The Astrophysical Journal, 935:23 (14pp), 2022 August 10 https://doi.org/10.3847/1538-4357/ac6f5a

Improving Astronomical Time-series Classification via Data Augmentation with Generative Adversarial Networks

Germán García-Jara[1], Pavlos Protopapas[2], and Pablo A. Estévez[1,3]
[1] Dept. of Electrical Engineering, Universidad de Chile, Chile; ggarciajara@gmail.com
[2] Institute for Applied Computational Science, Harvard University, USA
[3] Millennium Institute of Astrophysics, Chile

Received 2022 April 5; accepted 2022 May 11; published 2022 August 9

Abstract

Due to the latest advances in technology, telescopes with significant sky coverage will produce millions of astronomical alerts per night that must be classified both rapidly and automatically. Currently, classification consists of supervised machine-learning algorithms whose performance is limited by the number of existing annotations of astronomical objects and their highly imbalanced class distributions. In this work, we propose a data augmentation methodology based on generative adversarial networks (GANs) to generate a variety of synthetic light curves from variable stars. Our novel contributions, consisting of a resampling technique and an evaluation metric, can assess the quality of generative models in unbalanced data sets and identify GAN-overfitting cases that the Fréchet inception distance does not reveal. We applied our proposed model to two data sets taken from the Catalina and Zwicky Transient Facility surveys. The classification accuracy of variable stars is improved significantly when training with synthetic data and testing with real data with respect to the case of using only real data.

Unified Astronomy Thesaurus concepts: Variable stars (1761); RR Lyrae variable stars (1410); Cepheid variable stars (218); Delta Scuti variable stars (370); Long period variable stars (935); Eclipsing binary stars (444); Astroinformatics (78); Astrostatistics (1882); Neural networks (1933); Time series analysis (1916); Light curve classification (1954); Surveys (1671)

1. Introduction

Deep learning models have become state of the art in an extensive range of tasks, such as image recognition, video analysis, and natural language processing, demonstrating their immense ability to solve complex problems and outperform existing algorithms. Based on this fact, applying deep learning models to the classification of astronomical time series arises as an interesting approach.

Models have progressively increased their number of parameters to achieve such results, from thousands to millions. Unfortunately, architectures with such a large number of parameters are vulnerable to overfitting. Overfitting occurs when models memorize the data available in the training set rather than learning meaningful characteristics from the data so that the model can generalize and perform well when testing new and unseen data. To avoid overfitting, models that achieve state-of-the-art results in different tasks are trained with annotated data sets that have been extensively processed and filtered, and that consist of a large number of samples for each class, thus preventing overfitting.

However, real-world problems present different scenarios in regard to data. For example, not only there is a small number of annotations in astronomical time-series data sets, but the annotations also have highly imbalanced class distributions. While small data sets already hinder learning by making algorithms fail at generalizing characteristics of the data, imbalanced distributions only accentuate this issue (Caruana 2000; He & Garcia 2009). These two characteristics, in addition to the irregularly time-spaced nature of astronomical observations, are a considerable difficulty for machine-learning algorithms and make the classification problem a unique challenge.

To overcome these problems, data augmentation techniques are frequently applied to transform small imbalanced data sets into large and balanced data sets. Most of these techniques, although widely applied in the domain of images, cannot be directly applied in the time domain due to its dissimilar properties. Consequently, augmentation techniques in the time domain remain a challenge and deserve more attention from the community (Wen et al. 2021).

Traditional augmentation techniques in the time domain, such as jittering, window warping, and slicing, assume that these transformations exist naturally in the data and that the augmented samples will be valid time series with properties similar to the existing ones. Moreover, appropriate augmentation techniques are specific to the data set (Iwana & Uchida 2021) and the task (Wen et al. 2021). An example of a data-set-specific technique could be jittering, where additive Gaussian noise is often used in sensor data sets. Yet this method cannot model the heteroscedastic nature of astronomical data. On the task-specific side, we could mention slicing or warping transformations that heavily discard or modify the context of the time series, potentially altering the original samples' class information.

A generative model for data augmentation allows avoiding assumptions about existing transformations in the data. Since we will use the generated samples for classification, the generative model should learn the class-conditional distribution of the data. Therefore, the model can learn how to generate new realistic samples directly from the data and preserve the class information simultaneously.

Because of their ability to model complex real-world data and the wide success they have achieved across a variety of domains (Sampath et al. 2021), generative adversarial networks (GANs; Goodfellow et al. 2014) are the generative models of our choice.

While previous works have explored GAN-based data augmentation methods for classification, most have focused on the image domain (Frid-Adar et al. 2018; Salehinejad et al. 2018; Zhu et al. 2018; Huang et al. 2020) and only a few on the time domain (Ramponi et al. 2018; Zhang et al. 2020). Furthermore, Ramponi et al. (2018) is the only work that addresses astronomical time-series generation.

To the best of our knowledge, none of the existing approaches is suitable for our use case: dealing with irregularly spaced data, allowing for both multi-class and physical parameter conditional generation, and focusing on the downstream task of classification. In addition, the literature lacks a GAN evaluation metric to select appropriate models for classification tasks.

In this work, we propose a GAN-based data augmentation methodology for time series to improve the classification accuracy on two astronomical data sets taken from the Catalina and Zwicky Transient Facility (ZTF) surveys. The main contributions are:

1. Proposing a GAN model capable of performing conditional generation based on class and physical parameters, suitable for irregularly spaced time series.
2. Revealing the incapability of the standard GAN evaluation metric (the Fréchet inception distance (FID)) to assess overfitting and proposing a novel evaluation metric that overcomes this issue to select an adequate generative model.
3. Proposing a resampling technique to delay the occurrence of overfitting.
4. Designing two new data augmentation techniques for time series that produce plausible time series preserving the properties of the original ones.

The remainder of the paper is structured as follows: Section 2 presents a theoretical background of the work. In Section 3 the utilized data sets and their preprocessing are explained. Section 4 explains the proposed methodology. Section 5 presents the obtained results, which are discussed in Section 6, stating its strengths and weaknesses. Finally, Section 7 presents the main conclusions of this work and future steps.

2. Background

2.1. *Imbalanced Data Sets*

Let $\mathcal{D} = \{x_i, y_i\}_{i=1}^{n}$ be a data set where x_i is a real example and $y_i \in Y = \{1, 2, \ldots, c\}$ a class label associated with x_i. $\mathcal{D}$ is said to be imbalanced if the distribution of Y differs significantly from the discrete uniform distribution $\mathcal{U}\{1, c\}$. Therefore, imbalanced data sets are composed of one or more classes (majority classes) that severely outrepresent other existing classes (minority classes) (He & Garcia 2009).

Given an imbalanced data set $\mathcal{D}$, we can apply sampling techniques to transform its class distribution into a uniform. The result of this transformation is a modified version of the original data set, its balanced counterpart $\mathcal{D}^u$.

2.2. *GANs*

The GAN framework consists of a game between two networks. Given an input data set of real samples $x_r \sim P_r$, the *generator* network (G) aims to implicitly approximate the data distribution P_r by performing a mapping between a source of noise and the real sample space. The result of this mapping are fake samples $x_g \sim P_g$ that attempt to resemble the real ones. In contrast, the *discriminator* network (D) tries to distinguish between x_r and fake samples generated by G.

During the training process, the two networks compete against each other without having control of the opponent's parameters. On the one hand, G is trained to generate samples that resemble the real ones, while on the other hand D is trained to predict whether a given sample comes from the input data set or was generated by G. At the end of the training, G will generate samples similar to the ones in the input data set, and the D will be unable to tell apart generated from real samples.

Since the creation of GANs, they have revolutionized the field of generative modeling, showing novel results especially in the domain of images. As a broad overview of the evolution process, we could mention includes conditional-generation models (Mirza & Osindero 2014; Odena et al. 2016), models that stabilize the erratic behavior of the original GANs (Arjovsky et al. 2017; Gulrajani et al. 2017; Miyato et al. 2018), and models that generate samples with an impressively high quality and resolution (Karras et al. 2017; Karras et al. 2018; Brock et al. 2019; Karras et al. 2019) among many other models and applications. An extensive description of GAN models in computer vision is provided in Wang et al. (2022).

GANs have also been applied to the time-series domain, with significant improvements in recent years. The first model capable of generating continuous sequential data was proposed by Mogren (2016) adding recurrent neural networks to the GANs' generator and discriminator to handle the time-series temporal evolution. This work was followed by Esteban et al. (2017), who added label-conditional generation and a focus on downstream medical tasks. More recently, Yoon et al. (2019) introduced a jointly trained embedding network that combines the unsupervised GAN framework with a supervised autoregressive model to capture the time-series conditional temporal dynamics. Lately, Ni et al. (2020) proposed a GAN framework to deal with long time-series data based on an approximation of the Wasserstein distance using the signature feature space, avoiding the usage of costly discriminators and claiming to achieve state-of-the-art results in measures of similarity and predictive ability.

The most related work corresponds to the T-CGAN (Ramponi et al. 2018), which proposes a method to generate irregularly spaced time series. Still, it does not include conditional generation with physical parameters of interest, it does not perform multi-class generation, and similarly to the works mentioned above, it does not tackle the problem of model selection for a downstream task.

2.2.1. *Wasserstein GAN*

The Wasserstein GAN (WGAN; Arjovsky et al. 2017) is one of the GAN models that is widely used and well known for its training stability. This GAN leverages an approximation of the Wasserstein-1 distance to measure the dissimilarity between P_r and P_g. An upgraded version of this model is the WGAN with gradient penalty (WGAN-GP, Gulrajani et al. 2017), which adds a regularization term to the original WGAN loss to satisfy the Lipschitz condition on D. The WGAN-GP objectives that are minimized during the training process are

$$L_G = \underset{x_g \sim P_g}{\mathbb{E}}[D(x_g)] - \underset{x_r \sim P_r}{\mathbb{E}}[D(x_r)], \tag{1}$$

$$L_D = -L_G + \lambda \underset{\hat{x} \sim P_{\hat{x}}}{\mathbb{E}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2], \tag{2}$$

where $P_{\hat{x}}$ is the distribution implicitly defined by sampling uniformly along linear paths between points sampled from P_r and P_g, and λ is the penalty coefficient that controls the strength of the gradient regularization.

2.3. Data Augmentation

Data augmentation refers to a set of techniques applied to a data set used to create new samples that are slightly different from the existing ones to increase the number of samples in the data set. It is frequently used to prevent overfitting, and it helps improve the performance of machine-learning models for various applications (Wen et al. 2021). Classic examples of this in the field of images are rotations, translations, crops, flips, among others.

In the time domain, data augmentation techniques are less standardized. Traditional techniques in time series correspond to nonparametric transformations such as jittering, scaling, window slicing, and window warping (Guennec et al. 2016). Parametric techniques can also be applied in data augmentation, such as the parametric model-based augmentation for transient phenomena proposed in Pimentel et al. (2022).

2.4. Overfitting in GANs

As described in Karras et al. (2020), overfitting in GANs occurs when training on small data sets. The less data there is, the earlier the discriminator becomes too confident in separating real from generated samples, which impedes the progress of G and eventually deteriorates the quality of the generated samples.

Even though Karras et al. (2020) proposed adaptive discriminator augmentation (ADA) as a technique to deal with overfitting in GANs, this technique requires the application of differentiable transformations to augment the training data. Since our goal is to provide a GAN-based data augmentation method motivated by the limited augmentation methods for time series, we intentionally do not include any augmentation method (apart from oversampling) in the GAN-training process; hence, we do not consider using ADA.

2.5. Evaluation of GANs

Even though the losses described in Equations (1) and (2) successfully describe the adversarial problem and quantify the distance between P_r and P_g, their high variance makes them unsuitable for using them as a stopping criterion. Even if they did not suffer from this issue, metrics based on D are specific to their corresponding G, and cannot generalize properties about the generated data set. Consequently, the framework requires additional evaluation metrics to assess the quality of the generated samples and select the definitive generator for the downstream task.

Evaluation of generative models requires a notion of the distance between P_r and P_g. Defining such a measure for high dimensional distributions is a challenging task and remains an open problem (Naeem et al. 2020).

An intuitive way of comparing these distributions is as follows: if a generative model can successfully capture P_r with P_g, the performance on any downstream task should be similar when our data comes from any of the two distributions. Setting the downstream task to classification leads to using classification metrics for evaluation.

2.5.1. Classification Metrics

Considering that the ultimate purpose of this work is to improve the classification of real astronomical objects, we naturally adopt the classification accuracy metric first proposed in Yang et al. (2017) and later used in Esteban et al. (2017), Santurkar et al. (2017), Shmelkov et al. (2018), and Ravuri & Vinyals (2019). For clarity, we choose to preserve the names in Esteban et al. (2017): train on synthetic test on real (TSTR) and train on real test on real (TRTR). These two scores are computed by training a classifier on synthetic (generated) data or real data and then evaluating its classification accuracy on real data.

2.5.2. Feature-based Metrics

Based on the difficulty of finding meaningful metrics in the input space, quantifying the distance between the distributions P_r and P_g often involves mapping samples $x \in \{x_r, x_g\}$ into a feature space with a transformation $x \mapsto \phi(x)$, where ϕ is an intermediate representation of a pretrained classifier (Salimans et al. 2016; Heusel et al. 2017; Sajjadi et al. 2018; Kynkäänniemi et al. 2019; Naeem et al. 2020). The classifier is generally a convolutional neural network (CNN) such as Inception-v3 (Szegedy et al. 2016), a widely used architecture in computer vision.

Since the dimensionality of ϕ is often lower than that of x, the distributions of the feature space are often called *manifolds*. We will informally understand these manifolds as connected regions with a relatively simple structure embedded in a more complex space.

When evaluating generative models, two desired characteristics are fidelity and diversity. The former describes how real the generated samples look in comparison to the real ones, while the latter measures how much of P_r the model can cover with P_g.

FID—This metric proposed by Heusel et al. (2017) consists of a Wasserstein-2 distance between Φ_r and Φ_g, the distributions of ϕ_r and ϕ_g, respectively.

Under the assumption that both distributions are multivariate Gaussians, their mean μ and covariance Σ are estimated to obtain a closed-form of the distance

$$\text{FID} = \underbrace{||\mu_r - \mu_g||^2}_{\text{(a)}} + \text{Tr}(\underbrace{\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}}_{\text{(b)}}). \tag{3}$$

While (a) can be interpreted as a measure of fidelity that indicates the average distance between the two distributions, (b) can be interpreted as a measure of diversity that compares the variability of the two distributions.

A particularly relevant limitation of FID in the presence of highly imbalanced distributions is that computing the last term in (b) requires full-rank Σ matrices, which makes the calculation of a per class FID unfeasible if the minority classes contain fewer samples than the dimensionality of Φ.

Furthermore, even if we had enough samples to compute it, a per class score would be unreliable for the minority classes since FID is known to suffer from high bias for small sample sizes (Bińkowski et al. 2018).

Precision and recall—Sajjadi et al. (2018) proposed separating fidelity and diversity into two relative-density-based metrics: precision and recall. These two metrics improve upon FID by identifying cases of mode dropping or mode inventing in the generated distribution, in the pathological case where different models achieve similar FID values by privileging either one of the two terms in Equation (3).

Improved precision and recall—Motivated by the failure at identifying models with poor variability, Kynkäänniemi et al. (2019) proposed improved precision and recall metrics (P and R). These metrics are computed by estimating the manifolds $\Phi \in \{\Phi_r, \Phi_g\}$ according to

$$\hat{\Phi} = \bigcup_{\phi \in \boldsymbol{\Phi}} B(\phi, NND_k(\phi)), \tag{4}$$

where $\boldsymbol{\Phi} \in \{\boldsymbol{\Phi}_r, \boldsymbol{\Phi}_g\}$ is a collection of feature samples $\phi \in \{\phi_r, \phi_g\}$, the ball $B(x, r)$ is the solid sphere around x with radius r, and $NND_k(\phi)$ is the distance from ϕ to its kth nearest neighbor within the corresponding manifold. In the presence of outliers, the KNN approach results in an overestimation of the manifolds due to the large distances between samples.

Density and coverage—Naeem et al. (2020) proposed density and coverage (D and C) motivated by the vulnerability of P and R to outliers. While P measures fidelity depending on the binary decision of whether a feature sample ϕ_g belongs to the real manifold Φ_r, D considers the amount of balls $B(\phi_r, NND_k(\phi_r))$ within each ϕ_g is contained, adding robustness to real distributions with outliers. On the other hand, C measures diversity based on the real manifold estimate instead of the generated one, in contrast to R.

In our practical case, we found that P and C saturate quickly, not providing meaningful information. Since these metrics directly depend on the real manifold estimates, we hypothesize that this behavior can be caused by the sparsity of Φ_r in the minority classes, leading to the same overestimation issue as outliers. Consequently, we decide to use D and R as our fidelity and diversity metrics.

Let B_r^k be the abbreviation of $B(\phi_r, NND_k(\phi_r))$, and $\hat{\Phi}_g$ the approximation of the generated manifold described in Equation (4), we compute the D and R metrics according to

$$D_{(\boldsymbol{\Phi}_r,\boldsymbol{\Phi}_g)} = \frac{1}{k|\boldsymbol{\Phi}_g|}\sum_{\phi_g\in\boldsymbol{\Phi}_g}\sum_{\phi_r\in\boldsymbol{\Phi}_r} 1_{B_r^k}(\phi_g), \tag{5}$$

$$R_{(\boldsymbol{\Phi}_r,\boldsymbol{\Phi}_g)} = \frac{1}{|\boldsymbol{\Phi}_r|}\sum_{\phi_r\in\boldsymbol{\Phi}_r} 1_{\hat{\Phi}_g}(\phi_r), \tag{6}$$

where $1_A(x)$ is the indicator function defined as

$$1_A(x) = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{if } x \notin A. \end{cases} \tag{7}$$

3. Data

3.1. Data Sets

Because of the recognizable shapes of their light curves when visualized in phase space, we focus on periodic variable stars. However, the framework could be effortlessly extended to other stars of interest if needed. We perform and validate our experiments on data captured by two time-domain astronomical surveys.

Table 1
Distribution of the Adopted Classes for the Catalina Surveys Data Release-1

New Class	Original Class
EBSD/D	Contact eclipsing binary (EW) Semidetached eclipsing binary (β Lyrae)
RRL	Fundamental mode RR Lyrae (RRab) First overtone mode RR Lyrae (RRc) Multimode RR Lyrae (RRd) Long-term modulation (Blazkho)
EBC	Detached eclipsing binary (EA)
LPV	Long period variables (LPV)
DSCT	High amplitude δ Scuti (HADS) Low amplitude δ Scuti (LADS)
CEP	Anomalous Cepheids (ACEP) Type II Cepheids (Cep-II)

Note. The original class acronyms as described in Drake et al. (2014) are shown in (·).

Catalina Surveys Data Release-1—This catalog described in Drake et al. (2014), captured with the 8.2 deg^2 field-of-view camera mounted on the CSS 27 inch Schmidt telescope, provides ∼61,000 light curves of periodic variable objects, with their corresponding periods and classes. To decrease the complexity of the multi-class problem induced by the large number of periodic classes provided, we only consider a subset of the periodic objects grouped following the mapping described in Table 1.

ZTF—This survey (Bellm et al. 2018) provides a public multiband stream of alerts captured by a 47 deg^2 field-of-view camera mounted on the Palomar 48 inch Schmidt telescope, is capable of scanning the entire northern sky every three nights and the plane of the Milky Way every night. To enable further analysis in follow-up telescopes, the alerts are processed by alert brokers that are designed to provide a rapid and self-consistent classification. We use the subset of periodic variable stars present in the ZTF training set created by the ALeRCE broker (Förster et al. 2021), along with their taxonomy. This training set was constructed considering sources observed by ZTF whose labels had been cross matched from different multiple catalogs.

Previous works (Carrasco-Davis et al. 2021; Sánchez-Sáez et al. 2021) have already used ZTF data processed by the ALeRCE broker to train different machine-learning algorithms. More details about the data processing can be found in Förster et al. (2021).

After preprocessing both data sets following the steps detailed in Section 3.2, we obtain the definitive versions of the data sets that will be used in our experiments, from now on referred to as the "Catalina" and the "ZTF" data sets. The class distributions of the preprocessed data sets are shown in Table 2.

3.2. Data Preprocessing

To use the data described in Section 3.1, some preprocessing steps need to be applied. The preprocessing consists of four

Table 2
Distribution of the Classes of the Preprocessed Data Sets

Catalina		ZTF	
Class	No. of Samples	Class	No. of Samples
EBSD/D	28,980	EB	31,477
RRL	7533	RRL	18,729
EBC	4500	LPV	5245
LPV	483	DSCT	507
DSCT	241	CEP	471
CEP	182		

main steps: period folding, outlier filtering, time sampling, and median centering.

3.2.1. Period Folding

Since the desired characteristic shapes of periodic light curves are only visible in the phase space, we start by folding the light curves into the period provided in both data sets. Denoting the light-curve period as T, and the observation time as t, the folding operation is performed by converting t into ϕ_t according to

$$\phi_T \equiv t (\mathrm{mod}\ \ T), \tag{8}$$

$$\phi_t = \frac{\phi_T}{T}, \tag{9}$$

where the congruence symbol $\equiv$ in Equation (8) refers to the modulo operator with modulus T.

With this operation, we transform times with a variable range of values to phases with values bounded between 0 and 1. This transformation is convenient because multiple neural networks will process the phases, and having inputs with a similar range is a desirable property when training such algorithms.

3.2.2. Outlier Filtering

Considering that some of the light curves in the data sets can include a significant amount of noise, we filter out anomalous observations within each curve of both data sets. These anomalous observations are in general isolated observations with a magnitude that does not follow the general behavior of the magnitudes in the light curve, and including them could be detrimental to the performance of our algorithms. For the Catalina data set, the anomalous behavior is quite particular to each light curve, and a general threshold filtering cannot be applied; therefore, a different approach is needed.

The Catalina light curves are filtered by comparing each magnitude with the local statistics of the magnitudes' neighborhood. This comparison is performed using the z-score[4] of the magnitudes within a window that considers only a portion of the light curve. The process is performed by sliding the window through the entire light curve with a window size $w_s = 20$, removing the outlier observations that satisfy $z_{\mathrm{score}} > 3$, and repeating 2 times per light curve since consecutive outlier observations can significantly alter the moving window's statistics and not be detected in a single pass. The results of this filtering step are shown in Figure 1(b). After this step, we perform a second filtering stage by discarding the light curves that contain more than 90% of their magnitudes out of the range delimited by the class medians and class standard deviations.

On the other hand, anomalous observations in the ZTF data have been already marked with a magnitude of 100. Hence, these observations can be filtered out by a simple threshold. Following the filtering steps used by Sánchez-Sáez et al. (2021), we use $\mathrm{mag_{thr}} = 30$.

3.2.3. Time Subsampling

To bring the problem to a more straightforward domain, we set the length of the light curves to a predefined value for each data set. With this simplification, we can work with convolutional architectures rather than recurrent architectures that could hinder the GAN's training stability by violating the Lipschitz constraint, adding extra complexity to the problem.

Given a light curve with an arbitrary number of m observations, we obtain the fixed-length light curves by randomly choosing n from the m available observations. Considering that we choose our points with no particular bias, this approach should give a reasonable approximation of the original light curve if n is not too small compared to m.

Since both of our real data sets contain irregularly sampled light curves, and we perform the subsampling step after the period folding step, choosing an observation implies selecting a magnitude with its corresponding observation phase. Both magnitudes and phases are part of the input of our models, as will be detailed in Section 4. Figure 1(b) shows an example of the time subsampling step.

The light-curve length is set to 100 observations for the Catalina data set, whereas that of the ZTF data set is 40 observations, consistent with the fact that ZTF is a relatively new sky survey with a lower number of observations per object compared to the Catalina Survey.

After discarding the light curves that do not have the minimum length to perform this step, we end up with approximately 41 k and 56 k samples in the Catalina and ZTF data sets, respectively, whose class distribution is shown in Table 2.

3.2.4. Median Centering

The last step to get the data ready for data generation is centering it around zero so all the magnitudes have a consistent range that can be learned from the generator. This is done for each light curve by subtracting the center (median) of the magnitudes. We compute the median instead of the mean because of its robustness to outlier magnitudes.

This step is necessary because G is a neural network that outputs a tanh activation, and it can only generate values in a symmetrical range around zero. It is worth mentioning that we could center the data around any other offset, which would require to also include that offset to the output of the generator; the importance of performing this step is not the value of the offset itself, but rather the unification of all the magnitudes around a single value so our generator can model them.

4. Methodology

4.1. General Description

We propose a conditional-generation approach that extends the T-CGAN (Ramponi et al. 2018), adding the class and

[4] The z-score is the distance of an observed value x to the population mean μ, measured in terms of the population standard deviation σ. It is computed by $z = \frac{x-\mu}{\sigma}$.

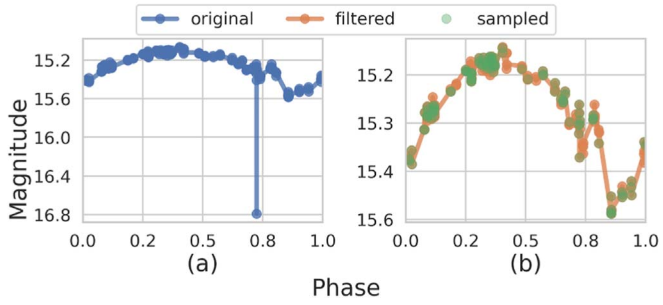


Figure 1. (a) Original Cepheid from the Catalina data set. (b) Filtered and subsampled versions of the original Cepheid.

amplitude of the light curves to the conditional parameters, which include the observation phases according to the original model. The details of how the conditional parameters are included into the model will be explained in Section 4.2.

A summary of the proposed methodology, that details the partitions of data sets for the models and metrics is provided in Figure 2.

We start by partitioning the preprocessed data set $\mathcal{D}$ into $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, and $\mathcal{D}_{\text{test}}$, the *train, validation*, and *test* sets. Each class in $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{test}}$ contains 20% of the total number of samples of the smallest class in $\mathcal{D}$. To train the GAN and the classifier we use $\mathcal{D}^{u}_{\text{gen}}$ a uniformly balanced version of the original $\mathcal{D}_{\text{train}}$ obtained through the resampling block that will be explained in Section 4.6.

After training the GAN, we use G to create a synthetic uniformly balanced data set $\mathcal{D}^{u}_{\text{gen}}$. Since G performs conditional generation, to generate a uniformly balanced data set we sample the conditional vectors $\bar{z}$ from $\mathcal{D}^{u}_{\text{train}}$. It is essential to mention that the generated data set will follow the distribution of the data set from which we sample the conditional vectors. For example, sampling them from $\mathcal{D}_{\text{train}}$ would imply generating a heavily unbalanced data set. To obtain the TSTR score, we train a classifier on $\mathcal{D}^{u}_{\text{gen}}$ and evaluate its accuracy on a real data set.

We compare the TSTR score to multiple TRTR scores, computed in a similar manner but using $\mathcal{D}^{u}_{\text{train}}$(or slightly modified versions of it) instead of $\mathcal{D}^{u}_{\text{gen}}$. This comparison is reasonable because the data sets used for evaluation ($\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{test}}$) are fixed and balanced by construction: their sampling process from $\mathcal{D}$ is designed to have the same amount of samples per class.

4.2. Details of the Data Structure

Let ϕ_t, a, and c denote the observation phases, amplitudes, and classes of the light curves, respectively, our GAN's generator requires a sample $\bar{z} = [\phi_t, a, c]$ from the real data set $\mathcal{D}_{\text{train}}$ and a sample $z \in \mathbb{R}^{\ell} \sim \mathcal{N}(0, I)$. The latent space dimensionality ℓ is set to 16 and 8 for the Catalina and ZTF data sets, respectively, obeying roughly the proportion between the light-curve lengths of the data sets. Following a CGAN-like approach (Mirza & Osindero 2014), the concatenation of z and $\bar{z}$ is passed as an input to G to generate synthetic samples.

The conditional parameters are also inputs of D similarly concatenated with real or generated magnitudes. We create a tensor version of the conditional parameters for this concatenation to be viable. Let $\boldsymbol{a}$ and $\boldsymbol{c}$ be tensor versions of a and c, and L and N denote the light-curve length and number or classes of a data set; we define $\boldsymbol{a} \in \mathbb{R}^{L}$ as a vector with value a in all its components, and $\boldsymbol{c} \in \mathbb{R}^{L \times N}$ as a one-hot encoding of c, composed by 0 and 1 vectors, where $\{0, 1\} \in \mathbb{R}^{L}$. The tensor version of $\bar{z}$ is $\bar{\boldsymbol{z}} = [\phi_t, \boldsymbol{a}, \boldsymbol{c}] \in \mathbb{R}^{L \times 2+N}$. The concatenation of$\bar{\boldsymbol{z}}$ and real or generated magnitudes will be the input of D, and will be dimensions $L \times 3 + N$.

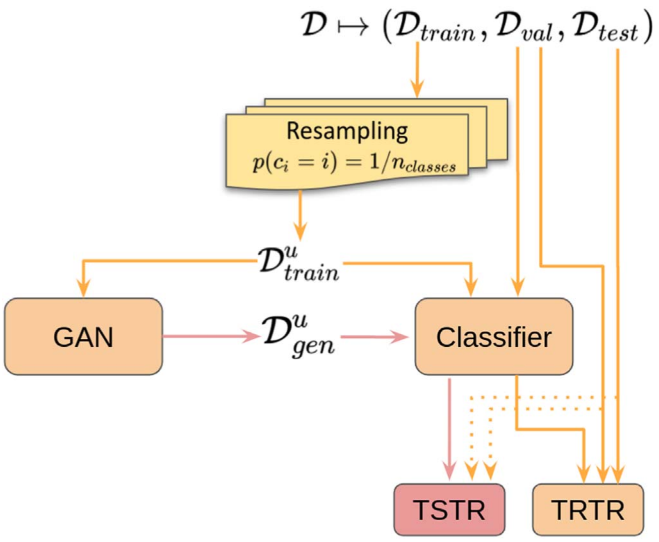

$\mathcal{D} \mapsto (\mathcal{D}_{train}, \mathcal{D}_{val}, \mathcal{D}_{test})$
Resampling
$p(c_i = i) = 1/n_{classes}$
$\mathcal{D}^{u}_{train}$
GAN
$\mathcal{D}^{u}_{gen}$
Classifier
TSTR
TRTR

Figure 2. Diagram of the methodology.

Table 3
Classifier Architecture

Input	$x \in \mathbb{R}^L$
	$\phi \in \mathbb{R}^L$
Conv. block	$2 \rightarrow 32$
Conv. block	$32 \rightarrow 64$
Conv. block	$64 \rightarrow 128$
Conv. block	$128 \rightarrow 64$
Conv. block	$64 \rightarrow 64$
Dense	$\lceil L/32 \rceil \times 64 \rightarrow 100$
BN, ReLU, dropout	$100 \rightarrow 100$
dense, softmax	$100 \rightarrow N$

Convolution block ($p_s = 2$, $k_s = 3$, c_{in}, c_{out})	
Block input	$l_i \times c_{\text{in}}$
1D convolution, BN	$l_i \times c_{\text{in}} \rightarrow l_i \times c_{\text{out}}$
Max pooling, ReLU	$l_i \times c_{\text{out}} \rightarrow \lceil l_i/2 \rceil \times c_{\text{out}}$

Note. L and N correspond to the light-curve length and number of classes, respectively, and they vary depending on the selected data set as mentioned in Section 3. The fixed block parameters p_s and k_s stand for pool size and kernel size. Since the convolution blocks always halve the temporal dimension, we only specify their channel dimensions c_{in} and c_{out}.

4.3. Classifier Details

To reduce the variance of the experiments, the classifier consists of an ensemble of five identical base classifiers trained independently. The base classifier is a CNN that receives the concatenation of the magnitudes x and phases ϕ_t following the classification scheme in Ramponi et al. (2018). The input is forwarded through a set of convolution blocks that halve the temporal dimension, followed by dense layers. The network is trained using the Adam optimizer (Kingma & Ba 2015) with $\alpha = 0.0001$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. Table 3 shows the detailed architecture of the base classifier. To compute all the feature-based metrics explained in Section 2.5.2, we use the output of the last convolution block of this base classifier, trained on each of the data sets separately.

4.4. GAN Details

In addition to the original WGAN-GP formulation, we include additional regularization terms to Equations (1) and (2). Following an AC-GAN-like approach (Odena et al. 2016), the output of D has two components: $D_{rg} \in \mathbb{R}$ that tries to separate real from fake samples and $D_y \in \mathbb{R}^N$ that tries to predict the class of the input. Therefore, we add a cross-entropy regularization of real and generated samples to the discriminator loss. Also, to prevent the GAN equilibrium from happening in any arbitrary offset, we add a regularization term to prevent $D_{rg}(x_r)$ from drifting too far away from zero, as proposed in Karras et al. (2017). To the generator loss, we only add the cross-entropy regularization of generated samples. Consequently, the losses minimized in the proposed framework are

$$\widetilde{L_D} = L_D + \xi(H_r + H_g) + \epsilon\mathbb{E}[D_{rg}(x_r)^2], \tag{10}$$

$$\widetilde{L_G} = L_G + \xi H_g, \tag{11}$$

where $H_r = H(y_r, D_y(x_r))$ and $H_g = H(y_g, D_y(x_g))$ correspond to the cross entropy between the real labels and the discriminator predictions, y_g are the real labels used to generate x_g, and $\xi = 0.001$ and $\epsilon = 1$ control the strength of each regularization term.

We perform $n_{\rm disk} = 5$ discriminator iterations per generator iteration, and train for 400 K generator iterations using the Adam optimizer with $\alpha = 0.0001$, $\beta_1 = 0.5$, and $\beta_2 = 0.9$. At the training time, we compute the exponential moving average (Yazici et al. 2019) with a decay of 0.999 for the generator weights, to be used when generating samples for evaluation. A full description of the GAN architecture is shown in Table 4.

On the one hand, G receives the concatenation of the noise source z and the conditional variables $\bar{z}$ as an input, and it forwards it through a dense layer followed by a set of strided deconvolutions that duplicate the temporal dimension of every block and simultaneously halving the number of channels (except for the last block). On the other hand, D receives the concatenation of the magnitudes x and the conditional tensor $\bar{\mathbf{z}}$, and it forwards it through a set of strided convolutions that halve the temporal dimension of every block and duplicate the number of channels, followed by a dense layer.

4.5. Preliminary Experiment: The u-GAN

With all details and parameters provided in the above sections, we perform a preliminary experiment using $\mathcal{D}^u_{\rm train}$—the uniformly balanced version of $\mathcal{D}_{\rm train}$—as the GAN-training set, to then generate $\mathcal{D}^u_{\rm gen}$ and obtain the TSTR accuracy scores. This GAN setup will be referred to as "u-GAN."

It is worth mentioning that this setup is the standard approach when training machine-learning algorithms, where $\mathcal{D}^u_{\rm train}$ is usually preferred over $\mathcal{D}_{\rm train}$ because it reduces the biases toward the most populated classes, induced by the highly imbalanced class distribution of $\mathcal{D}_{\rm train}$.

The first finding of performing this preliminary experiment is that the TSTR accuracy score can vary significantly, depending on how long we train the GAN. For this reason, we analyze the behavior of different GAN models throughout the training process to find an adequate criterion for model selection. Figure 3 shows the evolution of the validation TSTR accuracies and FID scores every 10 k iterations. Since computing TSTR accuracies involves training multiple classifiers, evaluating this score more frequently is unfeasible.

Table 4
GAN Architecture

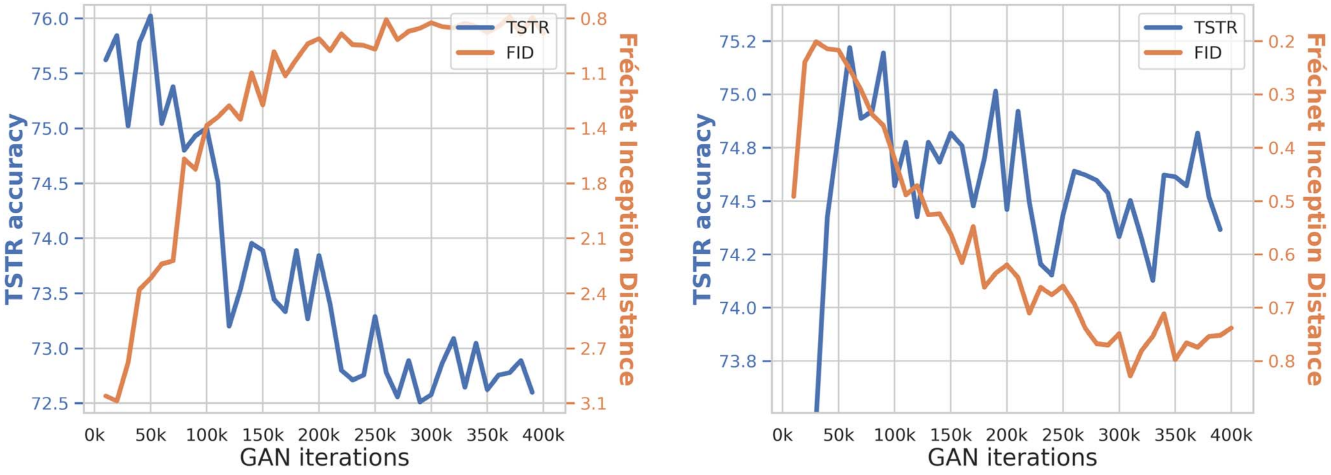
(a) Generator	
Input	$z \in \mathbb{R}^{\ell}$ $\bar{z} \in \mathbb{R}^{L+1+N}$
Dense, ReLU	$\ell + (L + 1 + N) \rightarrow 4 \times 1024$
Deconv. block	$1024 \rightarrow 512$
Deconv. block	$512 \rightarrow 256$
Deconv. block	$256 \rightarrow 128$
Deconv. block	$128 \rightarrow 64$
Deconv. block	$64 \rightarrow 1$
Tanh · s	$L \times 1$
Deconvolution block ($s = 2$, $k_s = 5$, $c_{\rm in}$, $c_{\rm out}$)	
Block input	$l_i \times c_{\rm in}$
1D deconvolution	$l_i \times c_{\rm in} \rightarrow 2l_i \times c_{\rm out}$
ReLU	$2l_i \times c_{\rm out}$
(b) Discriminator	
Input	$x \in \mathbb{R}^{L\times 1}$ $\bar{\mathbf{z}} \in \mathbb{R}^{L\times(2+N)}$
Conv. block	$1 + (2 + N) \rightarrow 64$
Conv. block	$64 \rightarrow 128$
Conv. block	$128 \rightarrow 256$
Conv. block	$256 \rightarrow 512$
Conv. block	$512 \rightarrow 1024$
Dense	$\lceil L/32 \rceil \times 1024 \rightarrow N + 1$
Convolution block ($s = 2$, $k_s = 5$, $c_{\rm in}$, $c_{\rm out}$)	
Block input	$l_i \times c_{\rm in}$
1D convolution	$l_i \times c_{\rm in} \rightarrow \lceil l_i/2 \rceil \times c_{\rm out}$
LeakyReLU	$\lceil l_i/2 \rceil \times c_{\rm out}$

Note. ℓ, L, and N correspond to the latent space dimensionality, light-curve length and number of classes, respectively, which depend on the selected data set as mentioned in Sections 3 and 4. The fixed block parameters s and k_s stand for stride and kernel size, respectively, and l_i represents the input length of the blocks. Since the convolution/deconvolution blocks always adjust the temporal dimension by a factor of 2, we only specify their channel dimensions $c_{\rm in}$ and $c_{\rm out}$.

The preliminary experiment shown in Figure 3 raises two major concerns that will be addressed in the following sections:

1. The TSTR accuracy reaches an optimal value early in the GAN training and then decreases consistently, coinciding with the GAN-overfitting phenomenon explained in Section 2.4.
2. The FID—the standard metric for evaluating GANs—cannot always measure the drop in sample quality reflected in the TSTR accuracy curve, as shown in Figure 3(a).

The behavior detailed in (a) can be understood as follows: in a balanced data set such as $\mathcal{D}^u_{\rm train}$, overfitting is not only strongly influenced by the limited amount of training samples, but it also is exacerbated by the amount of imbalance of the original class distribution of $\mathcal{D}_{\rm train}$. As the imbalance grows,



(a) Catalina

(b) ZTF

Figure 3. Evolution of the validation TSTR accuracy and FID over the course of GAN training for the different data sets. Both scores were computed every 10 k iterations of a single GAN model. The computation of the FID was done with 50 k generated samples divided into 10 batches and the entire real data set, as suggested in Heusel et al. (2017).

samples in the minority classes need to be excessively repeated in order to equate the number of samples in the majority classes, resulting in quick GAN overfitting caused by D learning fast how samples of the minority classes look. The rapid decay in validation TSTR accuracy is problematic considering that we need to compute this metric every 10 k iterations. Hence, the best model selected by this metric could be suboptimal if the decay occurs suddenly, which motivates the proposed resampling block explained in Section 4.6.

The discrepancy described in (b), although undesirable, is not surprising; it was also reported in Ravuri & Vinyals (2019), and it is completely plausible considering the limitations of FID related to mode dropping and mode inventing mentioned in Section 2.5.2. These two phenomena can drastically affect how P_g relates to P_r and thus affect the TSTR accuracy without being reflected in the FID, which suggests that FID is not always reliable in the presence of highly unbalanced data sets, and motivates the proposed $\mathcal{G}$-score for model selection explained in Section 4.7.

4.6. Resampling Block

Motivated by the rapid GAN overfitting shown in Figure 3, we propose a resampling operation that can successfully delay the occurrence of this behavior.

The resampling operation consists of continuously drawing samples from the N classes of a data set $\mathcal{D}$, to modify its class distribution. Let S be the number of samples of $\mathcal{D}$. We start by splitting $\mathcal{D}$ into N sub-data sets $\{\mathcal{D}_i\}_{i=1}^N$ of size $\{S_i\}_{i=1}^N$, where each data set $\mathcal{D}_i$ only contains samples from the ith class. From each sub-data set, we draw without replacement until there are no samples left, then $\mathcal{D}_i$ is shuffled and the sampling process continues.

The goal of this operation is to modify the class distribution of $\mathcal{D}$ by controlling the probability ${}_{pi}$ of drawing a sample from each $\mathcal{D}_i$. The resampling block serves as a generalization of the uniform balancing operation by extending the target class distribution to nonuniform distributions. To illustrate this clearly, we describe two edge cases. On the one hand, we could leave the original class distribution unbalanced by setting ${}_{pi} = S_i/S$, in which case the resampling block does not affect the class distribution, and it would be equivalent to a *shuffle and repeat* operation. On the other hand, we could obtain the balanced version of $\mathcal{D}$ by simply setting ${}_{pi} = 1/N$, which is how we get $\mathcal{D}^u_{\text{train}}$ from $\mathcal{D}_{\text{train}}$.

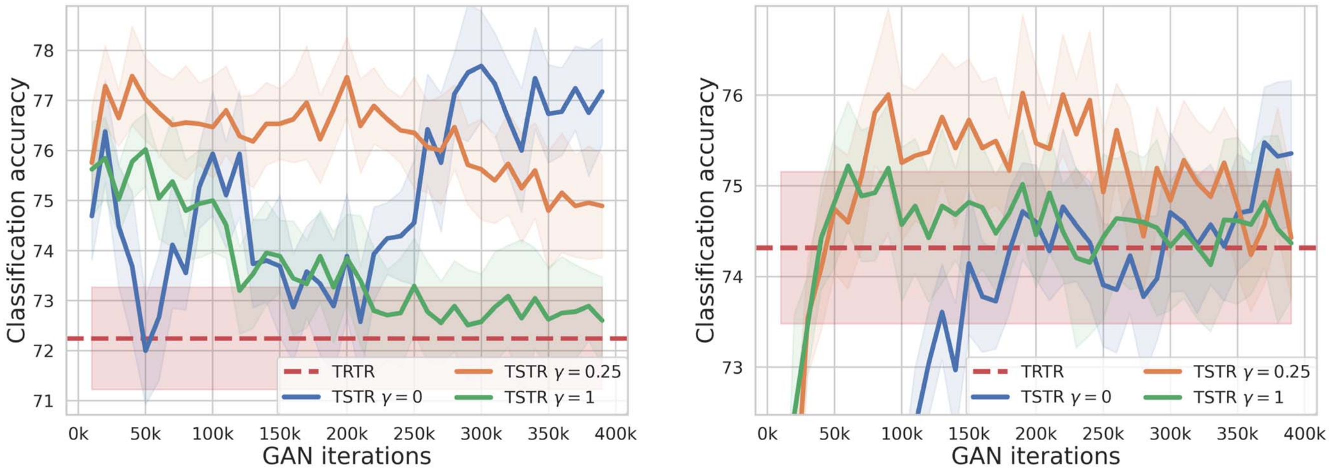
Apart from these two scenarios, we could also generate any data set $\mathcal{D}^\gamma$ whose class distribution lies *in between* that of $\mathcal{D}$ and $\mathcal{D}^u$, created by linearly interpolating between the aforementioned probabilities:

$$p_i = \gamma\left(\frac{1}{N}\right) + (1 - \gamma)\frac{S_i}{S}, \text{ where } 0 < \gamma < 1, \qquad (12)$$

where the two edge cases can be recovered with $\gamma = 0$ for the imbalanced $\mathcal{D}$, and $\gamma = 1$ for the balanced $\mathcal{D}^u$. By using the proposed γ-resampling we are able to control the overfitting speed of the model, as shown in Figure 4. Training a GAN with $\mathcal{D}^u(\gamma = 1)$ implies that all the samples from the minority classes are rapidly shown to the model, leading to fast overfitting. On the other hand, using $\mathcal{D}(\gamma = 0)$ implies that training batches rarely contain a sample from the minority classes (one every 230 samples will be Cepheids of the Catalina data set, roughly 1 Cepheid every four batches), avoiding fast overfitting but inducing slow and unstable training. Training with $\mathcal{D}^\gamma(0 < \gamma < 1)$ allows a reasonable learning pace without overfitting rapidly, as shown in Figure 4 for $\gamma = 0.25$. A model trained with $\mathcal{D}^\gamma$ will be referred to as "γ-GAN."

4.7. Model Selection: The $\mathcal{G}$-score

As mentioned in Section 4.5, the behavior of TSTR accuracies shown in Figure 3 evidences the need for a criterion to choose an adequate G. While using the validation TSTR accuracy for model selection might look appropriate, doing so involves training new classifiers for every candidate of G, an operation that becomes computationally expensive. The problem then lies in finding a fast-to-compute metric that correlates with the TSTR accuracy (and implicitly with the quality of the generated samples).



(a) Catalina

(b) ZTF

Figure 4. Evolution of TSTR accuracy over the course of GAN training for different values of γ. The figure shows mean ± standard deviation over 15 independent runs of the classifier and a single GAN model. The computation of both metrics was done every 10 k GAN iterations.

The natural option for this metric would be FID, but as also shown in Figure 3(a), it fails to measure the decrease in quality of the generated samples reflected in the TSTR accuracy curve. Additionally, since FID is only a measure of the distance between P_g and P_r, it cannot differentiate between the fidelity and diversity of the generated samples (Naeem et al. 2020), and it provides an arbitrarily weighted average between them.

As an alternative, we propose a metric that leverages equally two measures of fidelity and diversity: density(D) and recall (R). Figure 5 shows the results of computing the per class density and recall metric for the Catalina data set.

The fact that D values are not bounded by one is consistent with the formula presented in Equation (5) and can happen if points in the generated manifold in average belong to more than K balls of the real manifold, which is probably caused by the overestimation of the real manifold mentioned in Section 2.5.2, due to sparse feature spaces. An illustration of this situation is shown in Figure 6, where the sparsity in the real distribution causes that the generated samples in average belong to more than $K = 2$ balls, leading to $D = \frac{1}{4}\left(\frac{2}{2} + \frac{3}{2} + \frac{4}{3} + \frac{3}{2}\right) = 1.5$. Additionally, if we reduce the sparsity of the real distribution by removing the furthest sample (bottom left), we get $D = \frac{1}{4}\left(\frac{1}{2} + \frac{2}{2} + \frac{3}{3} + \frac{2}{2}\right) = 1$.

Since R is bounded between 0 and 1 by definition, the unbounded behavior of D is undesirable because it favors D over R in any mean we compute between them. In addition, we find that D also presents a clear bias toward the less populated classes. To overcome these problems, we perform a per class min-max normalization to D and R according to Equation (3).

$$D_i^{'} = \frac{D_i - D_i^{\min}}{D_i^{\max} - D_i^{\min}}, \tag{13}$$

$$R_i^{'} = \frac{R_i - R_i^{\min}}{R_i^{\max} - R_i^{\min}}, \tag{14}$$

where the subscript $(\cdot)_i$ denotes score of the ith class, and the superscripts $(\cdot)^{\min,\max}$ denote the minimum and maximum score of the class, respectively.

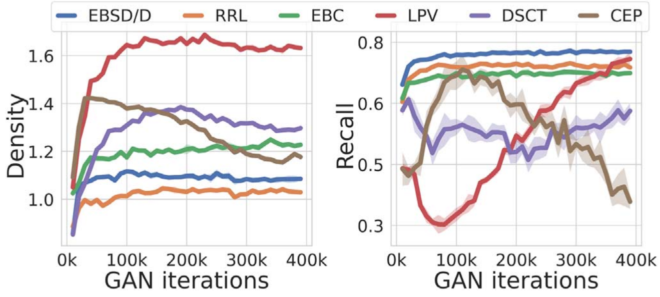


Figure 5. Class density and recall metrics of the Catalina data set.

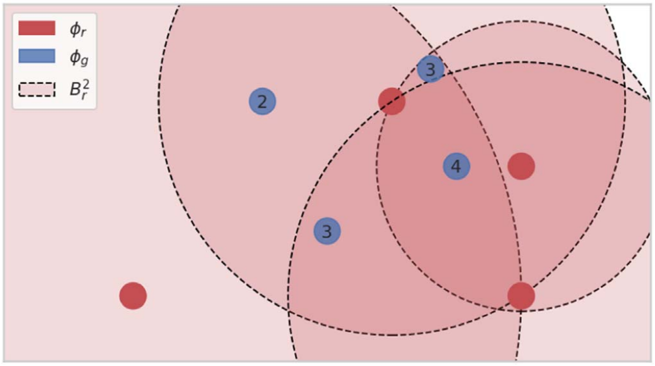


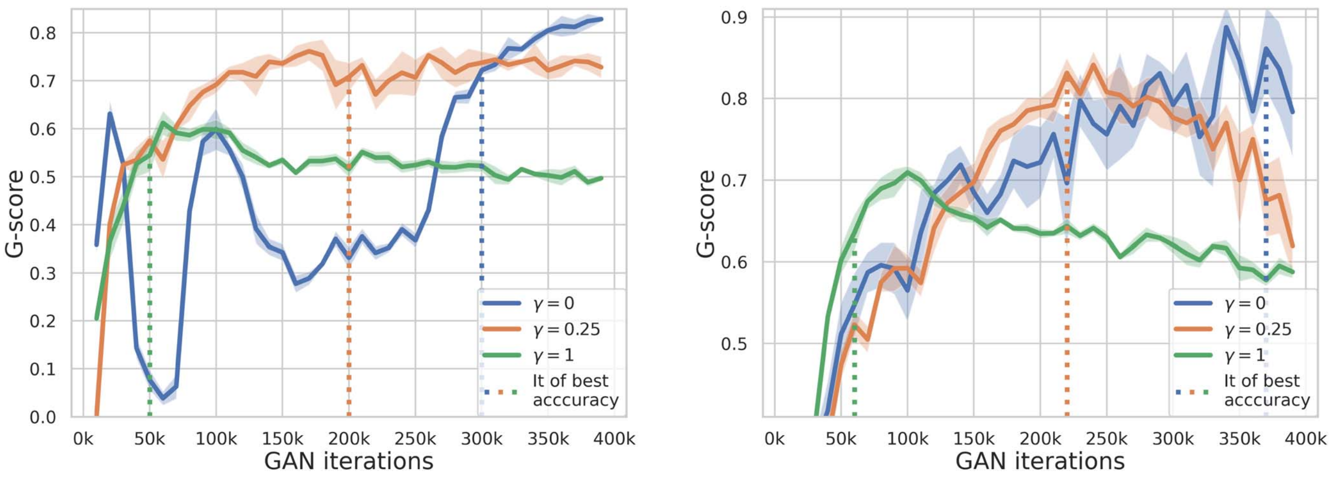
Figure 6. Two-dimensional scenario that illustrates a case in which D is not bounded by 1. The dashed lines show the regions B_r^2: circles around the real feature samples ϕ_r, with radii equal to the distance to their second nearest neighbors. The numbers inside each sample ϕ_g denote the number of circles that enclose the sample.

After the class scores are normalized, we combine them in an equally weighted F-score described in Equation (15). Finally, considering that we are equally interested in the different classes, the $\mathcal{G}$-score is obtained by computing the balanced F-score (macro F-score), as shown in Equation (16).

$$F_i = \frac{2D_i^{'}R_i^{'}}{D_i^{'} + R_i^{'}}, \tag{15}$$

$$\mathcal{G} - \text{score} = \frac{1}{N}\sum_i F_i. \tag{16}$$

When computing the balanced F-score, we prefer the mean of the class F-scores over the F-score of the class means



(a) Catalina

(b) ZTF

Figure 7. Evolution of the $\mathcal{G}$-score for different gamma values over the course of GAN training for the different data sets. Each curve shows mean ± standard deviation over five computations of the metrics, for a single GAN model.

intending to weight equally majority and minority classes, as suggested by Opitz & Burst (2019).

The results of computing the $\mathcal{G}$-score for multiple GANs trained with different values of γ are shown in Figure 7. As can be seen, the $\mathcal{G}$-score curves and validation accuracy curves from Figure 4 seem to have a high correlation, which becomes more evident when analyzing the $\gamma = 0$ curve for the Catalina data set.

4.8. Baselines

To evaluate our generated data sets in the classification task, we compare the TSTR classification accuracies to multiple baselines. These baselines consist of TRTR classification accuracy scores when training in augmented real data sets. It is worth mentioning that the training sets used to compute the scores are all balanced data sets, either GAN generated (TSTR) or real augmented (TRTR).

Acknowledging the heteroscedastic behavior of astronomical data, we do not consider jittering as a suitable operation for the problem. Additionally, we discard utilizing window-slicing techniques since our convolutional architectures work on preprocessed time series with a fixed number of observations. Consequently, our augmentation methods consist of oversampling and different window-warping-based operations.

4.8.1. Oversampling

The oversampling augmentation corresponds to generating the balanced data set $\mathcal{D}^{u}_{\text{train}}$ by repeating samples from the original data set $\mathcal{D}_{\text{train}}$, using the resampling block described in Section 4.6.

4.8.2. Window Warping

Let $x(t)$ be a continuous signal sampled at times t. The window-warping operation starts by selecting a random time window delimited by the values $[t_1, t_2]$, where all the times t_w in the window satisfy $t_1 \leqslant t_w \leqslant t_2$. The warping operation expands or contracts the signal by scaling the variations Δt in t_w and shifting the times $t > t_2$ accordingly, altering the time series' length.

Since we work with folded light curves in phase space, window-warping expansion could be incongruous with the fact that the phase space has an upper bound of 1. Consequently, we derive a new transformation to avoid such incongruence: soft window warping.

4.8.3. Soft Window Warping

We preserve the core idea of window warping by designing expansions and contractions that do not increase the time series' length. Given a random window, we formulate the problem as finding a mapping $t_w \mapsto f(t_w)$ such that the length of the transformed window is at most that of the original, this is $f(t_1) \geqslant t_1$, $f(t_2) \leqslant t_2$. We believe that expansions and contractions should be naturally performed with respect to the center of the window, expanding from the center to the limits and contracting from the limits to the center.

A mapping that meets these requirements is

$$\begin{aligned} f(t_w) &= a + b \cdot \tanh(k(t_w - c)) \\ a &= c = (t_1 + t_2)/2 \\ b &= (t_2 - t_1)/2, \end{aligned} \tag{17}$$

where the values of a, b, and c are determined by the desired behavior with respect to the center of the window. The constant k is randomly sampled in the interval $\left[\frac{1}{2a}, \frac{2}{a}\right]$ and it modulates the strength of the expansions or contractions by modifying the saturation degree of the $\tanh(\cdot)$, producing expansions when saturated and contractions otherwise.

Even though the proposed transformation is designed to be applied across the time axis, it can be easily extended to the signal axis by noting that since the time intervals are monotonous, t_1, t_2 are the minimum and maximum values in the window respectively. Hence, the natural extension to the

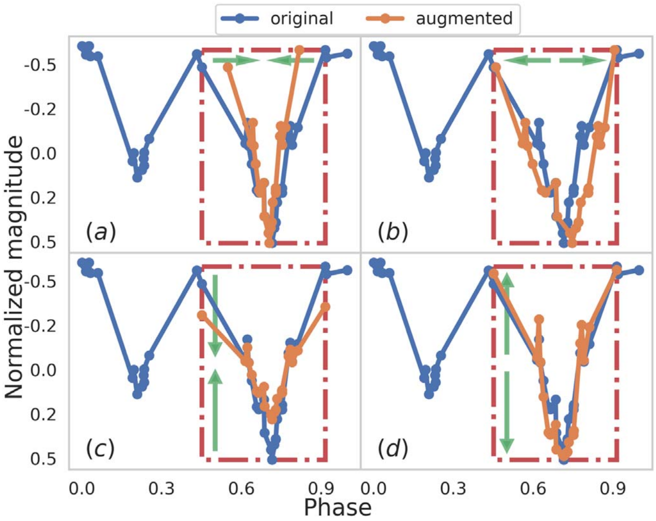


Figure 8. Examples of the soft window-warping transformations for an eclipsing binary of the ZTF data set. (a) Soft time-warping contraction. (b) Soft time-warping expansion. (c) Soft magnitude-warping contraction. (d) Soft magnitude-warping expansion.

signal axis is

$$\begin{aligned} f(x_w) &= a + b \cdot \tanh(k(x_w - c)) \\ m_1 &= \min_{t \in t_w} x(t) \\ m_2 &= \max_{t \in t_w} x(t) \\ a = c &= (m_1 + m_2)/2 \\ b &= (m_2 - m_1)/2. \end{aligned} \tag{18}$$

When applying these transformations to our astronomical light curves, we consider the signal axis as the magnitude axis, and the time axis as the phase axis. These two transformations referred to as soft time warping and soft magnitude-warping, are illustrated in Figure 8. The result of simultaneously applying these two transformations will be referred to as soft mixed warping.

5. Results

5.1. Generated Samples

Figure 9 shows some samples of the GAN-generated light curves. The conditional vector $\bar{z}$ used to generate these samples considers phases, amplitudes, and classes of the real data shown in the first two columns. Accordingly, and as it can be seen, most of the generated samples preserve the real class and amplitude. It is worth mentioning that although some generated samples present normal fluctuations in phase and magnitude with respect to the real ones, there are also samples that do not look plausible (see Figure 10), which could be attributed to the lack of truncation techniques or any type of filtering to improve the fidelity of the generated samples, which we address in Section 6.

5.2. Classification

The classification accuracies obtained by using different training sets are shown in Table 5. The first four rows show TRTR classification results when training on real data that has been augmented with the random transformations described in Section 4.8. The soft-warping transformations (rows B–D) are applied to the data set previously balanced by oversampling. The last four rows show TSTR classification results when training on GAN-generated data, comparing the proposed γ-resampling for GAN training (γ-GAN) against uniform resampling (u-GAN), and the proposed $\mathcal{G}$-score for model selection against the validation accuracy criterion.

As Table 5 shows, none of the soft-warping transformations achieves statistically significant differences with respect to the oversampling baseline (row A).

On the other hand, the benefits of using generative models are clear. Both GAN models achieve significant improvements with respect to the oversampling baseline, either using the validation accuracy criterion or the $\mathcal{G}$-score criterion for model selection.

We can also notice that using the γ-resampling can be beneficial in comparison to using the uniform approach. For both data sets, the minimum TSTR classification accuracy corresponds to the u-GAN (E for Catalina and F for ZTF), while the maximum corresponds to the γ-GAN (H for both data sets). Furthermore, for each data set, the best TSTR accuracy is always significantly better than the worst.

Regarding the model selection criteria, the $\mathcal{G}$-score shows to be an effective criterion, achieving accuracies that are at least statistically equivalent to the ones obtained by the computationally expensive validation accuracy criterion. Furthermore, it can sometimes obtain significantly better results, as shown in the ZTF data set by the γ-GAN model.

Interestingly, the combination of the proposed γ-GAN + $\mathcal{G}$-score obtains the best classification accuracies overall, statistically outperforming all existing methods for ZTF data set, and all but one (γ-GAN + val. accuracy) for the case of the Catalina data set.

6. Discussion

6.1. Quality of Generated Samples

Thus far, we have presented a framework for generating realistic light curves that can be used to improve the classification of real astronomical objects. In the entire process, we constantly generate sets of samples that are then compared to the set of real samples, computing global metrics that indicate the quality of the model based on the distance between the sets. However, no metrics to evaluate the quality of individual samples have been mentioned.

In fact, Figure 9 shows that although the generated samples look generally realistic, there can be samples that present artifacts, making them not the best candidates for the classes they intend to represent. While these could be easily solved by applying truncation techniques on latent space of G, it would not be informative about the quality of the individual samples themselves, impeding us from learning what makes a sample look realistic.

The selected metric to evaluate individual sample quality is the *realism score* (Kynkäänniemi et al. 2019), computed over the manifold representation used for the D and R metrics. Given a generated feature sample ϕ_g and a set of real samples $\boldsymbol{\Phi}_r = \{\phi_r\}$, the similarity between ϕ_g and the real manifold Φ_r is calculated as

$$\mathcal{R}(\phi_g, \Phi_r) = \max_{\phi_r \in \boldsymbol{\Phi}_r} \left\{ \frac{\|NND_k(\phi_r)\|_2}{\|\phi_r - \phi_g\|_2} \right\}, \tag{19}$$

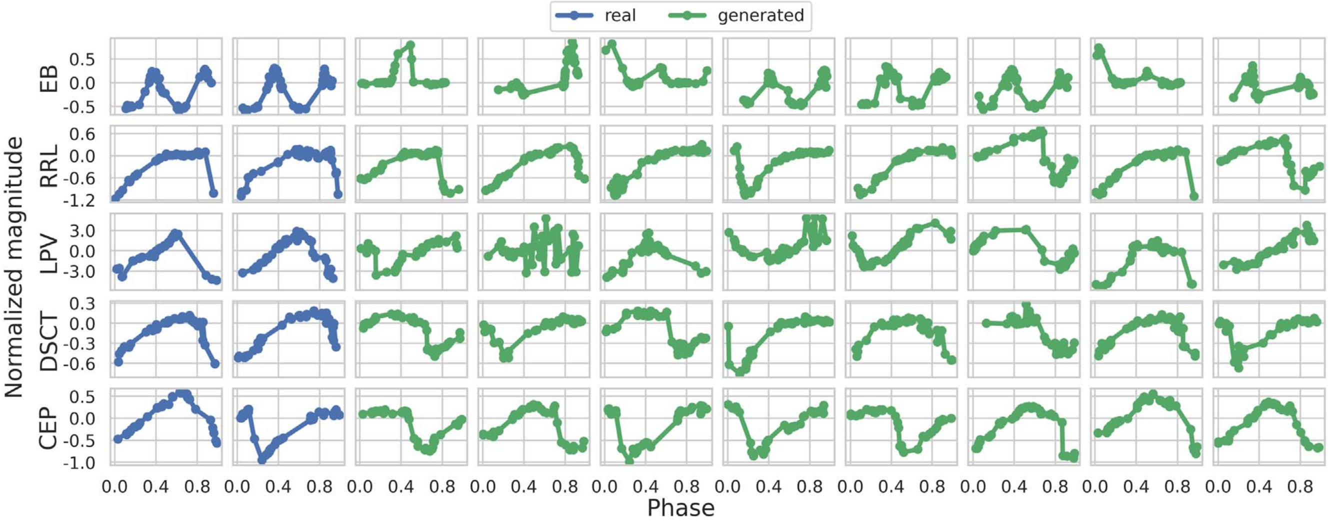


Figure 9. Real and generated light curves of the ZTF data set. To produce the synthetic curves in green, we perform conditional generation with the attributes (phases, class, and amplitude) of the real curves in blue.

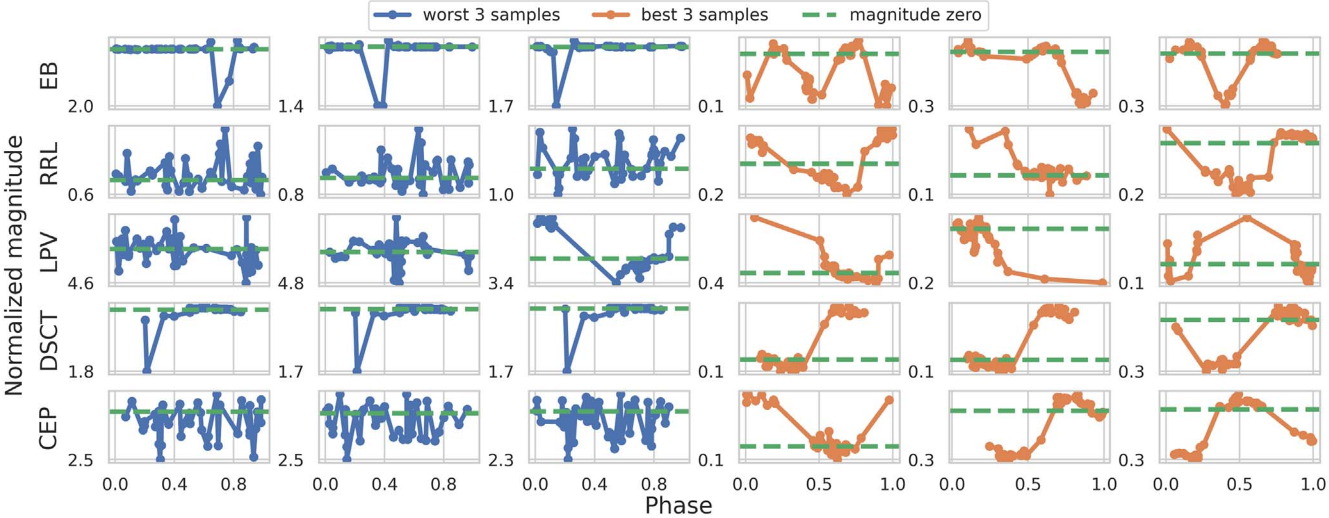


Figure 10. Realism score ranking of the ZTF generated light curves. To rank the samples, we first generated a replication of the $\mathcal{D}^{u}_{\text{train}}$, computed their realism score, and then selected the best and worst samples from the sorted realism scores.

Table 5
Accuracy of the Classification of the Different Augmentation Methods on Test Data Sets

		Method	Catalina					ZTF				
			Accuracy	p-value				Accuracy	p-value			
			[%]	A				[%]	A			
TRTR	A	Oversampling	73.44 ± 1.22					72.61 ± 0.69				
	B	Soft time warping	74.06 ± 1.04	.145				72.69 ± 0.99	.786			
	C	Soft mag warping	73.64 ± 1.79	.723				72.45 ± 0.70	.533			
	D	Soft mixed warping	73.82 ± 1.50	.452				72.53 ± 0.69	.753			
		u-GAN		A	E	F	G		A	E	F	G
TSTR	E	Val Acc	75.97 ± 0.94	<.001				74.17 ± 0.62	<.001			
	F	$\mathcal{G}$-score	76.28 ± 0.74	<.001	.324			73.79 ± 0.50	<.001	1.3.075		
		γ-GAN										
	G	Val Acc	76.86 ± 1.09	<.001	.024	.102		74.37 ± 0.51	<.001	.342	0.003	
	H	$\mathcal{G}$-score	**76.97 ± 0.79**	<.001	.004	.041	.752	**74.94 ± 0.44**	<.001	<.001	<.001	.002

Note. For each method, we report the mean and standard deviation calculated over 15 independent runs. We also report the p-value of the two-sided Welch's tests between each method (rows) and the baselines shown with capital letters in the columns. The method with the highest test classification accuracy's mean is marked in bold for each dataset.

where $NND_k(\phi)$ is the distance from ϕ to its kth nearest neighbor within the corresponding manifold. Equation (19) compares the radii of the KNN induced hyperspheres with center in ϕ_r to the distance between ϕ_r and the sample ϕ_g. Naturally, if ϕ_g does not belong to any of the hyperspheres, $\mathcal{R}$ will be low, and its value will increase the closer ϕ_g is to any ϕ_r.

The effect of ranking the generated samples of the ZTF data set by *realism score* is shown in Figure 10.

Because it can successfully identify artifacts that could be filtered out of the data set, we would in principle expect that using a *realism score* filtering would improve our results even further. However, this is not the case. Empirically, we found no statistical differences when applying this filtering to our generated data sets. We hypothesize that these artifacts, although undesirable, are not crucial when defining the decision boundaries of the problem, hence, they have little impact on the classification accuracy. Moreover, strongly filtered data sets cause a drop in the classification accuracy, probably caused by their over-constrained diversity.

6.2. Classification Results

Soft-warping transformations—Regarding the effects of the proposed soft-warping augmentations for classification, we can see that despite the fact that they create plausible light curves, they do not show improvements in the classification task. We hypothesize that the diversity added to the data set by these transformations is not substantial enough for the classifiers to benefit from it.

γ-resampling—The results suggest that the proposed resampling offers a clear improvement upon uniform resampling for GAN training. We believe that this improvement comes from the delay in the GAN overfitting, providing more potentially good models to choose from before the GAN completely overfits. With respect to the no-resampling model, Figure 4 shows that models trained $\gamma = 0$ and $\gamma = 0.25$ reach comparable accuracies, consistently with the fact that the resampling block does not add any extra information. Using the resampling block can offer a more stable training that reaches similar performance in a shorter training time. This can be particularly relevant if the defined iteration horizon is not long enough to capture the peak accuracy as in Figure 4(b). For this reason, we do not think that γ should be tuned thoroughly, and we set it to $\gamma = 0.25$, placing the $\mathcal{G}$-score peak within the extent of training iterations, earlier than the peak of $\gamma = 0$ but later than that of $\gamma = 1$.

$\mathcal{G}$-score—For the model selection criterion, the correlation between the metrics and the classification results validate the $\mathcal{G}$-score as a metric to evaluate the quality of the generated samples. Using this metric instead of the validation accuracy, it is interesting because of the subtle improvements in TSTR. It also offers faster computation times: computing $\mathcal{G}$-score is approximately 6 times faster than computing the validation TSTR accuracy.

We hypothesize that these subtle improvements come from the robustness of the G-score against overfitting. While the $\mathcal{G}$-score compares $\mathcal{D}^u_{\text{gen}}$ to the entire training set $\mathcal{D}_{\text{train}}$, the validation accuracy score is computed on the small data set $\mathcal{D}_{\text{val}}$ for evident reasons. Hence, it is more susceptible to overfitting. A fact that reinforces this hypothesis is the consistently lower variance of the models selected with the $\mathcal{G}$-score criterion compared to validation accuracy. On the other hand, computing the $\mathcal{G}$-score also has some drawbacks related to the normalization step restrictions. Since the normalization requires the minimum and maximum value of the D and R metrics, we cannot compute the $\mathcal{G}$-score during the training time, and we must first completely train the models. In addition to this, it only allows for comparison between different candidates of the same run, not permitting comparisons between different runs that likely have different normalization parameters.

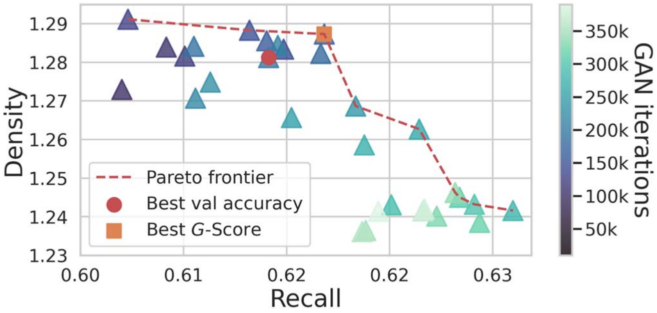


Figure 11. Macro density and recall metrics for the Catalina data set. Each point corresponds to the average of 5 independent computation of density and recall for a single GAN model.

6.3. Alternative to the $\mathcal{G}$-score

Evaluating generative models by fidelity and diversity can be posed as a multi-objective problem. Thus, we provide an alternative to the $\mathcal{G}$-score that considers both objectives (D and R) simultaneously, according to the problem's nature.

As an alternative to evaluate all candidates with TSTR validation accuracy, we propose evaluating only candidates that lie on the *Pareto frontier*[5] of the raw macro density and macro-recall. For example, in the case of the Catalina data set, doing so would imply evaluating approximately 1/4 of total candidates.

The disposition of the optima for the Catalina data set is shown in Figure 11. Interestingly, the model selected with the validation accuracy criterion is in the suboptimal region which supports the idea of overfitting explained in Section 6.2. On the other hand, the model selected with $\mathcal{G}$-score belongs to the Pareto frontier, which is not necessarily guaranteed considering the extra normalization step included in the computation of the $\mathcal{G}$-score.

Using this alternative offers an attractive advantage. Not performing the normalization step of the $\mathcal{G}$-score allows for comparing different GAN setups in the DR plane, which could also be used to perform hyperparameter optimization of the models. In this scenario, we first need to identify the models that lie in the Pareto frontier considering all the D and R scores and then evaluate these candidates based on the validation TSTR score to choose an operating point.

7. Conclusions

In this work, we have presented a GAN-based data augmentation methodology for astronomical time series, to improve the classification accuracy of periodic variable stars by mitigating the problems of small and imbalanced astronomical data sets.

[5] In multi-objective optimization, the Pareto frontier is the set of all the Pareto optimal solutions. A Pareto optimal solution is defined as a solution that cannot be improved in any individual objective without worsening others.

Using our methodology, we can generate diverse synthetic data sets of irregularly sampled time series that capture the original training sets' properties and leverage their diversity to outperform classifiers trained on real data. Motivated by the rapid overfitting of our generative model in this unbalanced setup, we propose a resampling technique (γ-resampling) to mitigate this behavior. Also, inspired by the incapability of FID to measure this overfitting, we propose a novel evaluation metric ($\mathcal{G}$-score) that correlates with TSTR classification accuracy; hence it helps select a generative model among the possible candidates saved during training.

The proposed model could be extended to work with classifiers that are currently operating in real-time such as the ALeRCE light-curve classifier (Sánchez-Sáez et al. 2021), boosting its performance on the ZTF stream and eventually on its successor, the Vera C. Rubin Observatory Legacy Survey of Space and Time (LSST; Zeljko et al. 2019), contributing to understanding the tridimensional structure and formation of our galaxy and its neighbors.

7.1. Future Work

Although effective in this simplified setup, the presented methodology could be improved by upgrading it to a scenario where the input data has a variable length. This upgrade should involve recent GAN models that include recurrent neural networks in their architectures, such as Yoon et al. (2019) or Ni et al. (2020). In addition, the generation of data with variable length should also be addressed.

Regarding conditional generation, we used the class-conditional parameter to generate data sets with uniform class distributions. Although our model permits other conditional parameters such as amplitude, in all experiments we replicated the distribution of their real counterparts. An interesting extension of the work could include analyzing how the results vary depending on the generated conditional distribution of these parameters, and other physical parameters that may be relevant to include.

Finally, all our synthetic data sets were generated by sampling z from a multivariate Gaussian distribution related to data samples generation. Evaluating different sampling methods, such as those presented in Kynkäänniemi et al. (2019), and inspecting how they affect the qualitative and quantitative results, could be an exciting path to follow.

The authors acknowledge support from the National Agency of Research and Developments Millennium Science Initiative through grant IC12009, awarded to the Millennium Institute of Astrophysics (G.G., P.E.) and from the National Agency for Research and Development (ANID) grants: FONDECYT Regular #1220829 (P.E.), and Magister Nacional/2019-22190949 (G.G.). This work was funded in part by the Institute for Applied Computational Science (IACS), Harvard University (P.P.).

ORCID iDs

Germán García-Jara https://orcid.org/0000-0001-8202-9314
Pavlos Protopapas https://orcid.org/0000-0002-8178-8463
Pablo A. Estévez https://orcid.org/0000-0001-9164-4722

References

Arjovsky, M., Chintala, S., & Bottou, L. 2017, arXiv:1701.07875

Bellm, E. C., Kulkarni, S. R., Graham, M. J., et al. 2018, PASP, 131, 018002

Bińkowski, M., Sutherland, D. J., Arbel, M., & Gretton, A. 2018, arXiv:1801.01401

Brock, A., Donahue, J., & Simonyan, K. 2019, arXiv:1809.11096

Carrasco-Davis, R., Reyes, E., Valenzuela, C., et al. 2021, AJ, 162, 231

Caruana, R. 2000, in Proc. Am. Assoc. for Artificial Intelligence (AAAI) Conf. (Palo Alto, CA: AAAI), 51, https://aaai.org/Papers/Workshops/2000/WS-00-05/WS00-05-011.pdf

Drake, A. J., Graham, M. J., Djorgovski, S. G., et al. 2014, ApJS, 213, 9

Esteban, C., Hyland, S. L., & Rätsch, G. 2017, arXiv:1706.02633

Frid-Adar, M., Diamant, I., Klang, E., et al. 2018, Neurocomputing, 321, 321

Förster, F., Cabrera-Vives, G., Castillo-Navarrete, E., et al. 2021, AJ, 161, 242

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., et al. 2014, arXiv:1406.2661

Guennec, A. L., Malinowski, S., & Tavenard, R. 2016, in ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data, https://halshs.archives-ouvertes.fr/halshs-01357973

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. 2017, arXiv:1704.00028

He, H., & Garcia, E. 2009, IEEE Transactions on Knowledge and Data Engineering, 21, 1263

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. 2017, arXiv:1706.08500

Huang, Y., Jin, Y., Li, Y., & Lin, Z. 2020, IEEE Access, 8, 88399

Iwana, B. K., & Uchida, S. 2021, PLoSO, 16, e0254841

Karras, T., Aila, T., Laine, S., & Lehtinen, J. 2017, arXiv:1710.10196

Karras, T., Aittala, M., Hellsten, J., et al. 2020, arXiv:2006.06676

Karras, T., Laine, S., & Aila, T. 2018, arXiv:1812.04948

Karras, T., Laine, S., Aittala, M., et al. 2019, arXiv:1912.04958

Kingma, D. P., & Ba, J. 2015, arXiv:1412.6980

Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J., & Aila, T. 2019, arXiv:1904.06991

Mirza, M., & Osindero, S. 2014, arXiv:1411.1784

Miyato, T., Kataoka, T., Koyama, M., & Yoshida, Y. 2018, arXiv:1802.05957

Mogren, O. 2016, arXiv:1611.09904

Naeem, M. F., Oh, S. J., Uh, Y., Choi, Y., & Yoo, J. 2020, arXiv:2002.09797

Ni, H., Szpruch, L., Sabate-Vidales, M., Xiao, B., Magnus, W., et al. 2021, in Proc. Second ACM Int. Conf. AI Finance (New York, NY: ACM)

Odena, A., Olah, C., & Shlens, J. 2016, arXiv:1610.09585

Opitz, J., & Burst, S. 2019, arXiv:1911.03347

Pimentel, O., Förster, F., & Estévez, P. A. 2022, arXiv:2201.08482

Ramponi, G., Protopapas, P., Brambilla, M., & Janssen, R. 2018, arXiv:1811.08295

Ravuri, S., & Vinyals, O. 2019, arXiv:1905.10887

Sajjadi, M. S. M., Bachem, O., Lucic, M., Bousquet, O., & Gelly, S. 2018, arXiv:1806.00035

Salehinejad, H., Valaee, S., Dowdell, T., Colak, E., & Barfett, J. 2018, in IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP) (New York, NY: IEEE Press), 990

Salimans, T., Goodfellow, I., Zaremba, W., et al. 2016, in Proc. 30th Int. Conf. on Neural Information Processing Systems (Red Hook, NY: Curran), 2234, https://proceedings.neurips.cc/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf

Sampath, V., Maurtua, I., Martín, J. J. A., & Gutierrez, A. 2021, J. Big Data, 8, 1

Sánchez-Sáez, P., Reyes, I., Valenzuela, C., et al. 2021, AJ, 161, 141

Santurkar, S., Schmidt, L., & Madry, A. 2017, arXiv:1711.00970

Shmelkov, K., Schmid, C., & Alahari, K. 2018, arXiv:1807.09499

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. 2016, in IEEE Conf. Computer Vision Pattern Recognition (CVPR) (New York, NY: IEEE), 2818

Wang, Z., She, Q., & Ward, T. E. 2022, ACM Computing Surveys, 54, 1

Wen, Q., Sun, L., Yang, F., et al. 2021, in Proc. 30th Joint Conf. Artificial Intelligence (IJCAI), ed. Z. Zhou (IJCAI), 4653

Yang, J., Kannan, A., Batra, D., & Parikh, D. 2017, arXiv:1703.01560

Yazici, Y., Foo, C.-S., Winkler, S., et al. 2019, arXiv:1806.04498

Yoon, J., Jarrett, D., & van der Schaar, M. 2019, in Advances in Neural Information Processing Systems, 32, ed. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox et al. (Red Hook, NY: Curran Associates, Inc.), https://proceedings.neurips.cc/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf

Željko, I., Kahn, S. M., Tyson, J. A., et al. 2019, ApJ, 873, 111

Zhang, Z., Han, J., Qian, K., et al. 2020, IEEE J. Biomedical and Health Informatics, 24, 300

Zhu, X., Liu, Y., Li, J., Wan, T., & Qin, Z. 2018, in Pacific-Asia Conf. Knowledge Discovery Data Mining, ed. D. Phung, V. Tseng, G. Webb, B. Ho, & M. Ganji (Cham: Springer), 349