist of Bayesian networks, autoencoders, and generative adversarial networks. GANs have been further explored and are, therefore, divided into three main areas: GANs for music generation, image generation, and tabular data. As seen in Section 3.3, most GAN architectures focus on image generation, so they are better represented in the organizational chart than the other two. In addition, the CGAN and WGAN can (and should) be used in the context of tabular data, as they have useful properties. In fact, the CTGAN uses a conditional generator, which is based on the CGAN properties, and the TabFairGAN uses the WGAN with gradient penalty. Therefore, an

overlapping region has been added to the organization chart to illustrate that CGAN and WGAN can be used in tabular domains.

Finally, the last part of the organizational chart consists of the evaluation techniques discussed in Section 5—descriptive statistics, graphical representations, machine-learning efficacy, GAN-train, GAN-test, and ($\alpha$-Precision, $\beta$-Recall, Authenticity). They are fundamental to evaluate the quality of the generated samples and, thus, the quality of the generative models.

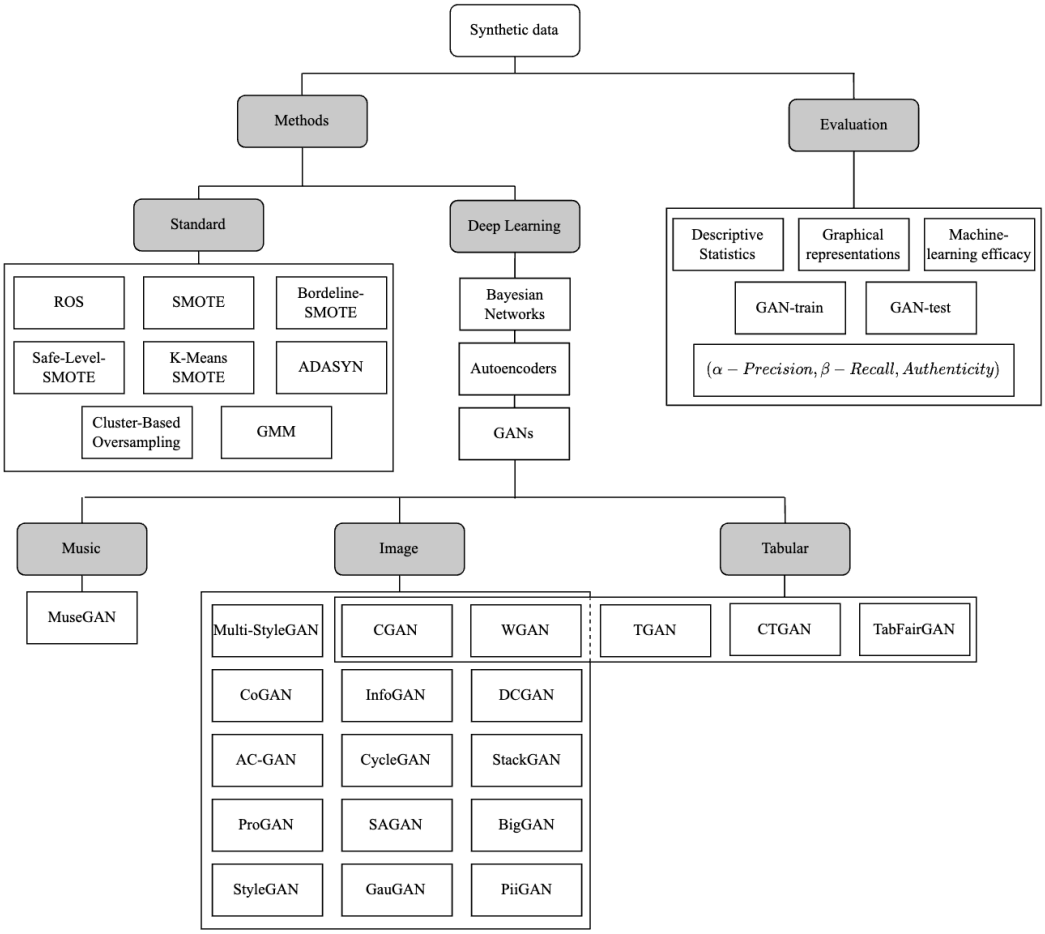



**Figure 28.** Organizational chart depicting several methods for the generation of synthetic samples and the evaluation techniques covered in this work.

Despite the concise representation offered by the organizational chart, it lacks a temporal dimension that can be interesting to visualize the evolution over time of the ideas that have been described. With this in mind, a timeline (Figure 29) was created. It is divided into two main themes, GAN architectures and methods for generating synthetic samples (the two themes sometimes overlap since GANs are generative models). While the topics are on the vertical axis, the horizontal axis is reserved for the temporal dimension. Given the large time span, the time axis is divided into years. We note that it is not clear when a paper on ROS, GMMs, Bayesian networks, or AEs was first published, so they are not included in the timeline.

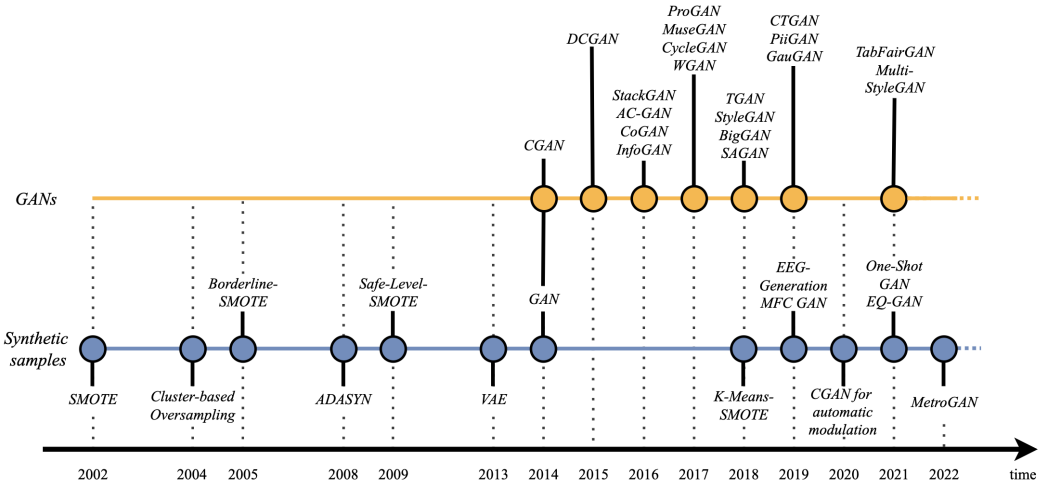



**Figure 29.** Timeline showing the different GAN architectures (in orange) and the generation of the synthetic sample methods (in blue) covered in this work.

Finally, a summary of the different synthetic data generation methods is given in Table 4. The methods are categorized by their type—standard or deep learning methods—and the references used throughout the document are shown in the References column.

**Table 4.** Summary of the synthetic data generation methods covered in this work.

| Methods | Method Type | References |
|---|---|---|
| Random Oversampling (ROS) | Standard | [63,64] |
| SMOTE | Standard | [2,65–67] |
| Borderline-SMOTE | Standard | [3,68,69] |
| Safe-Level-SMOTE | Standard | [4,70] |
| K-Means SMOTE | Standard | [6,75] |
| ADASYN | Standard | [5,71–73] |
| Cluster-Based Oversampling | Standard | [76] |
| GMM | Standard | [77,78] |
| Bayesian Networks | Deep Learning | [50,80–82] |
| Autoencoders | Deep Learning | [7,83–87] |
| GANs | Deep Learning | [8,13,20–39,42–48,54,88–90,92–94] |

## 7. Conclusions

To the best of our knowledge, this survey provides a comprehensive overview of the main synthetic data generation methods, the key breakthroughs in generative adversarial networks—with a special focus on GANs for tabular data—and how to assess the quality of synthetic samples. Unlike other existing surveys, e.g., [16], which focus on synthetic data, or [17], which surveys the recent advances in GANs, ours brings together both subjects.

We have provided a thorough explanation of what a GAN is, shown some of the main issues during training, the key breakthroughs in GAN architecture, and how GANs can deal with tabular data. We have also presented the more classical methods (we termed them standard methods) for the generation of synthetic samples and examples of published works showing their applicability. The same goes for deep learning methods, in which we showcased some research papers where they were used and how they work.

One of the core issues that we have come across in this work is that most research about GANs has been focused on image generation. As such, GANs for image generation tasks have been maturing and becoming more robust in the last few years. The same does not hold true for tabular data, a domain where GANs still have room for improvement.

Lastly, we showed how to evaluate the quality of synthetic data. It is crucial that the synthetic samples are of high quality, but defining them is quite problem-specific and depends on the domain.

In summary, there are four important points we have gained from this investigation. First, we have seen that Lecture Notes in Computer Science is the journal with the most publications in the area and that the authors Wang Y. and Nikolenko S. I. are of particular relevance—this may be useful for new researchers in the field to know where to find information. Second, we showed that deep learning methods were more complex than standard ones but tended to produce better results. Third, image generation has been the focus of research in GAN, while tabular data generation needs further research. Finally, evaluation techniques for synthetic data can be subjective, as they depend on the task and domain in question.

Given that three major topics were surveyed in this work—synthetic data generation methods, GANs, and synthetic data evaluation techniques, it has only been possible to cover the most important algorithms, architectures, and techniques. The literature is so vast that it would be impossible to summarize everything in a single paper. Another drawback of our study concerns the limitation of the query used (see Listing 1), since the datasets returned by it are necessarily incomplete and limited. Based on the numerous works in the literature, it is far from easy to create an all-inclusive query. This is something that could be revised and improved in future work. Nonetheless, we believe this review to be a good starting point for a new researcher in the field.

As mentioned before, one of the things that can be improved in future work is the quality of our query. Further, we want to compare the quality of the data produced by the tabular GANs outlined in Section 3.4—TGAN, CTGAN, and TabFairGAN. In order to do so, we will rely on datasets already well-established in the scientific community. Moreover, we intend to focus especially on unbalanced datasets and use machine-learning models to assess their performance in the minority class when synthetic samples are incorporated into their training.

**Author Contributions:** Conceptualization, A.F. and B.V.; methodology, A.F. and B.V.; software, B.V.; validation, A.F. and B.V.; formal analysis, B.V.; investigation, A.F. and B.V.; resources, B.V.; data curation, A.F. and B.V.; writing—original draft preparation, B.V.; writing—review and editing, A.F.; visualization, A.F. and B.V.; supervision, A.F.; project administration, A.F.; funding acquisition, A.F. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Emam, K.; Mosquera, L.; Hoptroff, R. Chapter 1: Introducing Synthetic Data Generation. In *Practical Synthetic Data Generation: Balancing Privacy and the Broad Availability of Data*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2020; pp. 1–22.
2. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
3. Han, H.; Wang, W.Y.; Mao, B.H. Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In Proceedings of the International Conference on Intelligent Computing, Hefei, China, 23–26 August 2005; pp. 878–887.
4. Bunkhumpornpat, C.; Sinapiromsaran, K.; Lursinsap, C. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Bangkok, Thailand, 27–30 April 2009; pp. 475–482.
5. He, H.; Bai, Y.; Garcia, E.A.; Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–8 June 2008; pp. 1322–1328.

6. Douzas, G.; Bacao, F.; Last, F. Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE. *Inf. Sci.* **2018**, *465*, 1–20. [CrossRef]
7. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
8. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems 27 (NIPS 2014), Montreal, QC, Canada, 8–13 December 2014; Volume 27.
9. Siddani, B.; Balachandar, S.; Moore, W.C.; Yang, Y.; Fang, R. Machine learning for physics-informed generation of dispersed multiphase flow using generative adversarial networks. *Theor. Comput. Fluid Dyn.* **2021**, *35*, 807–830. [CrossRef]
10. Coutinho-Almeida, J.; Rodrigues, P.P.; Cruz-Correia, R.J. GANs for Tabular Healthcare Data Generation: A Review on Utility and Privacy. In *Discovery Science*; Soares, C., Torgo, L., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 282–291.
11. Gupta, A.; Vedaldi, A.; Zisserman, A. Synthetic data for text localisation in natural images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 2315–2324.
12. Bolón-Canedo, V.; Sánchez-Maroño, N.; Alonso-Betanzos, A. A review of feature selection methods on synthetic data. *Knowl. Inf. Syst.* **2013**, *34*, 483–519. [CrossRef]
13. Frid-Adar, M.; Klang, E.; Amitai, M.; Goldberger, J.; Greenspan, H. Synthetic data augmentation using GAN for improved liver lesion classification. In Proceedings of the 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), Washington, DC, USA, 4–7 April 2018; pp. 289–293.
14. Koch, B. Status and future of laser scanning, synthetic aperture radar and hyperspectral remote sensing data for forest biomass assessment. *ISPRS J. Photogramm. Remote Sens.* **2010**, *65*, 581–590. [CrossRef]
15. Wu, X.; Liang, L.; Shi, Y.; Fomel, S. FaultSeg3D: Using synthetic data sets to train an end-to-end convolutional neural network for 3D seismic fault segmentation. *Geophysics* **2019**, *84*, IM35–IM45. [CrossRef]
16. Nikolenko, S.I. Synthetic Data Outside Computer Vision. In *Synthetic Data for Deep Learning*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 217–226.
17. Pan, Z.; Yu, W.; Yi, X.; Khan, A.; Yuan, F.; Zheng, Y. Recent progress on generative adversarial networks (GANs): A survey. *IEEE Access* **2019**, *7*, 36322–36333. [CrossRef]
18. Di Mattia, F.; Galeone, P.; De Simoni, M.; Ghelfi, E. A survey on gans for anomaly detection. *arXiv* **2019**, arXiv:1906.11632.
19. Saxena, D.; Cao, J. Generative adversarial networks (GANs) challenges, solutions, and future directions. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–42. [CrossRef]
20. Wang, Q.; Gao, J.; Lin, W.; Yuan, Y. Learning from synthetic data for crowd counting in the wild. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–17 June 2019; pp. 8198–8207.
21. Atapour-Abarghouei, A.; Breckon, T.P. Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2800–2810.
22. Liu, J.; Qu, F.; Hong, X.; Zhang, H. A small-sample wind turbine fault detection method with synthetic fault data using generative adversarial nets. *IEEE Trans. Ind. Inform.* **2018**, *15*, 3877–3888. [CrossRef]
23. Zhang, L.; Gonzalez-Garcia, A.; Van De Weijer, J.; Danelljan, M.; Khan, F.S. Synthetic data generation for end-to-end thermal infrared tracking. *IEEE Trans. Image Process.* **2018**, *28*, 1837–1850. [CrossRef] [PubMed]
24. Wang, Q.; Gao, J.; Lin, W.; Yuan, Y. Pixel-wise crowd understanding via synthetic data. *Int. J. Comput. Vis.* **2021**, *129*, 225–245. [CrossRef]
25. Chen, Y.; Li, W.; Chen, X.; Gool, L.V. Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 1841–1850.
26. Dunn, K.W.; Fu, C.; Ho, D.J.; Lee, S.; Han, S.; Salama, P.; Delp, E.J. DeepSynth: Three-dimensional nuclear segmentation of biological images using neural networks trained with synthetic data. *Sci. Rep.* **2019**, *9*, 18295. [CrossRef] [PubMed]
27. Kim, K.; Myung, H. Autoencoder-combined generative adversarial networks for synthetic image data generation and detection of jellyfish swarm. *IEEE Access* **2018**, *6*, 54207–54214. [CrossRef]
28. Torkzadehmahani, R.; Kairouz, P.; Paten, B. Dp-cgan: Differentially private synthetic data and label generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Long Beach, CA, USA, 16–17 June 2019.
29. Mirza, M.; Osindero, S. Conditional generative adversarial nets. *arXiv* **2014**, arXiv:1411.1784.
30. Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv* **2015**, arXiv:1511.06434.
31. Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 2180–2188.
32. Liu, M.Y.; Tuzel, O. Coupled generative adversarial networks. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 469–477.
33. Odena, A.; Olah, C.; Shlens, J. Conditional image synthesis with auxiliary classifier gans. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 2642–2651.

34. Zhang, H.; Xu, T.; Li, H.; Zhang, S.; Wang, X.; Huang, X.; Metaxas, D.N. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5907–5915.
35. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein generative adversarial networks. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 214–223.
36. Zhu, J.Y.; Park, T.; Isola, P.; Efros, A.A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2223–2232.
37. 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. *arXiv* **2017**, arXiv:1709.06298.
38. Karras, T.; Aila, T.; Laine, S.; Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *arXiv* **2017**, arXiv:1710.10196.
39. Zhang, H.; Goodfellow, I.; Metaxas, D.; Odena, A. Self-attention generative adversarial networks. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 7354–7363.
40. Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X. Improved Techniques for Training GANs. *arXiv* 2016, arXiv:1606.03498.
41. Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; Hochreiter, S. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *arXiv* **2018**, arXiv:1706.08500.
42. Brock, A.; Donahue, J.; Simonyan, K. Large scale GAN training for high fidelity natural image synthesis. *arXiv* **2018**, arXiv:1809.11096.
43. Karras, T.; Laine, S.; Aila, T. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4401–4410.
44. Park, T.; Liu, M.Y.; Wang, T.C.; Zhu, J.Y. Semantic image synthesis with spatially-adaptive normalization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 2337–2346.
45. Cai, W.; Wei, Z. PiiGAN: Generative adversarial networks for pluralistic image inpainting. *IEEE Access* **2020**, *8*, 48451–48463. [CrossRef]
46. Prangemeier, T.; Reich, C.; Wildner, C.; Koeppl, H. Multi-StyleGAN: Towards Image-Based Simulation of Time-Lapse Live-Cell Microscopy. *arXiv* **2021**, arXiv:2106.08285.
47. Xu, L.; Veeramachaneni, K. Synthesizing tabular data using generative adversarial networks. *arXiv* **2018**, arXiv:1811.11264.
48. Xu, L.; Skoularidou, M.; Cuesta-Infante, A.; Veeramachaneni, K. Modeling tabular data using conditional gan. *arXiv* **2019**, arXiv:1907.00503.
49. Chow, C.; Liu, C. Approximating discrete probability distributions with dependence trees. *IEEE Trans. Inf. Theory* **1968**, *14*, 462–467. [CrossRef]
50. Zhang, J.; Cormode, G.; Procopiuc, C.M.; Srivastava, D.; Xiao, X. Privbayes: Private data release via bayesian networks. *ACM Trans. Database Syst. (TODS)* **2017**, *42*, 1–41. [CrossRef]
51. Choi, E.; Biswal, S.; Malin, B.; Duke, J.; Stewart, W.F.; Sun, J. Generating multi-label discrete patient records using generative adversarial networks. In Proceedings of the Machine Learning for Healthcare Conference, PMLR, Boston, MA, USA, 18–19 August 2017; pp. 286–305.
52. Srivastava, A.; Valkov, L.; Russell, C.; Gutmann, M.U.; Sutton, C. Veegan: Reducing mode collapse in gans using implicit variational learning. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; Volume 30.
53. Park, N.; Mohammadi, M.; Gorde, K.; Jajodia, S.; Park, H.; Kim, Y. Data synthesis based on generative adversarial networks. *arXiv* **2018**, arXiv:1806.03384.
54. Rajabi, A.; Garibay, O.O. TabFairGAN: Fair Tabular Data Generation with Generative Adversarial Networks. *arXiv* **2021**, arXiv:2109.00666.
55. Andrews, G. What Is Synthetic Data? 2021. Available online: https://blogs.nvidia.com/blog/2021/06/08/what-is-synthetic-data/ (accessed on 14 February 2022).
56. Alanazi, Y.; Sato, N.; Ambrozewicz, P.; Blin, A.N.H.; Melnitchouk, W.; Battaglieri, M.; Liu, T.; Li, Y. A survey of machine learning-based physics event generation. *arXiv* **2021**, arXiv:2106.00643.
57. Assefa, S. Generating synthetic data in finance: Opportunities, challenges and pitfalls. In Proceedings of the International Conference on AI in Finance, New York, NY, USA, 15–16 October 2020.
58. Lan, L.; You, L.; Zhang, Z.; Fan, Z.; Zhao, W.; Zeng, N.; Chen, Y.; Zhou, X. Generative Adversarial Networks and Its Applications in Biomedical Informatics. *Front. Public Health* **2020**, *8*, 164. [CrossRef] [PubMed]
59. Chen, J.; Little, J.J. Sports camera calibration via synthetic data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Long Beach, CA, USA, 16–20 June 2019.
60. Barth, R.; IJsselmuiden, J.; Hemming, J.; van Henten, E.J. Optimising realism of synthetic agricultural images using cycle generative adversarial networks. In Proceedings of the IEEE IROS Workshop on Agricultural Robotics, Vancouver, BC, Canada, 28 September 2017; pp. 18–22.
61. Tremblay, J.; To, T.; Sundaralingam, B.; Xiang, Y.; Fox, D.; Birchfield, S. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv* **2018**, arXiv:1809.10790.

62. Nikolenko, S.I. Synthetic data for deep learning. *arXiv* **2019**, arXiv:1909.11512.
63. Batuwita, R.; Palade, V. Efficient resampling methods for training support vector machines with imbalanced datasets. In Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, Spain, 18–23 July 2010; pp. 1–8.
64. Drummond, C.; Holte, R.C. C4. 5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. *Workshop Learn. Imbalanced Datasets II* **2003**, *11*, 1–8.
65. Lusa, L. Evaluation of smote for high-dimensional class-imbalanced microarray data. In Proceedings of the 2012 11th International Conference on Machine Learning and Applications, Boca Raton, FL, USA, 12–15 December 2012; Volume 2, pp. 89–94.
66. Sun, J.; Lang, J.; Fujita, H.; Li, H. Imbalanced enterprise credit evaluation with DTE-SBD: Decision tree ensemble based on SMOTE and bagging with differentiated sampling rates. *Inf. Sci.* **2018**, *425*, 76–91. [CrossRef]
67. Sun, J.; Li, H.; Fujita, H.; Fu, B.; Ai, W. Class-imbalanced dynamic financial distress prediction based on Adaboost-SVM ensemble combined with SMOTE and time weighting. *Inf. Fusion* **2020**, *54*, 128–144. [CrossRef]
68. Lee, T.; Kim, M.; Kim, S.P. Data augmentation effects using borderline-SMOTE on classification of a P300-based BCI. In Proceedings of the 2020 8th International Winter Conference on Brain-Computer Interface (BCI), Gangwon, Korea, 26–28 February 2020; pp. 1–4.
69. Riafio, D. Using Gabriel graphs in Borderline-SMOTE to deal with severe two-class imbalance problems on neural networks. In *Artificial Intelligence Research and Development, Proceedings of the 15th International Conference of the Catalan Association for Artificial Intelligence, Alicante, Spain, 24–26 October 2012*; IOS Press: Amsterdam, The Netherlands, 2012; Volume 248, p. 29.
70. Siriseriwan, W.; Sinapiromsaran, K. The effective redistribution for imbalance dataset: Relocating safe-level SMOTE with minority outcast handling. *Chiang Mai J. Sci.* **2016**, *43*, 234–246.
71. Lu, C.; Lin, S.; Liu, X.; Shi, H. Telecom fraud identification based on ADASYN and random forest. In Proceedings of the 2020 5th International Conference on Computer and Communication Systems (ICCCS), Guangzhou, China, 21–24 April 2020; pp. 447–452.
72. Aditsania, A.; Saonard, A.L. Handling imbalanced data in churn prediction using ADASYN and backpropagation algorithm. In Proceedings of the 2017 3rd International Conference on Science in Information Technology (ICSITech), Bandung, Indonesia, 25–26 October 2017; pp. 533–536.
73. Chen, S. Research on Extreme Financial Risk Early Warning Based on ODR-ADASYN-SVM. In Proceedings of the 2017 International Conference on Humanities Science, Management and Education Technology (HSMET 2017), Taiyuan, China, 25–26 February 2017; pp. 1132–1137.
74. MacQueen, J. Classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*; University of California Press: Berkeley, CA, USA, 1967; pp. 281–297.
75. Sarkar, S.; Pramanik, A.; Maiti, J.; Reniers, G. Predicting and analyzing injury severity: A machine learning-based approach using class-imbalanced proactive and reactive data. *Saf. Sci.* **2020**, *125*, 104616. [CrossRef]
76. Jo, T.; Japkowicz, N. Class imbalances versus small disjuncts. *ACM Sigkdd Explor. Newsl.* **2004**, *6*, 40–49. [CrossRef]
77. Learn, S. Gaussian Mixture Models. 2022. Available online: https://scikit-learn.org/stable/modules/mixture.html (accessed on 23 February 2022).
78. Chokwitthaya, C.; Zhu, Y.; Mukhopadhyay, S.; Jafari, A. Applying the Gaussian Mixture Model to Generate Large Synthetic Data from a Small Data Set. In *Construction Research Congress 2020: Computer Applications*; American Society of Civil Engineers: Reston, VA, USA, 2020; pp. 1251–1260.
79. A Comprehensive Introduction to Bayesian Deep Learning. Available online: https://jorisbaan.nl/2021/03/02/introduction-to-bayesian-deep-learning (accessed on 11 February 2022).
80. Soni, D. Introduction to Bayesian Networks. 2019. Available online: https://towardsdatascience.com/introduction-to-bayesian-networks-81031eeed94e (accessed on 29 January 2022).
81. Russell, S.J.; Norvig, P.; Chang, M.W. Chapter 13: Probabilistic Reasoning. In *Artificial Intelligence: A Modern Approach*; Pearson: London, UK, 2022; pp. 430–478.
82. Goodfellow, I.; Bengio, Y.; Courville, A. Chapter 20: Deep Generative Models. In *Depp Learning*; MIT Press: Cambridge, MA, USA, 2016; pp. 654–720.
83. Foster, D. Chapter 3: Variational Autoencoders. In *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play*; O'Reilly: Sebastopol, CA, USA, 2019; pp. 61–96.
84. Goodfellow, I.; Bengio, Y.; Courville, A. Chapter 14: Autoencoders. In *Depp Learning*; MIT Press: Cambridge, MA, USA, 2016; pp. 502–525.
85. Zhang, X.; Fu, Y.; Zang, A.; Sigal, L.; Agam, G. Learning classifiers from synthetic data using a multichannel autoencoder. *arXiv* **2015**, arXiv:1503.03163.
86. Wan, Z.; Zhang, Y.; He, H. Variational autoencoder based synthetic data generation for imbalanced learning. In Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, USA, 27 November–1 December 2017; pp. 1–7.
87. Islam, Z.; Abdel-Aty, M.; Cai, Q.; Yuan, J. Crash data augmentation using variational autoencoder. *Accid. Anal. Prev.* **2021**, *151*, 105950. [CrossRef] [PubMed]
88. Fahimi, F.; Zhang, Z.; Goh, W.B.; Ang, K.K.; Guan, C. Towards EEG generation using GANs for BCI applications. In Proceedings of the 2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI), Chicago, IL, USA, 19–22 May 2019; pp. 1–4.

89. Patel, M.; Wang, X.; Mao, S. Data augmentation with Conditional GAN for automatic modulation classification. In Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning, Linz, Austria, 13 July 2020; pp. 31–36.
90. Ali-Gombe, A.; Elyan, E. MFC-GAN: Class-imbalanced dataset classification using multiple fake class generative adversarial network. *Neurocomputing* **2019**, *361*, 212–221. [CrossRef]
91. Ali-Gombe, A.; Elyan, E.; Savoye, Y.; Jayne, C. Few-shot classifier GAN. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
92. Sushko, V.; Gall, J.; Khoreva, A. One-shot gan: Learning to generate samples from single images and videos. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 2596–2600.
93. Niu, M.Y.; Zlokapa, A.; Broughton, M.; Boixo, S.; Mohseni, M.; Smelyanskyi, V.; Neven, H. Entangling quantum generative adversarial networks. *Phys. Rev. Lett.* **2022**, *128*, 220505. [CrossRef]
94. Zhang, W.; Ma, Y.; Zhu, D.; Dong, L.; Liu, Y. MetroGAN: Simulating Urban Morphology with Generative Adversarial Network. *arXiv* **2022**, arXiv:2207.02590.
95. Yann LeCun Quora Session Overview. Available online: https://www.kdnuggets.com/2016/08/yann-lecun-quora-session.html (accessed on 2 February 2022).
96. Anscombe, F.J. Graphs in statistical analysis. *Am. Stat.* **1973**, *27*, 17–21.
97. Zhou, Y.; Dong, F.; Liu, Y.; Li, Z.; Du, J.; Zhang, L. Forecasting emerging technologies using data augmentation and deep learning. *Scientometrics* **2020**, *123*, 1–29. [CrossRef]
98. Shmelkov, K.; Schmid, C.; Alahari, K. How good is my GAN? In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 213–229.
99. Alaa, A.M.; van Breugel, B.; Saveliev, E.; van der Schaar, M. How Faithful is your Synthetic Data? Sample-level Metrics for Evaluating and Auditing Generative Models. *arXiv* **2021**, arXiv:2102.08921.