mathematics

MDPI

Article

AutoGAN: An Automated Human-Out-of-the-Loop Approach for Training Generative Adversarial Networks

Ehsan Nazari *, Paula Branco and Guy-Vincent Jourdan

School of Electric Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada
* Correspondence: enaza030@uottawa.ca

Abstract: Generative Adversarial Networks (GANs) have been used for many applications with overwhelming success. The training process of these models is complex, involving a zero-sum game between two neural networks trained in an adversarial manner. Thus, to use GANs, researchers and developers need to answer the question: "Is the GAN sufficiently trained?". However, understanding when a GAN is well trained for a given problem is a challenging and laborious task that usually requires monitoring the training process and human intervention for assessing the quality of the GAN generated outcomes. Currently, there is no automatic mechanism for determining the required number of epochs that correspond to a well-trained GAN, allowing the training process to be safely stopped. In this paper, we propose AutoGAN, an algorithm that allows one to answer this question in a fully automatic manner with minimal human intervention, being applicable to different data modalities including imagery and tabular data. Through an extensive set of experiments, we show the clear advantage of our solution when compared against alternative methods, for a task where the GAN outputs are used as an oversampling method. Moreover, we show that AutoGAN not only determines a good stopping point for training the GAN, but it also allows one to run fewer training epochs to achieve a similar or better performance with the GAN outputs.

Keywords: generative adversarial models; automatic training

MSC: 68T01; 68T07

Citation: Nazari, E.; Branco, P.; Jourdan, G.-V. AutoGAN: An Automated Human-Out-of-the-Loop Approach for Training Generative Adversarial Networks. *Mathematics* **2023**, *11*, 977. https://doi.org/10.3390/math11040977

Academic Editors: Alvaro Figueira and Francesco Renna

Received: 23 January 2023
Revised: 9 February 2023
Accepted: 10 February 2023
Published: 14 February 2023

1. Introduction

Generative models are a crucial part of many important machine learning and computer vision algorithms. In recent years, they have been used in multiple applications with outstanding success (e.g., [1]). Generative Adversarial Networks (GANs) [2] are a subset of the generative models that have attracted the attention of the scientific community. GANs have also been used in a diversity of real-world applications involving data generation, such as image generation [3–7], image-to-image translation [8–10], image super resolution [11], video generation [12,13], music generation [14], graphs generation [15], or text generation [16]. They have also been applied to address other complex tasks, namely data de-identification [17], over-sampling [18,19], improving classification accuracy [20,21], dealing with missing values [22], trajectory prediction [23], or spatio-temporal prediction [24].

The procedure to train a GAN is more complex and challenging than training a standard learning algorithm [25,26]. In effect, to train a GAN, we do not aim at optimizing an objective function. Instead, the learning task is related to game theory, which is defined as a minimax problem in which two players compete against each other, one trying to maximize an objective function, while the other is trying to minimize it. The solution to this minimax problem could be the Nash equilibrium of the game [27]. However, finding the Nash equilibrium is a complex and difficult task when compared with optimizing an objective function [2,27].

Besides being a complex task, other important challenges arise when training a GAN. One of these challenges is the lack of a systematic criterion to evaluate when the GAN is already sufficiently trained for the task at hand as the training epochs take place. Several attempts have been made to address this critical problem, with a growing body of research being put forward through the proposal of multiple alternative measures. So far, no single measure has been identified as the gold standard method for use across multiple applications and/or data modalities.This leaves researchers and end-users with the problem of determining when to stop the GAN training process for a given task through a non-systematic trial-and-error procedure. In this paper, we focus on finding a systematic solution to the problem of when to stop training a GAN that can be used across different data modalities.

The currently existing metrics for addressing the described problem can be categorized into 'qualitative evaluation' and 'quantitative evaluation'. The former involve human judgment, while the latter are based on different mathematically defined distance functions. A commonly used qualitative method of evaluating when to stop training a GAN is through human visual inspection of the generated samples [1]. This is a very direct and intuitive way of evaluating the quality of images. Nevertheless, it is a very costly and time-consuming task that cannot be applied to tabular data. The visual inspection process was regulated and standardised in a more systematic manner [1,28–30]. However, this solution has multiple disadvantages that prevent it from being broadly adopted. Namely, the following issues were found: (i) in some domains (e.g., medical domain), it might be difficult for a human to learn what is realistic or not; (ii) there are limitations regarding the number of images that can be reviewed in a reasonable period of time; (iii) the reviewers' biases and opinions may be incorporated into the process [30]; and (iv) visual inspection methods are restricted to images and cannot be applied to tabular data.

Regarding quantitative evaluation solutions, several measures have been proposed for assessing GANs trained on image datasets. Inception Score (IS) [31] and the Fréchet Inception Distance (FID) [32] are relatively popular measures in this category. Their goal is to use a large, pre-trained model, the InceptionNet in this case, to derive a good metric of the quality of the generated images. Other solutions, such as the Fréchet Confidence and Diversity (FCD) score, exchange the use of the InceptionNet model with an Autoencoder. Still, all these solutions will output a score, and it will be the end-user's responsibility to decide whether the GAN's training process can be stopped. Other quantitative evaluation metrics are trained on both imagery and non-imagery data. For instance, in [8,33,34], measures based on classification accuracy are introduced. While quantitative measures do not encounter the same issues as direct human (qualitative) evaluation, they may not directly correspond to how humans view and judge generated samples [1].

A new type of GAN called Wasserstein GAN (WGAN) was proposed in [35] to remove the oscillatory behaviour observed on the loss values of GAN components. WGAN can be observed as a way of incorporating qualitative measures into the loss functions of a GAN. Plotting WGAN learning curves has multiple applications, including improving sample quality explanability, which can help with the problem of deciding when to stop training GANs. However, WGAN does not allow the comparison of results between different GAN architectures, the estimation of the Wasserstein distance may be inaccurate [35], and WGAN still requires human visual validation in imagery datasets when observing the loss values and GAN performance.

Overall, the qualitative measures proposed require human intervention during GAN training, while quantitative measurements that address certain shortcomings of qualitative measures still require human monitoring of the metric during training. Therefore, the human remains in the loop for both types of measures. Moreover, from a data modality perspective, we observe that qualitative measurements are exclusively applicable to GANs trained on imagery datasets. In addition, the majority of the popular quantitative measures are also tailored for imagery datasets. Consequently, not only are the possibilities for applying GANs to non-image datasets limited, but human inspection or supervision is still

required. In this paper, we tackled these gaps by proposing a **human-out-of-the-loop algorithm, named AutoGAN, where the usage of quantitative measures is fully automated**. In a nutshell, AutoGAN starts the training of a GAN and evaluates the improvements achieved at each iteration using an oracle. Note that we use the term oracle in the classical sense of Computing Theory, that is, a machine that (via some black-box process) solves a decision problem in constant time (see, e.g., [36]). The oracle is a central component of the algorithm that encapsulates the end-user preferences for the task at hand. An oracle should provide a score for a given generator, which should correspond to the generator's ability to synthesize "better". This score should match the end-user goals and thus define the end-user's understanding of the notion of "better" samples. We provide several examples of oracle instances in this paper to illustrate the different oracles that can be used. AutoGAN will rely on the oracle outputs to assess the training of the GAN. Furthermore, it will allow the performance to deteriorate while waiting for the GAN to recover. This is necessary to deal with the known oscillatory behaviour of the GAN's performance. When no further improvements are observed for a certain number of consecutive iterations, the AutoGAN will return the overall best GAN model obtained during this process. AutoGAN requires minimal human intervention and is applicable to different data modalities (tabular and images). Our extensive experiments demonstrate a clear advantage of using AutoGAN, even when compared to GANs trained under a thorough human visual inspection of the generated images.

The key contributions of this paper are four-fold:

- Present a comprehensive review of the literature on: (i) multiple distances and performance measures that can be used to assess the performance of a GAN; and (ii) existing algorithms involving automation in GANs;
- Introduce the AutoGAN Algorithm, a new automatic human-out-of-the-loop approach for determining when to stop training a GAN that is applicable to a variety of data modalities, including imagery and tabular datasets;
- Provide an extensive experimental comparison using multiple imagery and tabular datasets that include multiple GAN evaluation metrics;
- Provide all of our code so that AutoGAN can be easily used and to allow the reproducibility of our research.

This paper is organized as follows. Section 2 describes the background and related works. In Section 3, we present our algorithm, AutoGAN, designed to solve the problem of automatically determining when to stop training a GAN. We also provide in this section a set of oracle instances that can be used in our AutoGAN Algorithm. In Section 4, we describe the experiments carried out, including the datasets considered and the settings used, while in Section 5, we present and discuss the main results of the experiments. Finally, Section 6 concludes the paper and provides some interesting future research avenues.

2. Background and Related Work

This section starts by providing a brief background on GANs. Then, we review multiple distance measures and provide a detailed explanation of how they can be used to evaluate the quality of GANs. Finally, we review related works on automation in GANs that involve a neural architecture search.

2.1. Generative Adversarial Networks

Generative Adversarial Networks are a system composed of two differentiable functions, the generator and the discriminator [2]. The generator receives a sample drawn from a multivariate distribution and maps it to a sample from another distribution. The discriminator's goal is to discriminate between the synthetic samples obtained through the generator and the real samples. In the first phase of each iteration of the learning procedure, the generator attempts to deceive the discriminator into accepting its outputs as actual data, while in the second phase, the discriminator attempts to discern between real and fake samples. The generator and discriminator are trained together, competing against

each other, and thus we can think about them as adversarial in the game theory sense. If trained on an image dataset, after a significant number of repetitions, the generator will be able to produce samples that are nearly indistinguishable from the actual images, i.e., the generator will generate synthetic samples from the distribution of the dataset.

A simple vanilla GAN does not need the class labels of the examples; thus, it can be regarded as an unsupervised model. Other variants of GANs, such as the Conditional Generative Adversarial Networks (CGANs) [37], use the class labels during the training process as an extra input to both the generator and the discriminator. The advantage of CGANs is that, after training, the generator provides the ability to generate samples from a specific class, whereas in a vanilla GAN, that control is not present.

2.2. Relevant Distances and Performance Measures

Measuring the quality of a GAN is crucial. In this subsection, we study several measures that have been used to assess the quality of GANs, including some distances and other measures used for performance assessment. We will focus on three particular measures: the Kullback–Leibler divergence, the Wasserstein distance, and the F1-score. We include here the F1-score as an example of a potentially interesting performance assessment metric that is useful in many problems involving GANs due to their application to imbalanced domains. Other metrics could be used, but the F1-score is one of the most frequently applied.

The Kullback–Leibler (KL) divergence is a well-known statistical distance that measures how similar two given distributions are [38]. Consider two distributions P and Q, where p and q denote the probability densities of P and Q, respectively. The KL-divergence is defined as shown in Equation (1).

$$D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx \tag{1}$$

An alternative way to compute the distance between two probability distributions is provided by the Wasserstein distance [39]. Let X and Y be two random variables with finite p-moments, $X \sim P$ and $Y \sim Q$. Assume that $J(P, Q)$ represents all joint distributions J for random variables (X, Y). The p-Wasserstein distance is defined in Equation (2).

$$W_p(P, Q) = \left(\underset{J \in J(P,Q)}{inf} \int \|x - y\|^p dJ(x, y) \right)^{1/p} \tag{2}$$

where $p \geq 1$. A special case of this distance is when $p = 1$. This distance is called the Earth Mover or 1-Wasserstein distance. The 2-Wasserstein distance is also called the Fréchet distance.

To determine the performance of a model, one can take advantage of the F1-score. This metric depends on the notions of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), which are the cases correctly classified from the positive and negative class and the cases that were incorrectly classified as positive or negative class cases, respectively. The F1-score is defined as the harmonic mean of the precision (cf. Equation (3) and recall (cf. Equation (4). When dealing with binary classification problems, typically the F1-score of the minority class (or positive class, or class of interest) is reported as shown in Equation (5). The F1-score can also be calculated for each class in the domain, and its macro- or micro-average variants can be used as the global F1-score on a multiclass problem.

$$\text{precision} = \frac{TP}{TP+FP} \tag{3}$$

$$\text{recall} = \frac{TP}{TP+FN} \tag{4}$$

$$\text{F1-score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} = \frac{2 * TP}{2 * TP + FP + FN} \tag{5}$$

2.3. From Distances and Measures to Experimental Configurations for Evaluating the Performance of GANs

Different distances and measures can be used to assess the performance of a GAN. However, the setting in which these distances or measures are used can also change substantially. This section discusses these configurations in terms of the experimental setting they advocate as well as their underlying assumptions. We will review six configurations, explaining in detail how they are used to assess the performance of GANs.

2.3.1. Classification Accuracy Score: Train on Synthetic Data, and Test on Real Data

Ravuri and Vinyals [33] argue that if we have a good and well-trained generative model that accurately captures the data distribution, then any downstream task should perform similarly regardless of whether it uses generated or original data. The authors train several class-conditional generative models, such as CGANs and variational auto-encoders (VAEs), on real and labeled data. Following this step, a classifier is trained on synthetic data and used to predict the label of real images. A performance assessment metric such as the F1-score can be used to evaluate the resulting model. This configuration, termed CAS-real, has also been studied in [34], where the authors state that this setting shows how diverse the samples generated by a GAN are. While we can apply CAS-real to GANs trained on both imagery and tabular data, it requires a multi-class, labeled dataset. Furthermore, it can only be used with conditional GANs, which might be a disadvantage in some deployment scenarios.

2.3.2. Classification Accuracy Score: Train on Real Data, and Test on Synthetic Data

If we assume that the images generated with a well-trained GAN are realistic, then the classifiers trained on real images should have no problem identifying the synthesized image correctly as well. This assumption was put forward by Isola et al. [8], motivating the configuration we termed CAS-syn. Compared to the previous CAS-real configuration, CAS-syn switches the roles of synthetic and real data, i.e., in CAS-syn, a classifier is trained on real data and tested on synthetic data, while the reverse happens for CAS-real, which uses real data on the training phase and synthetic data on the testing phase. CAS-syn has also been studied in [34], where the authors argue that the CAS-syn configuration can measure how well the generated samples approximate the unknown real distribution on image data. GANs trained with either imagery or tabular data can use this configuration. However, CAS-syn requires a multi-class labeled dataset, and it is only applicable to CGANs.

2.3.3. Inception Score

A different approach named the Inception Score (IS) is proposed by Salimans et al. [31]. To begin, the conditional label distribution $p(y, x)$ is computed by applying an InceptionNet model to each image produced by a GAN trained on the original dataset. The assumptions underlying this approach are that: (i) the GAN should generate images with high confidence for each label, i.e., $p(y|x)$ has a low entropy; and (ii) the GAN should produce varied images, i.e., the marginal $\int p(x = G(z))\, dz$ has a high entropy. Based on these two requirements, the authors propose the use of the following metric $\exp[\mathbb{E}_{x \sim p_g} D_{KL}(p(y|x)||p(y))]$, which they claim has a direct correlation with human judgment. The expected value of KL-divergence is exponentiated to allow for an easier comparison.

Although this is an extensively used approach, several weaknesses of this configuration must be taken into account. In particular, the disadvantages of IS have been enumerated in [30] and include: (1) sensitivity to parameters and implementations of model parameters; (2) biased towards ImageNet dataset and InceptionNet models; (3) diversity within classes is not captured; (4) requires a high sample size to be reliable; (5) a high IS is achievable by inputting only one example per ImageNet class; and (6) it can only be used for GANs trained on certain imagery datasets because the InceptionNet model uses colored images as inputs, uses convolutional layers, and is trained in the ImageNet dataset. Therefore, IS cannot be used with either black-and-white images or tabular data.

Moreover, the usage of IS on GANs trained on colored images other than ImageNet can be misleading [40].

2.3.4. Confidence and Diversity Score

The methodology described for IS can be altered to obtain a different configuration named the Confidence and Diversity Score (CDS). In this setting, we carry out the following steps: (1) use a neural network classifier with an arbitrary architecture; and (2) train the classifier on the target dataset instead of the ImageNet dataset [41]. This modification allows this new score to be applicable to GANs trained on non-imagery and imagery data, whereas the IS is confined to a special area of imagery data. However, the CDS requires the data to be clustered into two or more classes and requires the class labels to be available in order to calculate the score, while the original IS does not require class labels of the target dataset.

2.3.5. The Fréchet Inception Distance

Heusel et al. [32] proposes to use the features extracted from both real (r) and generated data (g) to assess the quality of images created by a GAN. Typically, an InceptionNet model is trained on the ImageNet dataset, and the results of the last pooling layer prior to the output classification layer are used as the extracted features.

The extracted features from real (X_r) and generated (X_g) data are viewed as continuous multivariate Gaussian distributions, i.e., $X_r \sim \mathcal{N}(\mu_r, \Sigma_r)$ and $X_g \sim \mathcal{N}(\mu_g, \Sigma_g)$, where μ_r and μ_g are the means of X_r and X_g and Σ_r and Σ_g are the covariance matrices of the two distributions. The parameters of the two underlying distributions are then estimated and the Fréchet distance between the two distributions is calculated. The Fréchet Inception Distance (FID) is used to compare the similarity between two multivariate Gaussian distributions, and is calculated as shown in Equation (6) for X_r and X_g.

$$d^2 = |\mu_X - \mu_Y|^2 + tr(\Sigma_X + \Sigma_Y - 2(\Sigma_X\Sigma_Y)^{1/2}) \tag{6}$$

The smaller the FID is, the closer the two estimated distributions are; consequently, the better the GAN outputs are. Unlike IS, FID captures the diversity within classes [30]. FID, on the other hand, is highly biased [30]; and small samples sizes can result in an overestimation of the true FID [30]. Although both FID and IS are based on the InceptionNet, the IS uses the trained InceptionNet model as is, while the FID uses this model as a feature extractor. Given that the InceptionNet is trained on colored images (the imageNet dataset), this could make both IS and FID only applicable to gray-scale images. However, the FID has the potential to be more generic and could be applied to both colored and gray-scale images, given the fact that the InceptionNet is merely used as a feature extractor. We will test the versatility of FID by also testing it on black-and-white images in our experiments to confirm this hypothesis.

2.3.6. Fréchet Confidence and Diversity Score

Modifications to FID were proposed by Obukhov and Krasnyanskiy [41] to obtain the Fréchet Confidence and Diversity (FCD) score. In this version, the InceptionNet model is replaced by an auto-encoder model trained on the target data. After training the auto-encoder, the encoder is used as a feature extractor. Different sizes of the encoder's output layer, i.e., the number of extracted features, were investigated.

Given the results obtained in this work [41], the authors considered that while the FCD score has a correlation with the quality of generated images, the same does not happen for the generated tabular and non-imagery data. The authors based this conclusion on the fact that no correlation was observed with a reduction in the errors of the generator and the discriminator. However, the known oscillatory behaviour of the generator and discriminator [35] leads us to conclude that a decrease in the errors does not necessarily

imply proper training. When we present our experimental results in Section 5, we will go over this in greater detail.

2.4. Algorithms Involving Automation in GANs

Regarding automation in GANs, several efforts have been made to search for the best generator architecture in GANs. The proposed neural architecture search methods have provided good outcomes as they outperform architectures that are manually designed on multiple tasks [42,43]. In this field, Wang et al. [44] presented an algorithm for an automated neural architecture search for deep generative models. On the other hand, Gong et al. [45] define the search space for the generator architectural variations and propose the usage of a Recurrent Neural Network (RNN) to guide the search process. This study used the IS as the reward and applied a multi-level search strategy to search for the neural architecture progressively. Still, several research avenues remain to be explored, such as increasing the search space and also extending this search to the discriminator, incorporating class labels, or testing the search on high resolution images.

The research with automation gives rise to other concerns, for instance, related to the model's size. Building on the neural architecture search solutions, Fu et al. [43] presented the AutoGAN-Distiller (AGD), the first AutoML framework dedicated to GAN compression. However, several aspects remain to be studied. One of the main problems when training a GAN with a specified architecture is determining when the GAN is well trained and when the process can be stopped. However, as far as we know, no effort has been made to automate the process of training a GAN for a given architecture. In effect, many applications dealing with images rely on domain experts to evaluate when the images being generated have the required quality and thus stop the training procedure of the GAN. A domain expert needs to define when the GAN has trained for a sufficient number of iterations and the training process stops with his input. Moreover, frequently, the generated images are analysed to obtain that answer, which is not only a time-consuming process but can also be biased and susceptible to the subjectivity of the expert. Finally, such inspection is not possible when non-imagery data is used. The AutoGAN solution that we present in the following section seeks to answer these questions.

3. Proposed AutoGAN Method

This section describes our proposed solution, AutoGAN, for automating the training process of GANs. We begin by defining the goals and requirements of AutoGAN and then proceed to the details of our algorithm. Finally, we provide a detailed illustration of several oracle instances, an important component of AutoGAN. Notice that the term oracle is used in the classical sense of Computing Theory, representing a machine that (via some black-box process) solves a decision problem in constant time (e.g., [36]).

3.1. Algorithm Goals and Requisites

The goal of our solution is to provide the end-user with a GAN model that is well trained for the specific target task. To achieve this, the end-user needs to provide the task requirements as well as the particular GAN architecture that needs to be trained. Besides providing these settings, which reflect the preferences for the task at hand, the end-user will have no more intervention in the AutoGAN algorithm. Our AutoGAN algorithm will use this information to output the trained GAN model given the provided end-user preferences. This way, the end-user obtains a trained model in a fully automated way, i.e., except for the design aspect there is no human intervention, such as, for instance, setting the number of epochs to run or inspecting the results at certain checkpoints to determine whether the training process can be stopped.

AutoGAN includes a critical component that we named the oracle. This oracle is defined by the end-user and should encapsulate the requirements of the tasks being addressed. It is the user's responsibility to define the correct oracle for the task he aims to solve. The oracle can be observed as a function that receives the GAN generator and

outputs a score corresponding to the ability of the generator to synthesize high-quality samples. The oracle parameters are defined by the end-user. Section 3.3 contains a detailed description of several oracle instances that can be used with our AutoGAN algorithm in various contexts and with different goals.

3.2. The AutoGAN Algorithm

When deciding when to stop training a GAN on imagery datasets, a human is typically in the loop and visually inspects the GAN's output samples to make that decision. On tabular data, this inspection is impossible to be carried out, and thus the task becomes harder. Some metrics can be monitored, but a human should still be involved in the process of deciding when to stop training. Frequently, we observe that researchers limit the GAN training to an arbitrary number of epochs, leading to GAN models trained sub-optimally.

To address this problem, we propose a systematic approach, AutoGAN, that automates the training process involving a quantitative measure. Our solution eliminates humans from the process of deciding when enough epochs have been run and the training of the GAN can be stopped. AutoGAN is an automated human-out-of-the-loop approach for training GANs that allows an end-user to determine when to stop training a GAN in a fully automatic way and without the need of inspecting any data, images, or metrics.

The key idea of the AutoGAN algorithm is to allow the training of a GAN to continue even when the GAN output samples do not show any improvements after several training iterations. This means that AutoGAN aims to provide the GAN with sufficient opportunities to overcome potential local unfavorable points, allowing it to reach an improved model. AutoGAN depends on the end-user definition of an oracle instance. AutoGAN uses the oracle instance provided to assess the quality of the output samples in an iterative manner until no further improvement in the GAN is expected. For a certain iteration of GAN outputs, the oracle produces a scalar score that corresponds to the quantitative measure that the end-user considers suitable to be monitored to estimate the GAN's performance.

The pseudocode of our proposed solution is presented in Algorithm 1. We consider the four following inputs: (1) the maximum number of failed attempts, which encapsulates how long we are willing to wait without observing any improvements in the GAN; (2) the unit used for the training iterations; (3) an untrained GAN with a particular architecture; and (4) an instance of an oracle that contains the task requirements set by the end-user. After the initialization, the GAN is trained in an iterative way. At each iteration, the GAN is trained and evaluated using the oracle settings. If a better solution is found, then it is stored as the best GAN trained up to that moment, and the best score achieved is also recorded. If the GAN obtained is worse than the best GAN stored, then that solution is discarded and a new iteration is run. This local deterioration of the performance is expected to happen due to the known relationship between the generator and discriminator losses [26]. For this reason, we need to assume that a certain number of consecutive failed attempts will happen. However, when the maximum number of failed attempts is reached, i.e., when the GAN was trained for the maximum number of consecutive rounds set without improvements, then, the algorithm stops and provides the best model that was obtained.

3.3. Potential Oracle Instances

An important component of our algorithm concerns the definition of an oracle instance. This task, although being the end-user's responsibility, can be difficult and needs to be carefully considered as the evaluation of AutoGAN will depend on the oracle defined. Thus, the instantiation of the oracle is a crucial step. This section provides an illustration of multiple oracle instances that can be used in different contexts concerning specific data and GAN requirements. Namely, we will describe oracle instances that have different assumptions regarding the characteristics of the used data (e.g., modalities or availability of class labels) and the GAN architectures.

Algorithm 1 The AutoGAN algorithm

Input: *Max_failed_attempts*: The maximum number of failed attempts accepted;
Train_unit: The unit used for training iterations;
Untrained_GAN: The GAN architecture selected and not trained;
Oracle: The selected oracle instance;
Output: *best_GAN*: The trained GAN model

.
$current\_GAN \leftarrow Untrained\_GAN$
$best\_GAN \leftarrow Untrained\_GAN$
$best\_score \leftarrow -\infty$
$failed\_attempts \leftarrow 0$
while $failed\_attempts \leq Max\_failed\_attempts$ **do**
  Train *current_GAN* for one *Train_unit*
  $score \leftarrow oracle(current\_GAN)$
  if $score \geq best\_score$ **then**
    $failed\_attempts \leftarrow 0$; `// the new trained GAN is better than the best known`
    $best\_GAN \leftarrow current\_GAN$
    $best\_score \leftarrow score$
  else
    $failed\_attempts \leftarrow failed\_attempts + 1$; `// the trained GAN is worst than the best known`
  end
end
return *best_GAN*

As previously mentioned, an oracle is a function that maps a given generator into a score that corresponds to the generator's ability to synthesize "better" samples. The oracle instance should define what the end-user understands by the term "better" sample. For instance, in some cases, a "better sample" might be to approximate as closely as possible the distribution of the dataset. However, in other situations, a "better sample" can indicate the presence of more diversity in the data or even, in the case of images, higher quality and sharpness. Figures 1–6 provide an overview of the six oracle instances that we will describe in more detail in the next sections.

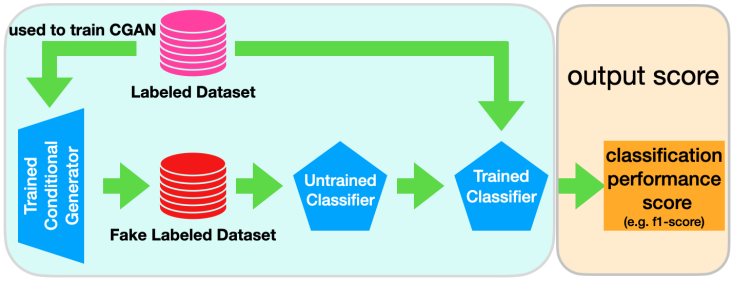

Figure 1. The architecture of an oracle instance based on CAS-real score.

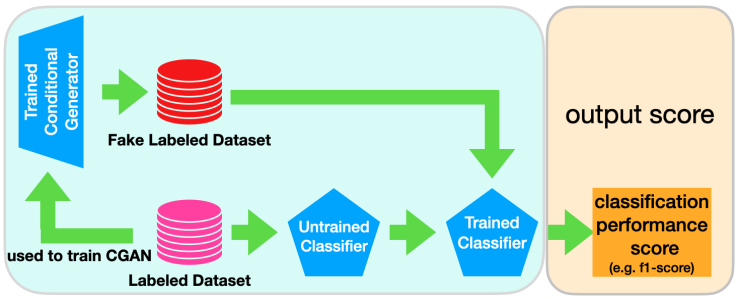

Figure 2. The architecture of an oracle instance based on CAS-syn score.

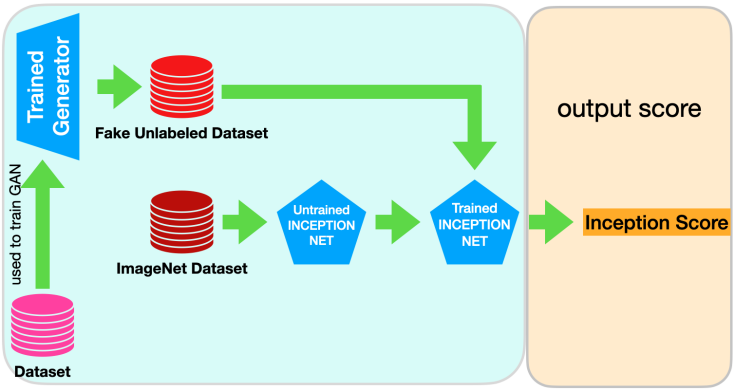

Figure 3. The architecture of the oracle based on IS.

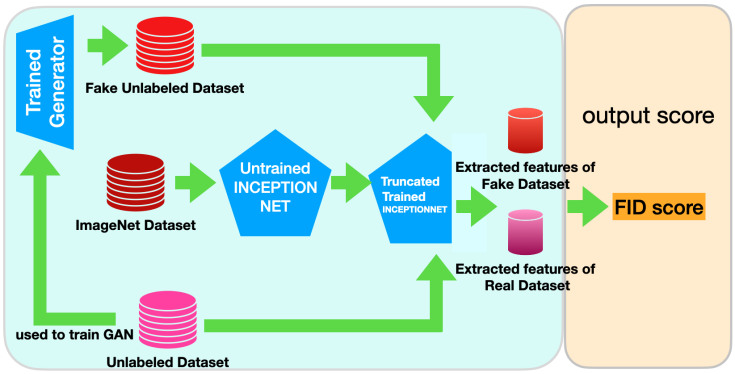

Figure 4. The architecture of an oracle instance based on FID Score.

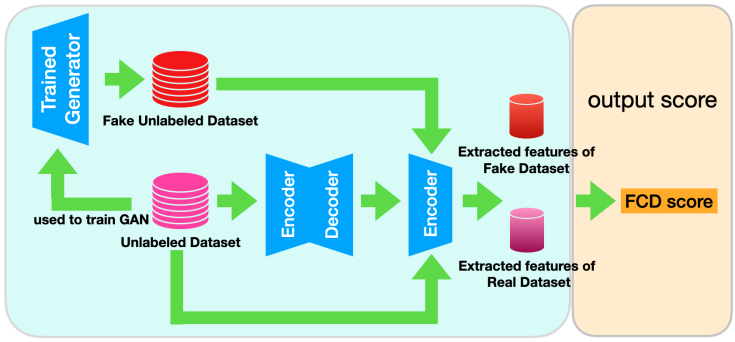

Figure 5. The architecture of an oracle instance based on FCD score.

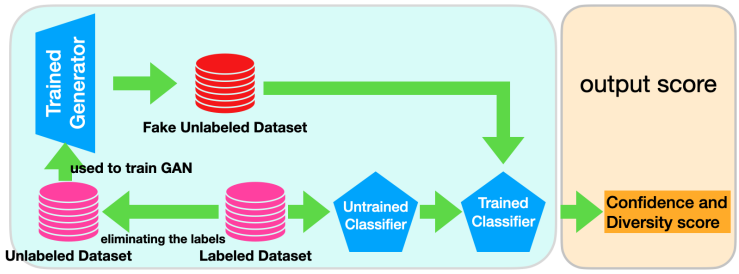

Figure 6. The architecture of an oracle instance based on Confidence and Diversity score.

The oracle instance used for a given task reflects the end-user preferences, and should be adapted to different deployment scenarios. Moreover, the oracle instance is not simply a metric or score used; it involves the entire architecture used to compute that score. For instance, a certain oracle instance may use only synthetic data to compute the desired score,

while another oracle instance can use only real data to compute the same score. Other oracles may use some or all layers of a given neural network, while others might use a different classifier for that task. For this reason, the oracle entity encapsulates multiple settings that can and should be adapted to the specific task being solved, and thus, the oracle instances differ from other existing fixed solutions.

3.3.1. Oracle Instance Based on CAS-Real

This oracle begins by using the available labelled data to train a CGAN. Then, the CGAN is asked to generate a labelled dataset, which is fed to a classifier. The classifier is trained using the generated data. Finally, the trained classifier's performance is evaluated on the real labelled dataset. Any selected performance assessment metric can be used to evaluate the performance of the classifier. In our implementation, we selected the F1 score to be used because we decided to test this oracle instance in an imbalanced problem, and the F1 score is a suitable metric in this context. In this particular case, the higher the CAS-real score, the higher the quality of the GAN outputs. Figure 1 displays the key structure of the described oracle instance based on CAS-real.

3.3.2. Oracle Instance Based on CAS-syn

This oracle instance works in a similar way as the previous one that was based on CAS-real. The main difference concerns a change in the roles of real and synthetically generated data. This oracle starts by training a classifier with real labelled data. This same real data is also used to train a CGAN. Then, the generator of the CGAN is used to generate a synthetic test set utilized to assess the performance of the classifier. The trained classifier is tested on the generated labeled test set and the performance is recorded using a selected performance assessment metric. In our implementation, we used the F1 score because we tested this oracle in an imbalanced domain. In this case, the higher the CAS-syn score, the higher the quality of the GAN outputs. An overview of an oracle instance based on CAS-syn is shown in Figure 2.

3.3.3. Oracle Instance Based on Inception Score

In this oracle, the InceptionNet model is first trained on the ImageNet dataset, while the GAN is trained on the given real dataset. The GAN generator is then used to obtain new samples to be fed into the InceptionNet model. The IS is computed based on the output vectors of the model and used to assess the quality of the GAN. The higher the IS, the better the quality of the GAN outputs. This oracle can be used with both conditional and non-conditional GANs. An overview of the described oracle instance is displayed in Figure 3. We must highlight that this oracle based on IS is not applicable to tabular data or black and white images, as explained in Section 2.3.

3.3.4. Oracle Instance Based on Fréchet Inception Distance

This oracle instance also relies on the InceptionNet but uses a truncated version of this neural network architecture. The process starts with the training of the InceptionNet model on the ImageNet dataset. The target real dataset is used to train a GAN, whose generator is then used to obtain a certain number of fake samples. The real and fake data are given to the truncated version of the InceptionNet model to obtain their InceptionNet-represented features. Finally, the extracted features of both the fake and real data are used to calculate the FID score. The lower the FID score, the higher the quality of the GAN outputs. To obtain a positive correlation between the FID and performance in our implementation, we multiplied the FID score by minus one. This oracle can be used with both conditional and non-conditional GANs. Figure 4 shows the overview of an oracle instance based on FID.

3.3.5. Oracle Instance Based on Fréchet Confidence and Diversity Score

In [46], an extensive study is performed comparing different metrics to the original FID. The authors test several measurements that are based on the InceptionNet model pre-

trained on different datasets. Other metrics were also tested using several self-supervised models that were used as feature extractors. Furthermore, the applicability of the FCD score on imagery and tabular data was investigated in [41].

This oracle instance is based on FCD and begins by training a GAN and an auto-encoder using the available real dataset. Then, the GAN generator is used to generate new fake data. The real and generated data are given to the encoder so that we can obtain a new representation of their features. Finally, the extracted features of both real and generated data are used to calculate the FCD score. The lower the FCD score is, the higher the quality of the GAN outputs. Similar to the modified FID introduced in [41], we implemented an oracle using the FCD, as shown in Figure 5. We also included a minor implementation detail by multiplying the FCD score by minus one; this allowed us to observe a positive correlation between the FCD score and the quality of the outputs. This oracle instance can be used with both conditional and non-conditional GANs.

3.3.6. Oracle Instance Based on Confidence and Diversity Score

Since the introduction of the IS in [31], it has been widely used in the assessment of imagery generative models [4,9,47,48]. The InceptionNet model consists of 2D-convolutional networks, which are suitable for processing images. Moreover, IS is obtained via an InceptionNet model that is pre-trained on the ImageNet dataset. Therefore, as mentioned in [41], the application of the IS is not possible for real-valued tabular data.

By replacing the InceptionNet model with an arbitrary neural network classifier trained on the target dataset, the applicability of the Confidence and Diversity Score on imagery datasets is investigated in [41]. As shown in Figure 6, we built an oracle instance that uses the CDS to run tests on both imagery and tabular data. This oracle uses the end-user dataset to train both a neural network classifier and a GAN model. Then, the GAN's generator is used to obtain a certain number of synthetic samples that are used as the classifier's test set. Finally, the CDS is calculated based on the model's results on this test set. The higher the CDS, the better the quality of the GAN outputs. This oracle can be used with both conditional and non-conditional GANs.

3.3.7. Overview of the Oracle Instances and Their Characteristics

The different oracle instances described have multiple assumptions regarding the data and the GAN architecture, and thus can be used in diverse contexts. These instances serve as examples of possible oracle instances that can be used. However, the specific task requirements and the end-user preferences should drive the selection of the oracle instance to be used for a certain problem involving training a GAN. A comparison between the different oracle instances used in this paper is shown in Table 1. We observe that some oracles are only suitable for being used with CGANs (CAS-real and CAS-syn), while others can use any selected GAN. In terms of the required class labels, only FCD does not require the existence of labeled data during the training, while all the other alternatives discussed require those labels. We also verify that all oracles can be applied to imagery data, while only four of the implemented oracles are applicable to tabular data. Table 1 also displays the number of times the network/classifier in the oracle is required to train (column "train times"). The indication of "one time" means that it is trained in the beginning and no further training is necessary, and "multiple" means that a new training process is necessary after each change in the GAN weights. CAS-real is the only oracle instance that needs to have the training repeated multiple times to obtain the desired score. The remaining oracles only need to be trained once. By comparing all the six implemented oracle instances, we observe that the FCD is the most flexible, working for both tabular and image data while not requiring class labels and allowing the usage on any GAN architecture. We must also highlight that all oracle instances can be used with colored images and almost all can also be used with black-and-white images. Following some preliminary test, we observed that the oracle IS is not suitable for black-and-white images.

Table 1. A comparison between the requirements of the described oracle instances.

Oracle Instance	Required Labels		Type of GAN	Type of Data			Train Times	Metric	Source
	Labeled Data during Training	Labeled Data to Generate Score		Imagery Color	Imagery B & W	Tabular			
CAS-real	Required	Required	Requires a CGAN	✓	✓	✓	Multiple	F1-score	[33,34]
CAS-syn	Required	Not required	Requires a CGAN	✓	✓	✓	One time	F1-score	[8,34]
IS	Required	Not required	Any GAN	✓			One time	KL-divergence	[31]
FID	Required	Not required	Any GAN	✓	✓		One time	Fréchet distance	[32]
CDS	Required	Not required	Any GAN	✓	✓	✓	One time	KL-divergence	[41]
FCD	Not required	Not required	Any GAN	✓	✓	✓	One time	Fréchet distance	[41]

4. Experimental Evaluation

This section presents all the experiments carried out to observe the performance of our proposed AutoGAN algorithm. We begin by providing an overview of our experiments. Then, we describe the datasets used and provide the complete experimental settings. To allow the reproducibility of our results, all the code used in this paper is freely available to the research community at https://github.com/enazari/autoGAN (accessed on 8 February 2023).

4.1. Experiments' Overview

Our goal is to assess the ability of the AutoGAN algorithm (cf. Algorithm 1) in addressing the "when to stop training a GAN" problem. To verify whether a GAN is well-trained (or if the training process is automatically stopped at a good point), we use the GAN as an over-sampling tool under a class imbalance setting. Our assumption is that, if the GAN is conveniently trained, it will provide advantages similar to those of GANs that have been carefully trained with human intervention. This experimental setting allows us to inspect the effectiveness of our proposed solution in both imagery and tabular datasets.

The key structure of the experiments carried out is as follows. We start by assessing the performance of a certain classifier on a given imbalanced domain to observe the baseline performance (Initial method). Then, we train a GAN using different methods that will determine when the training should be stopped. Each one of these GAN models is then employed to generate new synthetic samples, which are used to balance the number of examples in the training set. The performance of the classifiers trained on the different balanced training sets is assessed and will determine whether the GANs generated high-quality synthetic data and thus were well trained. We must highlight that the synthetic cases are only used in the training set, and the test set is kept untouched and separated from the remaining data that are used exclusively to test the classifier. In particular, we are interested in observing whether AutoGAN is able to find a stopping iteration that leads to a well-trained GAN. If the outcomes of our proposed method are at least equal to those of the alternative methods, we may conclude that we have successfully addressed the quandary of when to stop training a GAN.

In our experiments, we use different alternative methods to decide when to stop training the GAN. The different alternatives tested depend on the type of dataset used (imagery or tabular). Overall, we tested the following four main alternatives: (i) **Initial**: a baseline where the original imbalanced dataset is used to train a classifier; (ii) **Fixed**: GAN trained for a fixed number of iterations, selected based on experiments from other research papers or experiments/guidelines where the GAN shows good results; (iii) **Manual**: GAN trained using human visual inspection to determine the number of iterations required to obtain a well-trained GAN; and (iv) **AutoGAN**: GAN automatically trained using the AutoGAN algorithm and one of the oracle instances defined in Section 3.3. We compare the performance obtained through these different methods against each other and the baseline (Initial).

To make the comparisons fair, we start the training process of a certain GAN and use the different alternative methods in parallel to determine when the process should stop, as shown in Figure 7. As a result, we obtain the final state of the GAN using different

methods while following the same training process. The GAN's training process is only stopped after all tested methods provide a stopping signal. By testing our solution using this framework, we are able to assess the quality of GANs trained under several different processes on both imagery and tabular datasets. Moreover, we can also observe the number of epochs used by the different methods and their impact on the performance.

Figure 7. An example of when each alternative method might give the stop training signal to a given GAN.

We run three main experiments, which we describe briefly in Table 2. The first set of experiments is carried out on tabular datasets and involves the application of the initial, fixed, and AutoGAN methods. Given the nature of these datasets, a manual inspection is not possible as a human cannot visually inspect the quality of the generated tabular data. For the fixed method, we obtain the fixed number of stopping iterations to use to train the GAN from previous experiments in [19]. For the AutoGAN method, we are only able to apply four of the Oracles described in Section 3.3, namely CAS-real, CAS-syn, CDS, and FCD, which are the only ones applicable to tabular datasets. In the second and third sets of experiments, we use imagery datasets. We carried out extensive tests with a high number of imagery datasets in our second experiment, in which we applied the initial, fixed, and AutoGAN with six different oracles. Our third experiment was conducted on a smaller subset of the imagery datasets, for which we also tested the manual method besides all the other methods used for the second experiment. This third experiment could not have been run for all the imagery datasets due to the time required to run the manual method, which involves a human inspecting all the images generated by the GAN after each iteration to decide upon their quality. Being an extremely time-consuming task, we decided to run this method only for a subset of the imagery datasets. This allows us to include in our tests a frequently used method to assess the quality of the GAN while keeping the time to run our experiments manageable.

Table 2. Overall description of the three main experiences carried out. (* AutoGAN-IS was only used with imagery datasets that contain colored images).

Experiment	**Experiment #1**	**Experiment #2**	**Experiment #3**
Data Used	**Tabular Data**	**Imagery Data 1**	**Imagery Data 2**
Alternative Methods	Initial Fixed AutoGAN-CAS-real AutoGAN-CAS-syn AutoGAN-CDS AutoGAN-FCD	Initial Fixed AutoGAN-CAS-real AutoGAN-CAS-syn AutoGAN-CDS AutoGAN-FCD AutoGAN-FID AutoGAN-IS *	Initial Fixed Manual AutoGAN-CAS-real AutoGAN-CAS-syn AutoGAN-CDS AutoGAN-FCD AutoGAN-FID

4.2. Datasets

For our experiments, we considered four base tabular datasets and four base imagery datasets. For each of the base datasets, we created binary versions and obtained a total of 17 binary datasets. Figure 8 shows an overview of the eight base datasets and the binary versions created and used in this paper. The orange branches in this figure show the four base tabular datasets, upon which several binary datasets are built and are displayed in the red branches. Similarly, the blue branches show the four base imagery datasets, and their respective binary versions are represented in the purple branches. The dataset's names

and the two classes selected for the binary task are shown in the red and purple branches. Overall, we consider a total of 17 binary datasets: 7 tabular and 10 imagery datasets.

The class imbalance setting we are creating assumes that the two classes of the predictive tasks are not represented with a similar frequency, i.e., one of the classes is represented by a higher number of examples (majority or negative class) in the available data, while the other one is poorly represented (minority or positive class). To create different class imbalance tasks with different characteristics, we generated multiple variants with a varying number of majority and minority class examples. For each of the 17 binary datasets, we create 16 variants by changing the imbalance ratio (the imbalance ratio is defined as the ratio between the number of minority class samples and the number of majority class samples) and the sample size. We used the same procedure described in [19] to obtain these variants of the datasets. Namely, we select a random sample of majority class examples matching a desired majority class count. A set of minority class examples is then randomly selected from the dataset and added to the previous majority class examples to obtain the required imbalance ratio. We considered all combinations of four majority class counts (100, 200, 500, and 1000) and four imbalance ratios (0.1, 0.2, 0.3, and 0.4). A total of 272 (17 binary datasets $\times$16 variants) imbalanced datasets are generated and used in our experiments.

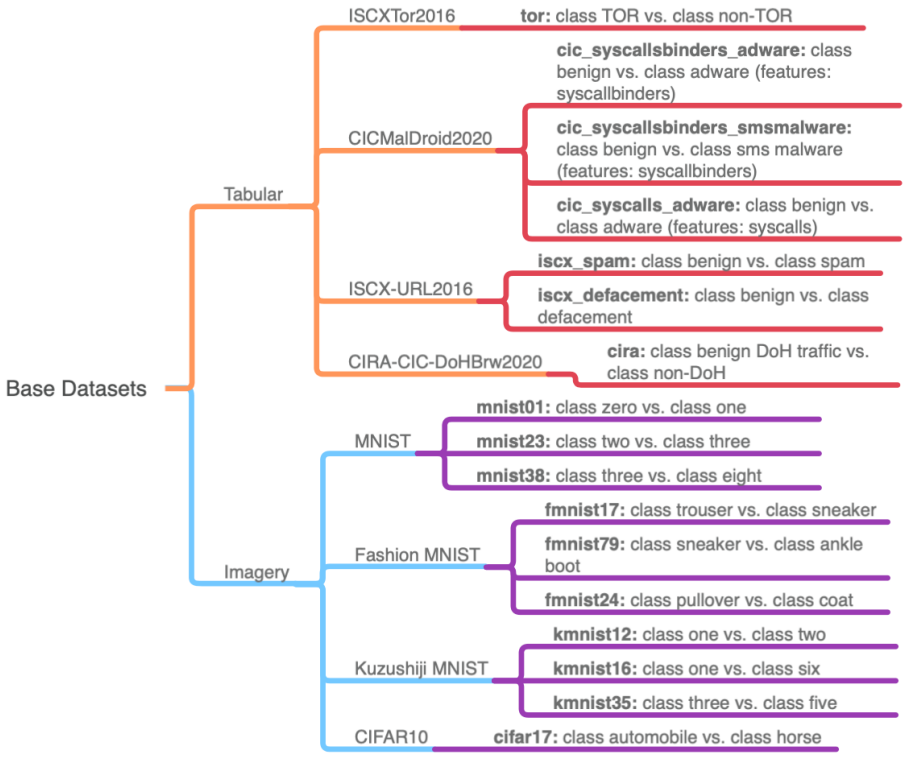

Figure 8. The base datasets (in blue and orange) used for creating 17 binary datasets (in red and purple). Dataset name and respective classes shown in the rightmost branch.

The following sections provide a more detailed description of the base tabular and imagery datasets as well as the 17 derived binary datasets that are used in our experiments.

4.2.1. Tabular Datasets

We used the same four base tabular datasets used in [19], namely ISCXTor2016 [49], CICMalDroid2020 [50], ISCX-URL2016 [51] and CIRA-CIC-DoHBrw2020 [52], which we transformed into seven different binary classification problems.

The ISCXTor2016 dataset [49] contains features extracted from network traffic using the ISCXFlowMeter. The binary classification dataset we named tor is derived from the target variable labels for TOR and non-TOR traffic.

The CICMalDroid2020 [50] dataset contains information about Android malware and includes the following five classes: benign, SMS malware, riskware, adware, and banking. The features are extracted from a total of 11,598 APK files and are grouped into two categories: (i) syscallbinders, which include the frequencies of system calls, binders, and composite behavior in a total of 470 features; and (ii) syscalls, which include the frequencies of system calls in a total of 139 features. Using the syscallsbinders features, we created two binary datasets: the cic-syscallsbinders-adware and the cic-syscallsbinders-smsadware. For the first one, only the instances with classes benign and adware were included, while for the second, only the instances with classes benign and sms malware were kept. We also created another binary dataset using the syscalls features. We named this dataset cic-syscalls-adware which includes the instances from adware and benign classes.

We obtained two binary datasets from the base dataset ISCX-URL2016 [51] (which contained five different classes): iscx-spam and iscx-defacement. The first one contains the cases from the classes benign and spam, and the second one contains the cases from the classes benign and defacement.

The last base tabular dataset considered is the CIRA-CIC-DoHBrw2020 [52], which has a target class with three labels: benign DoH traffic, malicious DoH traffic, and non-DoH traffic. We built a binary dataset that we named cira that includes only the benign DoH and non-DoH classes.

4.2.2. Imagery Datasets

In our experiments, we used the following four imagery base datasets: MNIST [53], Fashion-MNIST [54], Kuzushiji-MNIST [55], and CIFAR10 [56].

MNIST is a collection of 70,000 black-and-white images of 10 handwritten digits (i.e., 10 classes) with 28×28 pixels. We derived the following three binary datasets from the MNIST dataset: (1) mnist01: instances with classes zero and one are selected for this dataset; (2) mnist23: instances with digits (classes) two and three are selected for this dataset; and (3) mnist38: class three and class eight are chosen for this dataset. The different classes picked for these three binary datasets allow us to obtain a varying complexity of the predictive task, as we surmise that distinguishing between classes 0 and 1 is the simplest task while distinguishing between classes 3 and 8 can be observed as the most difficult task.

The Fashion-MNIST dataset contains 70,000 black-and-white images of 10 different clothing classes, each with 28×28 pixels. We derived three binary datasets from Fashion-MNIST by selecting different classes and using the same intuition used for the MNIST dataset to obtain tasks of different complexity. The following datasets were created: (1) fmnist17, using the classes trousers and sneakers; (2) fmnist79, using the classes sneaker and ankle boot; and (3) fmnist24, using the classes pullover and coat. We hypothesize that fmnist17 is the easiest of these tasks, fmnist79 has an intermediate complexity, and fmnist24 has the highest complexity of the three datasets.

Kuzushiji-MNIST dataset also contains 70,000 black and white images of 10 classes of handwritten Hiragana Japanese syllabary, each with 28×28 pixels. In this case, we also generated three different datasets by selecting two out of the 10 classes available in the original dataset. The following datasets were built: (1) kmnist12 with classes 1 and 2; (2) kmnist16 containing examples from classes 1 and 6; and (3) kmnist35 including cases from classes 3 and 5. We randomly selected the two classes used in each one of the cases.

Finally, we also derived one dataset from CIFAR10, a dataset containing 60,000 colored images of 10 classes of objects with $32 \times 32 \times 3$ pixels. We extracted automobile and horse classes to build our binary dataset, which we named cifar17.

4.3. Experimental Setting

Our three experiments, depicted in Figure 9, use 17 different base datasets, each with 16 variants with different imbalance ratios and sample sizes (see Section 4.2). For each experiment, we evaluated different alternative methods to provide a stopping signal to the GAN. We evaluate the method initial, which corresponds to using the original imbalanced

training set, and the other alternatives, which correspond to using the images generated by the GAN stopped with a certain stopping signal to oversample the training set, making the two classes balanced. For these methods, we use different oracles with the AutoGAN algorithm and also use the fixed (a set number of iterations to train the GAN) and manual (a human manual inspection of the GAN images).

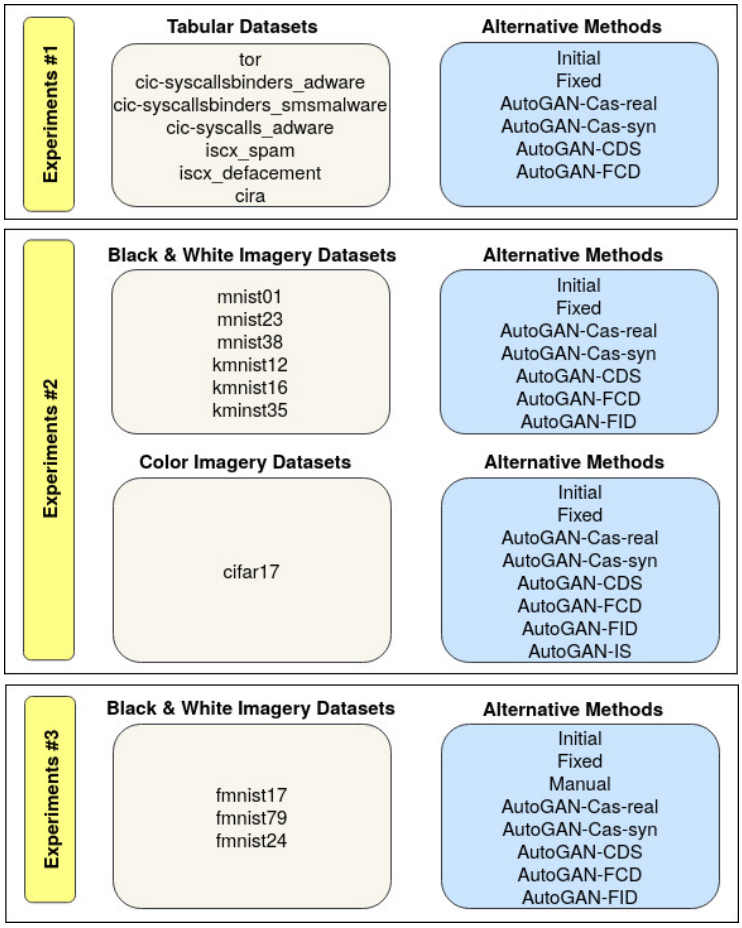

Figure 9. Detailed description of the three main experiments carried out.

One single GAN is trained until all methods give the stopping signal, as previously described and illustrated in Figure 7. We used a stratified five-fold cross-validation process. The F1-scores of both the minority and majority classes were recorded, irrespectively of the metric used internally by the AutoGAN algorithm. We report the average and standard deviation of the five-fold results. We also report the number of training iterations used for the GAN under each alternative method. This will allow us to observe the relationship between performance and the length of the required training. Finally, we analyse the Pearson correlation between the results of the different alternative methods for stopping the GAN training that we tested.

We selected a CGAN for our experiments. Two main architectures were considered: (i) fully connected hidden layers for both the discriminator and the generator; and (ii) convolutional layers for both the generator and discriminator. The output layer of the generator and the input layer of the discriminator were adapted to match the number of features in each dataset. Full details of the architectures are provided in the Appendix A.1.

The parameters used for AutoGAN with the oracles implemented are described in Appendix A.2. For the fixed alternative, we used the parameters used in [19] for the tabular datasets, while for the imagery datasets, we experimentally determined this value based on trial and error. For the manual alternative, we relied on the inputs from a human expert.

We selected a fully connected deep neural network for the classification task. The number of hidden layers and perceptrons used in each layer were adapted to each dataset. Full details on the network architecture are provided in Appendix A.3.

5. Results and Discussion

This section summarizes the main results and conclusion of the three experiments conducted.

5.1. Experiment #1: Tabular Datasets

We observed three different patterns in the results obtained from the tabular datasets. We present those patterns by showing the results of one dataset representative of the pattern. We will specifically discuss the results of the tor, iscx_defacement, and cira-based datasets. The detailed results of the remaining tabular datasets are provided in the Supplemental Material where additional figures (Figures S2–S9) and tables (Tables S1–S12, S33 and S34) are provided.

tor-based datasets

The results obtained on tor-based datasets exhibit a pattern that is also observed in the cic-syscallsbinders-adware, cic-syscallsbinders-smsmalware, and cic-syscalls-adware datasets. Figure 10 shows the average F1-scores for the minority class of the neural network trained on the different variants of the tor dataset (The corresponding results for the majority class can be observed in Figure S1 of the Supplemental Material). This figure demonstrates that all alternative methods applied to train the GAN produced better results than the initial setting (where no oversampling is applied). This demonstrates that, overall, all methods trained a GAN that generates images of comparable good quality. These results were aggregated by a majority class count and by imbalance ratio and are displayed in Figure 11 and Figure 12, respectively.

The following observations can be drawn from these results: (1) as an over-sampling technique, CGAN results in little or no deterioration of the majority class's F1-scores, whereas the minority class's F1-scores improve significantly; (2) as the difficulty factors (imbalance ratio and sample size) become more extreme, the classification task becomes more difficult, resulting in lower F1-scores; (3) if the initial F1 scores are lower, the amount of improvement obtained through CGAN over-sampling is greater; (4) the results of AutoGAN algorithm are relatively similar to that of the fixed setting in terms of F1-scores; and (5) AutoGAN algorithm achieves these results with a lower number of iterations for training the GAN when using three of the four tested oracles (FCD, CAS-real and CDS).

The stopping iterations displayed in the rightmost plot in Figures 11 and 12, show a clear difference in the oracle instances. For example, while the F1-scores of oracles CAS-real and CAS-syn in both minority and majority classes are similar, we observe that they stop at around 1000 and 2500 iterations, respectively. Furthermore, CAS-real achieves similar F1-scores to the fixed method while using a lower number of training iterations.

Figure 10. Box-plot of F1-scores of the minority class with different stopping methods for tor-based datasets.

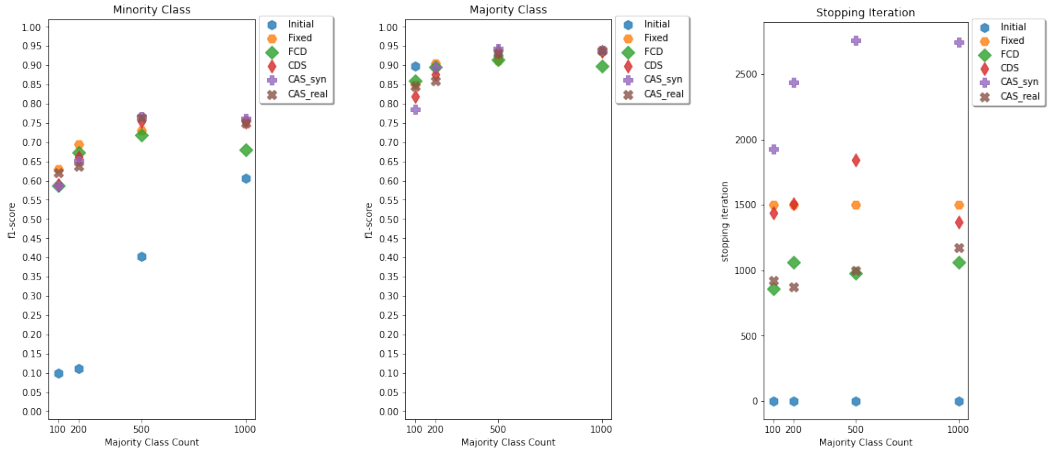

Figure 11. Average F1 results of tor-based datasets by majority class count for minority (**left**) and majority (**center**) classes, and average number of iterations until the training is stopped (**right**).

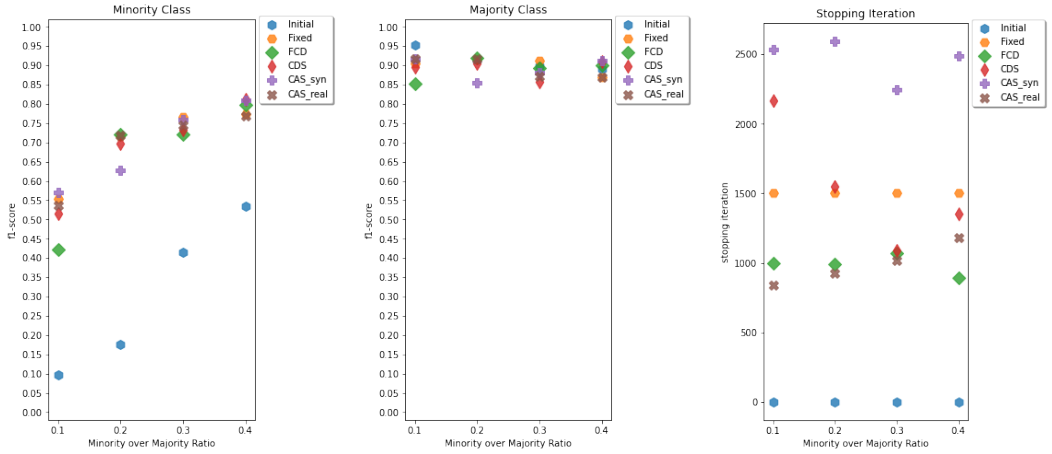

Figure 12. Average F1 results of tor-based datasets by imbalance ratio for minority (**left**) and majority (**center**) classes, and average number of iterations until the training is stopped (**right**).

iscx_defacement-based datasets

As the F1-score increases, the room for improvement shrinks, so that, after a certain point, fewer improvement opportunities are present. The experiments with iscx_defacement-based datasets, summarized in Figures 13 and 14, show no improvement when using any of the trained GANs when compared to the initial setting. The F1 results of the AutoGAN algorithm are also very close to the fixed setting. However, some differences are noticeable with regards to the number of iterations necessary for training the GAN. The fixed method used 1500 iterations, while, for instance, CDS consistently used fewer iterations.

cira-based datasets

The pattern observed in cira-based datasets was also observed in iscx-spam-based datasets. The three first observations identified in tor-based datasets are also present here (Figures 15 and 16). In addition, the results demonstrate that the performance outputs of the AutoGAN algorithm consistently surpass those of the fixed technique. Moreover, when compared against the fixed method, the results produced by AutoGAN with FCD and CDS Oracles are better while requiring fewer iterations. In fact, if we look at AutoGAN with FCD Oracle, the F1 performance achieved is among the best, despite the fact that it only took around 500 training iterations to stop the GAN (fixed used 1500 iterations).

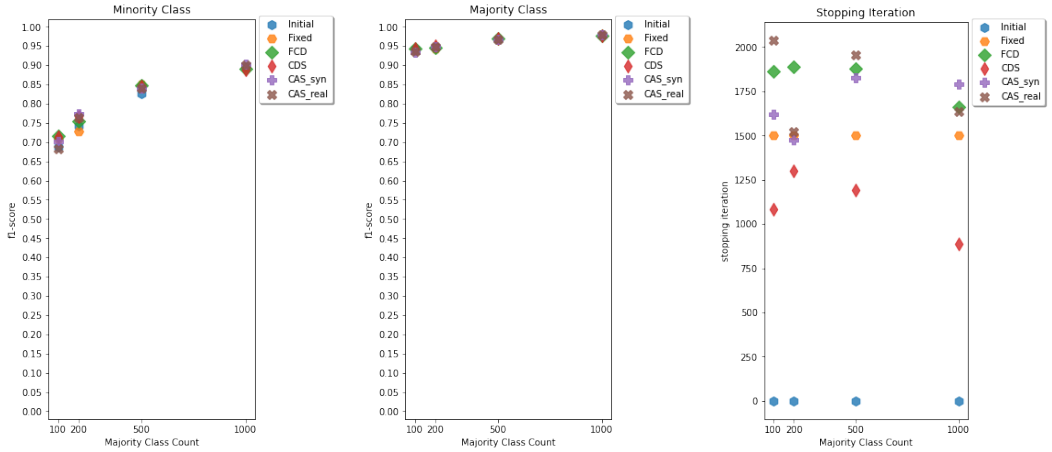

Figure 13. Average F1 results of iscx_defacement-based datasets by majority class count for minority (**left**) and majority (**center**) classes, and average number of iterations until the training is stopped (**right**).

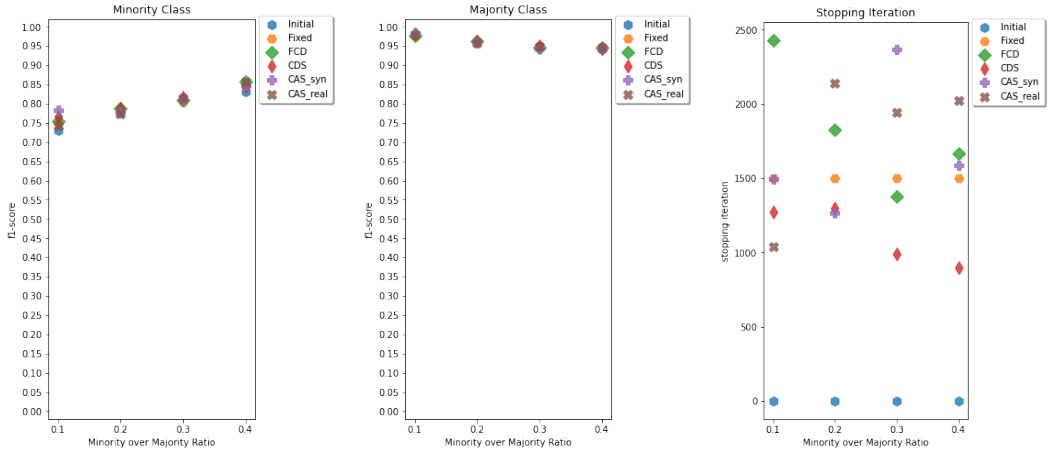

Figure 14. Average F1 results of iscx_defacement-based datasets by imbalance ratio for minority (**left**) and majority (**center**) classes, and average number of iterations until the training is stopped (**right**).

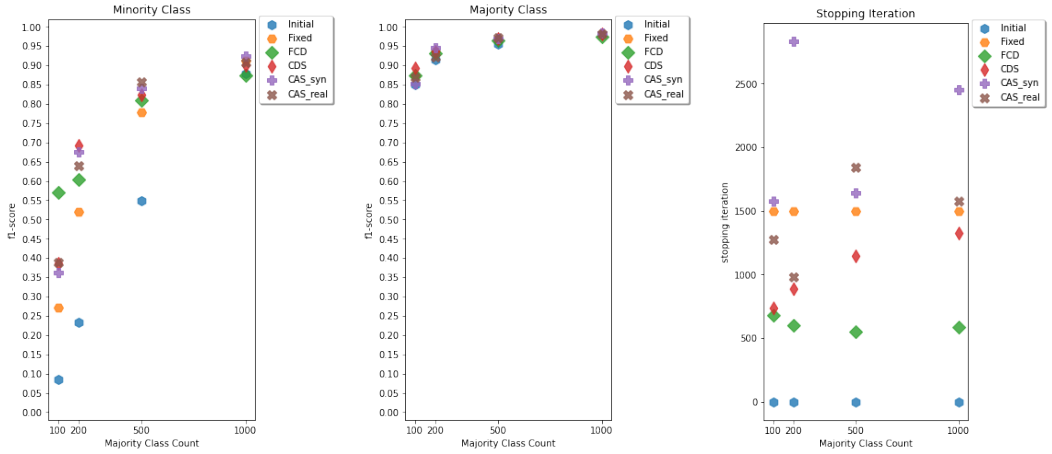

Figure 15. Average F1 results of cira-based datasets by majority class count for minority (**left**) and majority (**center**) classes, and average number of iterations until the training is stopped (**right**).

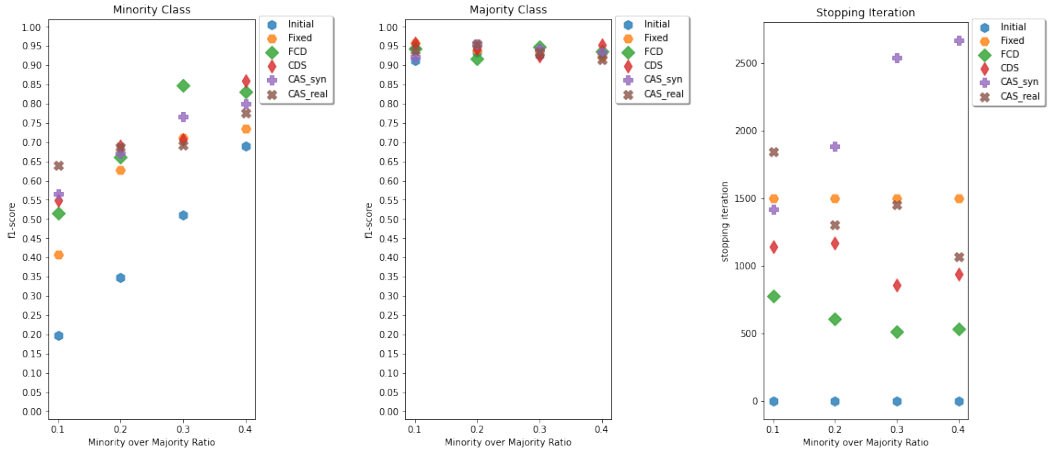

Figure 16. Average F1 results of cira-based datasets by imbalance ratio for minority (**left**) and majority (**center**) classes, and average number of iterations until the training is stopped (**right**).

5.2. Experiment #2: Imagery Datasets

5.2.1. Black-and-White Imagery Datasets

kmnist-based datasets

We selected kmnist-based datasets for showing the results of this experiment. Similar results were achieved with the mnist-based datasets. These additional results for mnist-based datsets are available in the Supplemental Material (Figures S12–S14 and Tables S21–S26). Figure 17 displays the results of all the variants of kmnist12 by imbalance ratio. As we can observe, all alternatives perform better than the initial method. Moreover, we also observe that all oracles tested in the AutoGAN algorithm perform similarly to the fixed alternative for both the minority and majority classes. However, there is a significant advantage to using AutoGAN when considering the number of training iterations. In fact, all the oracles tested use a much lower number of iterations than the fixed approach, although they achieve the same performance. The fixed approach was set to use more than 7000 iterations. The automatic method, on the other hand, uses between 1000 and 4000 iterations with the FCD and FID oracles. All the remaining oracles tested use a number of iterations between these two values. Further figures for Kmnit-based datasets are available in the Supplemental Material (Figures S10 and S11). Further detailed results for these datasets are available in Tables S27–S32.

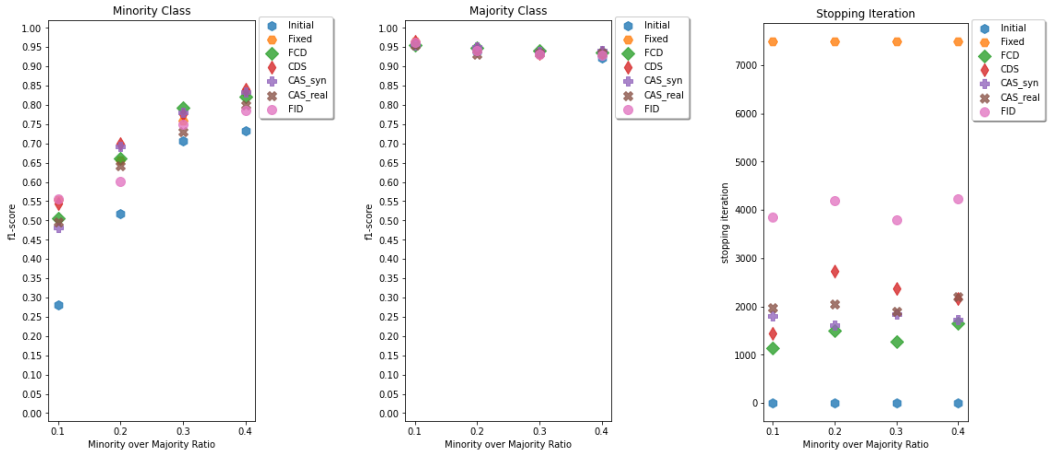

Figure 17. Average F1 results of kmnist12-based datasets by imbalance ratio for minority (**left**) and majority (**center**) classes, and average number of iterations until the training is stopped (**right**).

5.2.2. Color Imagery Datasets

cifar17 datasets

The results of the experiments on the 16 variants of the binary cifar17 dataset are shown in Figure 18. We do not observe a considerable improvement on the oversampling technique. However, in the most extreme case where the number of majority class count is 100, we observe improvements in the minority class with little effect on the majority class. In this case, the AutoGAN approach yields a similar performance to the fixed number of iterations, while, on average, requiring a much lower number of iterations for training (see Figure 18 on the right). Note that unlike black-and-white imagery datasets, we can also apply the IS oracle to this dataset. This shows that all the oracles tested use less training to achieve the same results. In effect, FID and FCD use approximately 5000 to 10,000 iterations, while the fixed method requires 20,000. These results also demonstrate the adaptability of our AutoGAN, which can increase the required training iterations dynamically without any human intervention. This is especially noticeable for CDS and IS oracles, which required a greater number of training iterations in the cases of the 100 and 200 majority class count. This number was not considered necessary for the other cases and was found automatically. Tables with the detailed results for cifar-based datasets are available in the Supplemental Material (Tables S19 and S20).

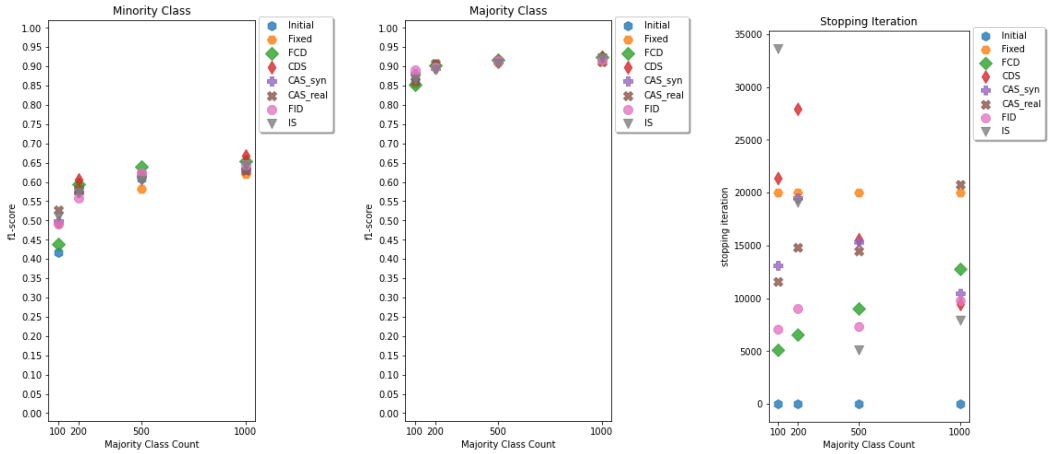

Figure 18. Average F1 results of cifar17-based datasets by imbalance ratio for minority (**left**) and majority (**center**) classes, and average number of iterations until the training is stopped (**right**).

5.3. Experiment #3: Imagery Datasets with Human Inspection

Fashion-MNIST-based datasets

The fmnist datasets was used to build three binary class problems: fmnist17, fmnist79, and fmnist24. We select the classes to obtain different difficulty levels for a human to visually distinguish between the two classes. The two classes are clearly distinguishable visually for the fmnist17 dataset, the task is more challenging for the fmnist79 dataset, and the fmnist24 dataset presents the most challenging task. The overall F1 results and training iterations required for these datasets are shown in Figure 19. Experiments indicate that the classification challenge for a neural network classifier is equivalent in difficulty to human visual classification. This can be observed on the leftmost plots of Figure 19, where the easiest task (top left plot) shows very high results for all methods, the intermediate task (middle left plot) shows some degradation of the performance across all methods, and the most difficult task (bottom left plot) displays much lower F1-scores for all methods. This shows clearly that, as the classification task becomes more difficult, the F1-score of the minority class deteriorates. GAN over-sampling offers the most benefits in the most difficult case, where there is significant room for improvement.

The following key observations can be drawn from these results: (1) in almost all cases, very little change in the performance of the majority class is observed; (2) AutoGAN

results are very similar to the values obtained with the human manual inspection (human) and the fixed number of iterations (fixed); (3) when comparing the manual inspection with the fixed iterations, we observe that similar performance results are achieved with fewer iterations for the manual method; (4) AutoGAN algorithm, with any of the oracles tested, achieves the same performance results as manual method with a lower number of training iterations. This is in fact an outstanding result: AutoGAN is able to stop the training to obtain the same performance that we could obtain with a human inspection of the generated images, but it achieves this with less training. Additional detailed results are available for fmnist-based datasets inthe Supplemental Material (Tables S13–S18).

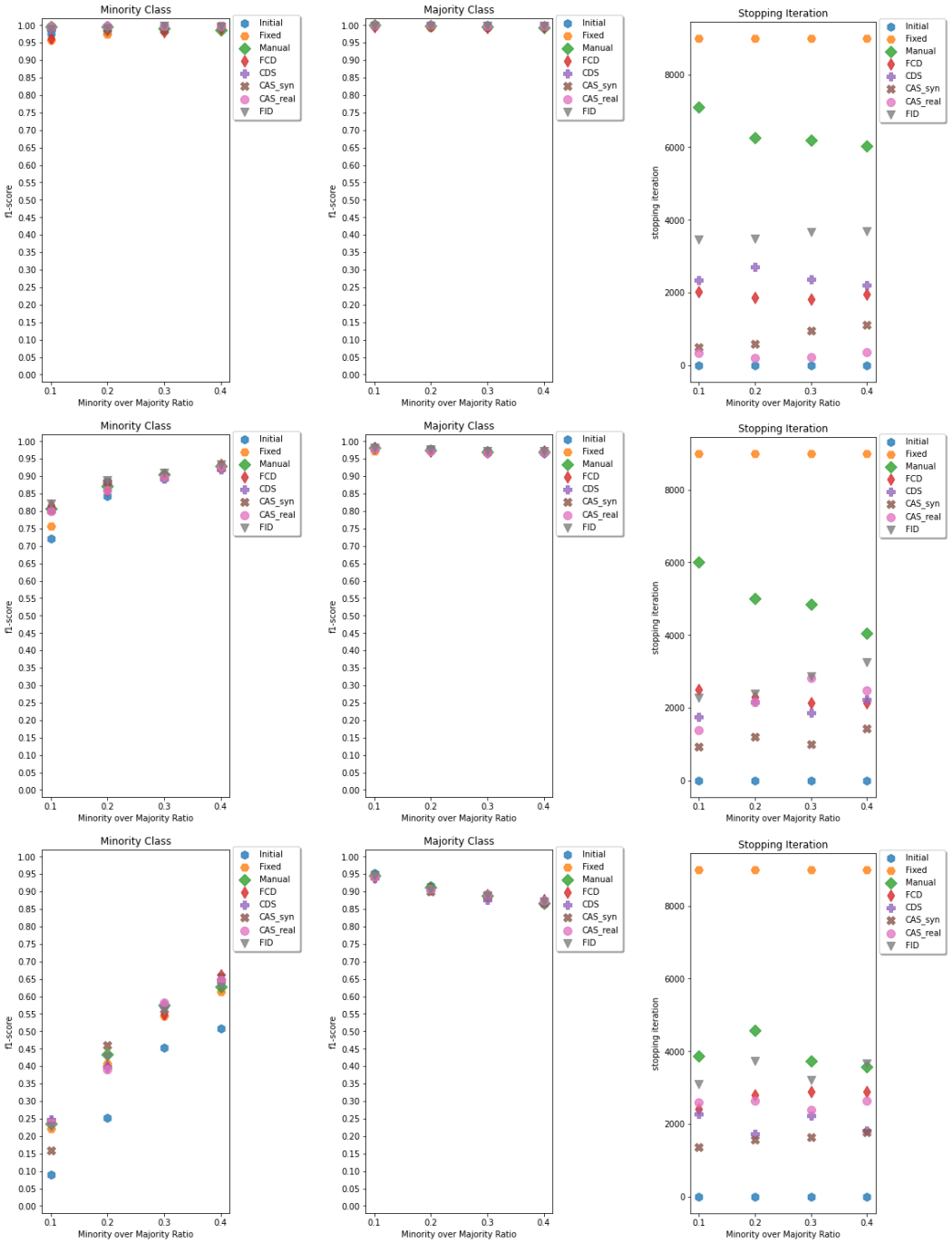

Figure 19. Average F1 results of fmnist17 (**top**), fmnist79 (**middle**), and fmnist24 (**bottom**)-based datasets by imbalance ratio for minority (**left**) and majority (**center**) classes, and average number of iterations until the training is stopped (**right**).

5.4. Correlation Analysis of the Methods Used for Stopping the GAN Training

We further studied the correlation between the different methods for stopping a GAN's training. We focused on the F1 results of the minority class. Our goal is to observe whether the results of AutoGAN with different oracles are similar to the other methods, including: initial, fixed, and manual. We computed the Pearson correlation between the F1 scores of four groups of experimental results. We considered the three main experiments carried out and formed four groups as follows: Group 1: includes all tabular datasets from experiment #1; Group 2: includes black-and-white imagery datasets from experiment #2 and the fmnist-based datasets; Group 3: includes colored imagery datasets used in experiment #2; and Group 4: includes only the fmnist-based datasets as they were tested with the manual option (experiment #3). Figures 20–23 show the correlation results and the corresponding dendrogram of the four groups described, respectively.

In Figures 20, 21, and 23, we observe that the initial method exhibits the lowest correlation with the remaining methods. However, for group 3, which contains the color imagery datasets, a different trend is shown, as displayed in Figure 22. We hypothesize that this difference can be explained by the over-sampling results (Figure 18); in the experiments where the initial results are sufficiently poor and the room for improvement is substantial, the GAN-oversampling demonstrates a significant improvement. In contrast, with cifar17-based datasets (group 3), GAN-oversampling yields negligible or no improvement. This lack of improvement from the initial results suggests that this might be linked to a higher correlation between the initial approach and the alternative methods.

When observing Figure 20, it is clear that all of the methods are clearly more separated from each other than from the initial. On the contrary, the figure highlights that the fixed exhibits more correlation with the other methods; namely, it is closer to the cluster containing CDS and CAS-syn.

Figure 21 shows the highest separation between initial and the remaining methods. We also observe that the fixed method is closer to FCD and FID for these experiments.

The correlation results of the group 3 (color imagery datasets) are the most surprising ones (Figure 22). Besides the fact that the initial method is not clearly separated from the remaining methods, we also observe a strong correlation between the IS and fixed. We must highlight that we only used IS in one color imagery base dataset (cifar17). Thus, these results might be biased towards the particular performance of these base datasets.

Finally, the correlation results observed in Figure 23 confirm that the fixed method is close to FID and FCD oracles. Moreover, these results highlight that the manual is very correlated with CAS-syn and also with CDS and CAS-real. This confirms the advantages of replacing a human's manual inspection of images with our AutoGAN algorithm, which can achieve results highly correlated with several of the oracles implemented and tested. Moreover, this is achieved with a lower number of training iterations, which brings benefits in terms of computational time and memory.

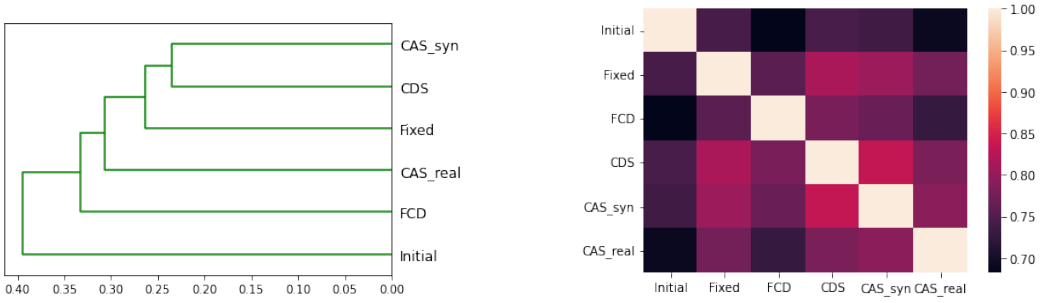

Figure 20. The Pearson correlation heatmap and dendrogram plots of the minority class of the experiments on all tabular datasets.

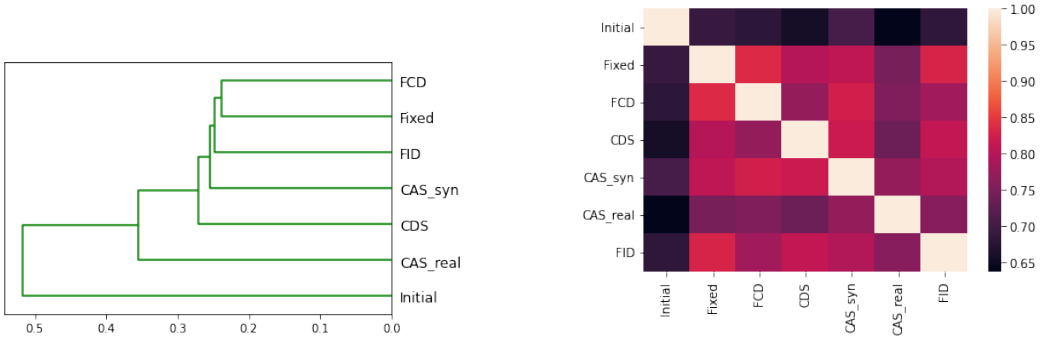

Figure 21. The Pearson correlation heatmap and dendrogram plots of the minority class of the experiments on all black-and-white imagery datasets (mnist-based, fmnist-based, and kmnist-based).

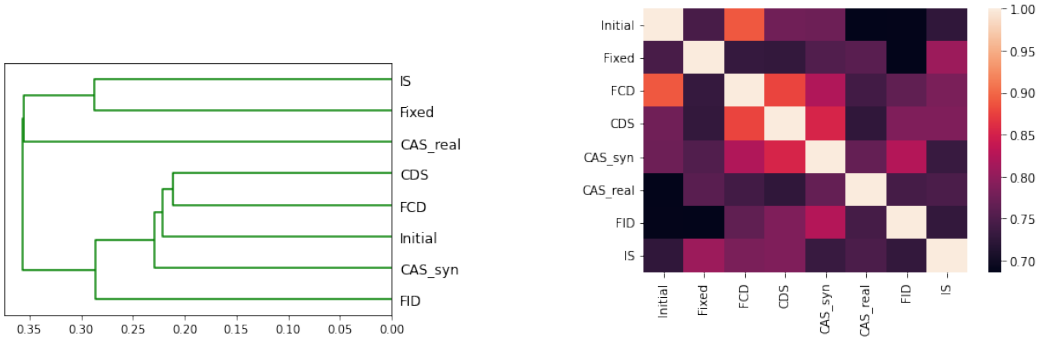

Figure 22. The Pearson correlation heatmap and dendrogram plots of the minority class of the experiments on color imagery datasets (cifar17-based).

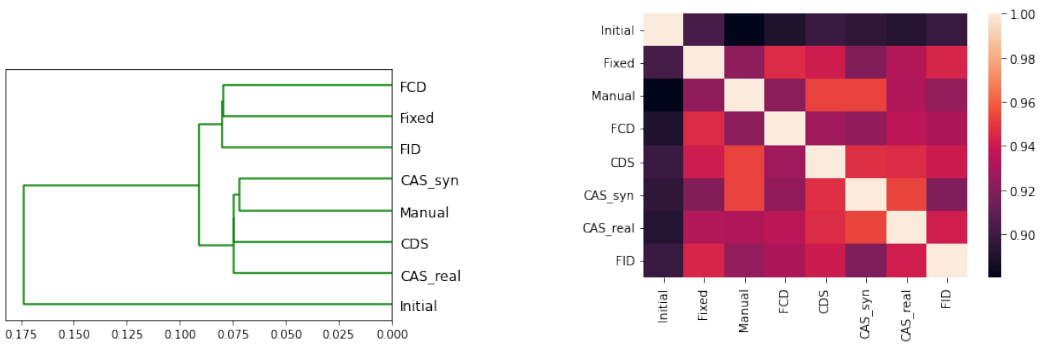

Figure 23. The Pearson correlation heatmap and dendrogram plots of the minority class of the experiments on fmnist-based datasets.

6. Conclusions

In this paper, we address the issue of when to stop training a GAN by combining quantitative measurements into an algorithm named AutoGAN. Our proposed AutoGAN is an "human-out-of-the-loop" solution, as it automatically decides when to stop training a GAN and requires minimal human monitoring or intervention during the GAN training process. The extensive set of experiments carried out on both tabular and imagery datasets show that AutoGAN achieves similar or superior results than all the alternative methods. Moreover, this solution provides gains in terms of the number of training iterations required, as it allows one to achieve a point of the GAN with fewer training iterations. Another key finding concerns the comparison with manual human inspection. In effect, we observe that AutoGAN consistently decides to stop the training at lower stages while ensuring that the data generated are of high quality.

Regarding our conclusions on tabular datasets, we must highlight that overall, the majority class performance is not affected while the minority class performance increases

or is comparable. The performance results of the AutoGAN algorithm are relatively similar to those of running a fixed number of training iterations, but the AutoGAN algorithm achieves these results with a much lower number of training iterations.

On the imagery data, similar conclusions are reached, despite the fact that more oracles and alternatives for stopping the GAN's training process were tried. Overall, the Oracles tested with AutoGAN provided competitive results against the option of training the GAN for a fixed number of iterations but also against the option of training a GAN using manual human inspection. These results are remarkable, demonstrating that, in fact, the human inspection, when possible, is a time-consuming, subjective process that will require more training iterations to achieve the desired result.

Finally, our study of the Pearson correlation between the different methods demonstrates that, overall, the initial method has the lowest correlation with the remaining methods. The correlation between the oracles and the other methods varies greatly with the datasets tested. Thus, no global conclusion regarding the other method's correlation is provided.

In terms of future research directions, our proposed framework opens the way to shift several GAN applications from the image domain to the tabular data domain, as no supervision is required to train the GAN. An interesting future research direction could explore, for instance, the transferability of image-to-image translation to tabular-to-tabular translation, or image de-noising to tabular data de-noising. AutoGAN can assist with these new tasks as it allows any GAN to be trained for tabular data without human intervention. Another relevant research aspect could involve using AutoGAN to address the problem of mode collapse and vanishing gradients. In fact, our early experiments demonstrate that the scores obtained by oracle instances for GAN suffering from mode collapse show heavy oscillations; on the contrary, we do not observe such behaviour when the mode collapse problem is not present. Facing the potential problem of detecting overfitting in the GAN's generator can also be investigated in the future. We hypothesize that specific oracle instances and/or modifications to the AutoGAN algorithm would be necessary to achieve this goal.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/math11040977/s1.

Author Contributions: Conceptualization, E.N., P.B. and G.-V.J.; Methodology, E.N., P.B. and G.-V.J.; Software, E.N.; Formal analysis, E.N.; Visualization, E.N.; Writing—original draft, E.N. and P.B.; Data curation, E.N.; Writing—review and editing, E.N., P.B. and G.-V.J.; Supervision, P.B. and G.-V.J. All authors have read and agreed to the published version of the manuscript.

Funding: The work of E.N., P.B. and G.-V.J. was undertaken, in part, thanks to funding from Discovery Grants from NSERC.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The base datasets we use are publicly available. Our code includes a function to generate the different imbalanced versions we created. By running this function on the base datasets, all our versions will be obtained.

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

Appendix A. Implementation Details

Appendix A.1. Details of the CGAN

The general architecture of GAN for all tabular data and black and white imagery data remains similar throughout all experiments. We used a fully connected conditional GAN architecture based on the code available on https://github.com/eriklindernoren/Keras-GAN/tree/master/cgan (accessed 8 February 2023). The only difference between the GANs is the output layer of the generator and the input layer of the discriminator,

which match the dimension of the specific dataset being used. Tables A1 and A2 show the architectures of the generator and the discriminator for a dataset with 784 features, in this particular case, the fmnist-based datasets.

For cifar10-based datasets, a different architecture for both the generator and the discriminator is used. Tables A3 and A4 show the architecture details of the generator and the discriminator, respectively.

Table A1. The architecture of the generator used for the fmnist-based datasets.

Explanation	Layer	Output
Input1: noise	1	100 Neurons
Input2: class label	1	1 Neuron
transforming class label into 100	2	100 Neurons
multiply transformed class label with input noise	3	100 Neurons
Dense	4	100 Neurons
Dense	5	1024 Neurons
Dense	6	784 Neurons

Table A2. The architecture of the discriminator used for the fmnist-based datasets.

Explanation	Layer	Output
Input	1	784 Neurons
Dense	2	512 Neurons
Dense	3	512 Neurons
Dense	4	512 Neurons
Dense	5	1 Neuron

Table A3. The architecture of the generator used for the cifar10-based datasets.

Explanation	Layer	Output
Input1: noise	1	100 Neurons
Input2: class label	1	1 Neuron
Transforming class label into 100	2	100 Neurons
Multiply transformed class label with input noise	3	100 Neurons
Dense	4	4096 Neurons
Reshape to (4, 4, 256)	5	(4, 4, 256)
Conv2DTranspose: filters: 128; kernel shape: (4, 4)	6	(8, 8, 128)
Conv2DTranspose: filters: 128; kernel shape: (4, 4)	7	(16, 16, 128)
Conv2DTranspose: filters: 128; kernel shape: (4, 4)	8	(32, 32, 128)
Conv2D: filters: 3; kernel shape: (3, 3)	9	(32, 32, 3)
Reshape to 3072	10	3072 Neurons

Table A4. The architecture of the discriminator used for the cifar10-based datasets.

Explanation	Layer	Output
Input1: the image	1	3072 Neurons
Input2: class label	1	1 Neuron
Transforming class label into 100	2	100 Neurons
Multiply transformed class label with input noise	3	100 Neurons
Reshape to (32, 32, 3)	4	(32, 32, 3)
Conv2D: filters: 64; kernel shape: (3, 3)	5	(32, 32, 64)
Conv2D: filters: 128; kernel shape: (3, 3)	6	(16, 16, 128)
Conv2D: filters: 128; kernel shape: (3, 3)	7	(8, 8, 128)
Conv2D: filters: 256; kernel shape: (3, 3)	8	(4, 4, 256)
Flatten	9	4096
Dense	10	1 Neuron

Appendix A.2. Parameters Used for AutoGAN Algorithm

The hyper-parameters of AutoGAN Algorithm that were used across all the experiments are the following:

- The number of accepted failed attempts: 15;
- Iterations unit: 100;
- The number of generated samples per class for calculating the scores = 500.

The parameters of each oracle that we implemented and tested with the AutoGAN algorithm are as follows:

- Oracle CAS_syn:
  - The number of hidden layers for the classifier = 2;
  - The number of perceptrons for the classifier = 100;
  - The number of training epochs for the classifier = 100;
  - Optimizer: adam;
  - Batch size: 32.
- Oracle CAS_real:
  - The number of hidden layers for the classifier = 2;
  - The number of perceptrons for the classifier = 100;
  - The number of training epochs for the classifier = 100;
  - Optimizer: adam;
  - Batch size: 32.
- Oracle CDS:
  - The number of hidden layers for the classifier of CDS = 2;
  - The number of perceptrons for the classifier of CDS = 100;
  - The number of training epochs for the classifier of CDS = 100;
  - Optimizer: adam;
  - Batch size: 32.
- Oracle FCD:
  - The autoencoder consists of 6 layers of sizes: 784, 784×2, 784, 784/2 (the bottleneck), 784×2, 784;
  - Optimizer = 'adam';
  - Loss = 'mse';
  - The number of training epochs for the autoencoder = 200.

Appendix A.3. Classifier Details

The classifier used in our experiments is a fully connected deep neural network. For each base dataset, different numbers of the hidden layer and different numbers of perceptrons are used according to the difficulty of the problem. Moreover, the input layer of each neural network is altered to match the feature number of the datasets. The activation functions used are rectified linear units.

- Tor-based datasets: one hidden layer of 10 perceptrons;
- cic_syscallsbinders_adware-based datasets: two hidden layers of 20 perceptrons;
- cic_syscallsbinders_smsmalware-based datasets: two hidden layers of 20 perceptrons;
- cic_syscalls_adware-based datasets: two hidden layers of 100 perceptrons;
- iscx_spam-based datasets: two hidden layers of 20 perceptrons;
- iscx_defacement: two hidden layers of 100 perceptrons;
- cira-based datasets: one hidden layer of 10 perceptrons;
- mnist-based, fashion-mnist-based, Kuzushiji-mnist-based, and cifar10-based datasets: one hidden layers of five perceptrons.

References

1. Borji, A. Pros and Cons of GAN Evaluation Measures. *arXiv* **2018**, arXiv:1802.03446. [CrossRef]
2. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Networks. *arXiv* **2014**, arXiv:1406.2661. [CrossRef]
3. Brock, A.; Donahue, J.; Simonyan, K. Large Scale GAN Training for High Fidelity Natural Image Synthesis. *arXiv* **2018**, arXiv:1809.11096. [CrossRef]
4. Karras, T.; Aila, T.; Laine, S.; Lehtinen, J. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *arXiv* **2017**, arXiv:1710.10196. [CrossRef]
5. Karras, T.; Laine, S.; Aila, T. A Style-Based Generator Architecture for Generative Adversarial Networks. *arXiv* **2018**, arXiv:1812.04948. [CrossRef]
6. Karras, T.; Aittala, M.; Laine, S.; Härkönen, E.; Hellsten, J.; Lehtinen, J.; Aila, T. Alias-Free Generative Adversarial Networks. *arXiv* **2021**, arXiv:2106.12423. [CrossRef]
7. Karras, T.; Laine, S.; Aittala, M.; Hellsten, J.; Lehtinen, J.; Aila, T. Analyzing and Improving the Image Quality of StyleGAN. *arXiv* **2019**, arXiv:1912.04958. [CrossRef]
8. Isola, P.; Zhu, J.Y.; Zhou, T.; Efros, A.A. Image-to-Image Translation with Conditional Adversarial Networks. *arXiv* **2016**, arXiv:1611.07004. [CrossRef]
9. Zhu, J.Y.; Park, T.; Isola, P.; Efros, A.A. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *arXiv* **2017**, arXiv:1703.10593. [CrossRef]
10. Emami, H.; Aliabadi, M.M.; Dong, M.; Chinnam, R.B. SPA-GAN: Spatial Attention GAN for Image-to-Image Translation. *IEEE Trans. Multimed.* **2021**, *23*, 391–401. [CrossRef]
11. Ledig, C.; Theis, L.; Huszar, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; et al. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *arXiv* **2016**, arXiv:1609.04802. [CrossRef]
12. Tulyakov, S.; Liu, M.Y.; Yang, X.; Kautz, J. MoCoGAN: Decomposing Motion and Content for Video Generation. *arXiv* **2017**, arXiv:1707.04993. [CrossRef]
13. Munoz, A.; Zolfaghari, M.; Argus, M.; Brox, T. Temporal Shift GAN for Large Scale Video Generation. *arXiv* **2020**, arXiv:2004.01823. [CrossRef]
14. 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. [CrossRef]
15. Bojchevski, A.; Shchur, O.; Zügner, D.; Günnemann, S. NetGAN: Generating Graphs via Random Walks. *arXiv* **2018**, arXiv:1803.00816. [CrossRef]
16. Guo, J.; Lu, S.; Cai, H.; Zhang, W.; Yu, Y.; Wang, J. Long Text Generation via Adversarial Training with Leaked Information. *arXiv* **2017**, arXiv:1709.08624. [CrossRef]
17. Park, N.; Mohammadi, M.; Gorde, K.; Jajodia, S.; Park, H.; Kim, Y. Data synthesis based on generative adversarial networks. *Proc. VLDB Endow.* **2018**, *11*, 1071–1083. [CrossRef]
18. Nazari, e.; Branco, P. On Oversampling via Generative Adversarial Networks under Different Data Difficult Factors. In Proceedings of the International Workshop on Learning with Imbalanced Domains: Theory and Applications, Online, 17 September 2021; PMLR 2021; pp. 76–89.
19. Nazari, E.; Branco, P.; Jourdan, G.V. Using CGAN to Deal with Class Imbalance and Small Sample Size in Cybersecurity Problems. In Proceedings of the 2021 18th International Conference on Privacy, Security and Trust (PST), Auckland, New Zealand, 13–15 December 2021; IEEE: New York, NY, USA, 2021; pp. 1–10. [CrossRef]
20. Pennisi, M.; Palazzo, S.; Spampinato, C. Self-improving classification performance through GAN distillation. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW), Montreal, BC, Canada, 11–17 October 2021; pp. 1640–1648. [CrossRef]
21. Chaudhari, P.; Agrawal, H.; Kotecha, K. Data augmentation using MG-GAN for improved cancer classification on gene expression data. *Soft Comput.* **2020**, *24*, 11381–11391. [CrossRef]
22. Luo, Y.; Cai, X.; Zhang, Y.; Xu, J.; Yuan, X. Multivariate Time Series Imputation with Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Montreal, QC, Canada, 2018; Volume 31.
23. Gupta, A.; Johnson, J.; Fei-Fei, L.; Savarese, S.; Alahi, A. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. *arXiv* **2018**, arXiv:1803.10892. [CrossRef].
24. Saxena, D.; Cao, J. D-GAN: Deep Generative Adversarial Nets for Spatio-Temporal Prediction. *arXiv* **2019**, arXiv:1907.08556. [CrossRef].
25. Mescheder, L.; Geiger, A.; Nowozin, S. Which Training Methods for GANs do actually Converge? *arXiv* **2018**, arXiv:1801.04406. [CrossRef].
26. Daskalakis, C.; Ilyas, A.; Syrgkanis, V.; Zeng, H. Training GANs with Optimism. *arXiv* **2017**, arXiv:1711.00141. [CrossRef].
27. Goodfellow, I. NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv* **2017**, arXiv:1701.00160. [CrossRef].

28. Zhou, S.; Gordon, M.L.; Krishna, R.; Narcomey, A.; Fei-Fei, L.; Bernstein, M.S. HYPE: A Benchmark for Human eYe Perceptual Evaluation of Generative Models. *arXiv* **2019**, arXiv:1904.01121. [CrossRef].
29. Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X.; Chen, X. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems*; Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R., Eds.; Curran Associates, Inc.: Barcelona, Spain, 2016; Volume 29.
30. Borji, A. Pros and Cons of GAN Evaluation Measures: New Developments. *arXiv* **2021**, arXiv:2103.09396. [CrossRef].
31. Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X. Improved Techniques for Training GANs. *arXiv* **2016**, arXiv:1606.03498. [CrossRef].
32. 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* **2017**, arXiv:1706.08500. [CrossRef].
33. Ravuri, S.; Vinyals, O. Classification Accuracy Score for Conditional Generative Models. *arXiv* **2019**, arXiv:1905.10887. [CrossRef].
34. 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.
35. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein GAN. *arXiv* **2017**, arXiv:1701.07875. [CrossRef].
36. Papadimitriou, C.H. *Computational Complexity*; Addison-Wesley: Reading, MA, USA, 1994.
37. Mirza, M.; Osindero, S. Conditional Generative Adversarial Nets. *arXiv* **2014**, arXiv:1411.1784. [CrossRef].
38. Kullback, S. *Information Theory and Statistics*; Courier Corporation: New York, NY, USA, 1997.
39. Ramdas, A.; Garcia, N.; Cuturi, M. On Wasserstein Two Sample Testing and Related Families of Nonparametric Tests. *arXiv* **2015**, arXiv:1509.02237. [CrossRef].
40. Barratt, S.; Sharma, R. A Note on the Inception Score. *arXiv* **2018**, arXiv:1801.01973. [CrossRef].
41. Obukhov, A.; Krasnyanskiy, M. Quality Assessment Method for GAN Based on Modified Metrics Inception Score and Fréchet Inception Distance. In *Software Engineering Perspectives in Intelligent Systems*; Silhavy, R., Silhavy, P., Prokopova, Z., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 102–114.
42. Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; Le, Q.V. Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–17 June 2019; pp. 2820–2828.
43. Fu, Y.; Chen, W.; Wang, H.; Li, H.; Lin, Y.; Wang, Z. Autogan-distiller: Searching to compress generative adversarial networks. *arXiv* **2020**, arXiv:2006.08198.
44. Wang, H.; Huan, J. Agan: Towards automated design of generative adversarial networks. *arXiv* **2019**, arXiv:1906.11080.
45. Gong, X.; Chang, S.; Jiang, Y.; Wang, Z. Autogan: Neural architecture search for generative adversarial networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27–28 October 2019; pp. 3224–3234.
46. Morozov, S.; Voynov, A.; Babenko, A. On Self-Supervised Image Representations for {GAN} Evaluation. In Proceedings of the International Conference on Learning Representations, Virtual Event, 3–7 May 2021.
47. Isola, P.; Zhu, J.Y.; Zhou, T.; Efros, A.A. Image-To-Image Translation with Conditional Adversarial Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
48. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A.C. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Long Beach, CA, USA, 2017; Volume 30.
49. Habibi Lashkari, A.; Draper Gil, G.; Mamun, M.S.I.; Ghorbani, A.A. Characterization of Tor Traffic using Time based Features. In Proceedings of the 3rd International Conference on Information Systems Security and Privacy—ICISSP, Porto, Portugal, 19–21 February 2017; INSTICC; SciTePress: Setubal, Portugal, 2017; pp. 253–262. [CrossRef]
50. Mahdavifar, S.; Abdul Kadir, A.F.; Fatemi, R.; Alhadidi, D.; Ghorbani, A.A. Dynamic Android Malware Category Classification using Semi-Supervised Deep Learning. In Proceedings of the 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), Online Event, 17–22 August 2020; pp. 515–522. . [CrossRef]
51. Mamun, M.S.I.; Rathore, M.A.; Lashkari, A.H.; Stakhanova, N.; Ghorbani, A.A. Detecting Malicious URLs Using Lexical Analysis. In *Network and System Security*; Chen, J., Piuri, V., Su, C., Yung, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 467–482.
52. MontazeriShatoori, M.; Davidson, L.; Kaur, G.; Habibi Lashkari, A. Detection of DoH Tunnels using Time-series Classification of Encrypted Traffic. In Proceedings of the 2020 IEEE DASC/PiCom/CBDCom/CyberSciTech, Online Event, 17–22 August 2020; pp. 63–70. [CrossRef]
53. Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Process. Mag.* **2012**, *29*, 141–142. [CrossRef]
54. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv* **2017**, arXiv:1708.07747. [CrossRef].

55. Clanuwat, T.; Bober-Irizar, M.; Kitamoto, A.; Lamb, A.; Yamamoto, K.; Ha, D. Deep learning for classical japanese literature. *arXiv* **2018**, arXiv:1812.01718.
56. Krizhevsky, A. *Learning Multiple Layers of Features from Tiny Images*; Technical Report; Computer Science-University of Toronto: Toronto, ON, Canada, 2009.