UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

IMPROVING ASTRONOMICAL TIME-SERIES CLASSIFICATION VIA DATA AUGMENTATION WITH GENERATIVE ADVERSARIAL NETWORKS

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

GERMÁN EDUARDO GARCÍA JARA

PROFESOR GUÍA:
PABLO ESTÉVEZ VALENCIA

PROFESOR CO-GUÍA:
PAVLOS PROTOPAPAS

MIEMBROS DE LA COMISIÓN:
PABLO ZEGERS FERNÁNDEZ
CLAUDIO PÉREZ FLORES

SANTIAGO DE CHILE
2022

RESUMEN DE LA TESIS PARA OPTAR AL
GRADO DE MAGÍSTER EN CIENCIAS DE LA
INGENIERÍA, MENCIÓN ELÉCTRICA
Y DE LA MEMORIA PARA OPTAR AL
TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: GERMÁN EDUARDO GARCÍA JARA
FECHA: 2022
PROF. GUÍA: PABLO ESTÉVEZ VALENCIA

MEJORANDO LA CLASIFICACIÓN DE SERIES DE TIEMPO ASTRONÓMICAS MEDIANTE REDES GENERATIVAS ADVERSARIAS

Debido a los últimos avances de la tecnología, telescopios que cubren amplias zonas del cielo producirán millones de alertas astronómicas por noche que deberán ser clasificadas de forma rápida y automática. Actualmente, la clasificación consiste en algoritmos supervisados de aprendizaje automático cuyo rendimiento está limitado por la cantidad de anotaciones existentes de objetos astronómicos y sus distribuciones de clases altamente desequilibradas. En esta tesis, proponemos una metodología de aumento de datos basada en Redes Generativas Adversarias (GAN, por sus siglas en inglés) para generar una variedad de curvas de luz sintéticas a partir de estrellas variables. Nuestras novedosas contribuciones, que consisten en una técnica de remuestreo y una métrica de evaluación, nos permiten evaluar la calidad de los modelos generativos en conjuntos de datos desequilibrados e identificar casos de sobreajuste de GAN que la distancia de Fréchet no revela. El modelo propuesto es evaluado en dos conjuntos de datos tomados de los *surveys* de Catalina y Zwicky Transient Facility. Las métricas de *accuracy* de clasificación de estrellas variables mejoran significativamente cuando los clasificadores se entrenan con datos sintéticos y reales con respecto al caso de usar solo datos reales.

RESUMEN DE LA TESIS PARA OPTAR AL
GRADO DE MAGÍSTER EN CIENCIAS DE LA
INGENIERÍA, MENCIÓN ELÉCTRICA
Y DE LA MEMORIA PARA OPTAR AL
TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: GERMÁN EDUARDO GARCÍA JARA
FECHA: 2022
PROF. GUÍA: PABLO ESTÉVEZ VALENCIA

IMPROVING ASTRONOMICAL TIME-SERIES CLASSIFICATION VIA DATA AUGMENTATION WITH GENERATIVE ADVERSARIAL NETWORKS

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 thesis, 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, allow us to assess the quality of generative models in unbalanced datasets and identify GAN-overfitting cases that the Fréchet Inception Distance does not reveal. We evaluate our proposed model using two datasets taken from the Catalina and Zwicky Transient Facility surveys. The classification accuracy of variable stars is improved significantly when adding synthetic data to our training sets with respect to the case of using only real data.

A mi familia

Agradecimientos

No podría partir sin primero agradecer a mi familia, quienes han estado de cerca en esta travesía, y a quienes debo en gran parte la persona que soy hoy. A mis padres, por su ilimitado amor y respaldo, y por darme el privilegio de sentirme orgulloso teniéndolos como padres. Este documento es para ellos, y espero con él retribuir en alguna medida ese orgullo. A mi madre, por ser mi principal soporte a lo largo de este viaje, por su objetividad y mente fría en las extensas conversaciones, y también por su comprensión en las no tan extensas. A mi padre, por su incondicional presencia y apoyo a pesar de la distancia, y por brindarme siempre una cuota de optimismo en los momentos que éste no abundaba. A mi segunda madre, Irma Ávila, por ser el pilar mío y de mi familia desde que tengo memoria, por la infinita dedicación, preocupación, cuidado y amor entregado. A mis hermanas, por la complicidad a lo largo de este proceso, extra gracias por la paciencia y aguante en las últimas etapas de cuarentena. Finalmente, muchas gracias a mis abuelos Teresa y Fernando, la alegría y satisfacción que manifestaban por cada uno de mis pequeños logros siempre fue una muy linda motivación.

Al profesor Pablo, principal responsable de que esto haya sucedido, por guiar esto desde nuestro primer ramo de especialidad juntos y por recibirme en su laboratorio con los brazos abiertos, permitiendome investigar un tema de mi interés. Por todos los recursos entregados, las numerosas conversaciones y retroalimentaciones, muchas gracias. A Pavlos, por creer en mis capacidades desde el comienzo, por hacerme creer de lo que era capaz, por abrirme las puertas del IACS y con ello un sin fin de experiencias y nuevas oportunidades, y por toda la dedicación entregada a este trabajo a través de extensas discusiones. A Francisco y Guillermo, por permitir en primer lugar que esta colaboración haya sido posible, haciendome partícipe de la colaboración Harvard-Chile, y del proyecto ALeRCE. A Rodrigo e Ignacio, por toda la ayuda brindada durante mi estadía en el laboratorio.

A las personas que por elección me han acompañado durante estos años. A Nicolás y Esteban, con quienes compartimos esta loca idea de hacer un Magister. Por todas las mañanas, tardes y noches de trabajo, tener la oportunidad de aprender de ustedes y con ustedes ha sido un real placer. A Pablo, por ser mi gran compañero en el paso por el pregrado, por todos los trasnoches y vivencias. A Camilo y Ricardo, por todas las anécdotas y buenos momentos. A Sergio E., Freddy y Mario, por asegurarse que algún día terminara esta tesis, por entender los numerosos "no puedo, tengo tesis", y por todas las conversaciones y risas. A mis amigos del colegio, los de siempre, que han crecido conmigo y estado para mí en esta amistad desde hace más de 18 años; Rocio, Juan, Antonio e Isidora. A mi primer mejor amigo, Sergio S., por su compañía en esta última etapa, volviendo a cultivar una bella amistad que se mantiene intacta a pesar de los años. Mil gracias a todos.

Agradezo también al apoyo de la Iniciativa Científica Milenio de la Agencia Nacional de Investigación y Desarrollo (ANID) a través de la beca IC12009, otorgada al Instituto Milenio de Astrofísica, a las becas de la Agencia Nacional de Investigación y Desarrollo: FONDECYT Regular #1220829, y Magíster Nacional/2019-22190949, y al *Institute for Applied Computational Science* (IACS) de la universidad de Harvard.

Table of Content

Chapter 1

Introduction

1.1 Motivation

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 that allow the model to generalize and perform well when testing on new and unseen data. To avoid overfitting, models that achieve state-of-the-art results in different tasks are trained with annotated datasets that have been extensively processed and filtered, and that consist of a large number of samples for each class.

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 datasets, but the annotations have also highly imbalanced class distributions. While small datasets already hinder learning by making algorithms fail at generalizing characteristics of the data, imbalanced distributions only accentuate this issue ([1]; [2]). 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 datasets into large and balanced datasets. 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 [3].

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 similar properties to the existing ones. Moreover, appropriate augmentation techniques are specific to the dataset [4] and the task [3]. An example of a dataset-specific technique could be jittering, where additive Gaussian noise is often used in sensor datasets. 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 class information of the samples.

A generative model for data augmentation permits 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. In this way, the model could 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 [5], Generative Adversarial Networks (GANs; [6]) 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 ([7]; [8]; [9]; [10]) and only a few in the time domain ([11]; [12]). Furthermore, [11] 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 thesis, we propose a GAN-based data augmentation methodology for time-series to improve the classification accuracy on two astronomical datasets taken from the Catalina and Zwicky Transient Facility surveys. The main contributions are the following:

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

1.2 Hypotheses

The hypotheses of this thesis are:

H1 It is possible to use GAN models to learn the data distribution of periodic light curves and generate realistic synthetic light curves that preserve basic properties of the original ones, such as class and amplitude.

H2 We hypothesize that using the generated synthetic light curves to train a classifier would improve the classification accuracy of variable stars.

1.3 General Objective

The general objective of this thesis is to propose and validate a GAN-based data augmentation methodology for irregularly-sampled time series that allows both multi-class and physical parameter conditional generation, which can be used to improve the classification accuracy of periodic light curves in astronomy.

1.4 Specific Objectives

The specific objectives of this thesis are:

O1 Design and implement a GAN model capable of dealing with irregularly-spaced time series and performing both class and parameter conditional generation.

O2 Propose a criterion for evaluating generative models that permits their use in a downstream task such as classification.

O3 Design and implement a light curve classifier based on neural networks that will be used as the baseline for the experiments.

O4 Compare the performance of the classifiers trained on generated synthetic and real data.

O5 Evaluate the proposed model for data augmentation and compare it with other classic techniques using the data from two astronomical surveys.

1.5 Structure of the document

The structure of this thesis is as follows: Chapter 2 presents a theoretical background that explains the foundations of the work. Section 2.1 describes the astronomy-related relevant concepts, and Section 2.2 the machine learning core concepts. In Chapter 3, Sections 3.1 and 3.2 describe the utilized datasets and their pre-processing procedure. Section 3.3 extensively explains the proposed methodology. In Chapter 4, Section 4.1 presents the obtained results, which are discussed in Section 4.2, stating its strengths and weaknesses. Finally, Chapter 5 presents the main conclusions of this work and future steps.

Chapter 2

Background

This chapter provides a theoretical background covering the fundamental concepts for understanding this thesis. First, an astronomy-related background is provided, including the current big data era of astronomy, a general astronomical glossary, a description of the astronomical objects of interest for this work, and the importance of accurate classification of astronomical objects. After that, a machine learning background is provided, emphasizing Generative Adversarial Networks, the core idea of this thesis.

2.1 Astronomical Background

Astronomy is a science motivated by the curiosity of the human to understand what is above us. This curiosity can be traced back the ancient civilizations. For example, Babylonians maintained a detailed record of the movement of the sun and moon. They are also believed to have documented the Halley's comet for the first time in 164 B.C. and for being the first ones in dividing the sky into different zones.

However, the scenario has drastically changed since then. Nowadays, far from analyzing the universe with their naked eyes, astronomers employ modern telescopes to capture electromagnetic radiation from the surrounding universe. These telescopes are generally placed in high-altitude zones with stable atmospheric conditions for observation to overcome the effect of atmospheric turbulence. Furthermore, there are also space telescopes that can observe the universe free from the obstruction of atmosphere, such as the launched James Webb Space Telescope [13]. Independent of the telescope's location, the study, and analysis of data captured allows astronomers to understand better the behavior of the celestial objects present in the universe.

With the latest advances in technology, computation, software, and optics fabrication, telescopes with extensive sky coverage will produce millions of astronomical alerts per night. These types of telescopes perform astronomical surveys and have given rise to the big data era of astronomy.

2.1.1 The Big Data Era

An astronomical survey aims to generate a general map of broad sky regions without focusing on specific observational targets. This mapping is done following a pre-defined observation plan that specifies which parts of the sky and spectrum frequencies will be observed at certain times. After performing multiple scans, the maps can be compared to find significant changes in luminosity between images of the same region, originating an alert. Examples of astronomical surveys are HiTS [14], Catalina [15], MACHO [16], ASASN [17] and ATLAS [18].

Two essential surveys that unequivocally link astronomy to big data are the Zwicky Transient Facility (ZTF; [19]) and its successor, the Vera C. Rubin Observatory Legacy Survey of Space and Time (LSST; [20]). These two surveys were designed to capture large portions of the sky with relatively low periods between observations (high cadence), generating massive amounts of data.

On the one hand, ZTF has been operating since 2019, observing the entire northern sky across three frequency bands and producing approximately one million alerts per night. On the other hand, LSST is expected to start operating in 2024, observing the entire visible sky from the southern hemisphere across six different frequency bands and generating around ten million alerts per night.

The large amounts of alerts provided by these telescopes are supported by the volume of the universe that the telescopes can observe in a single exposure. This volume is measured by the etendue of the telescope, computed as the product between the area of the sky observed by the telescope in a single exposure (field of view) and the telescope's diameter (light-collecting area). Figure 2.1 shows the etendue of different telescopes as the area of the corresponding circles in the field of view/light-collecting area plane, where the large volume of the universe that the LSST covers clearly stands out compared to other telescopes.

To enable further analysis in follow-up telescopes, the alerts are aggregated, annotated, and classified into different types by astronomical alert brokers that are designed to provide a rapid and self-consistent classification, extracting and reporting relevant information about the alerts. Examples of astronomical brokers are the Automatic Learning for the Rapid Classification of Events(ALeRCE; [21]), The Arizona-NOAO Temporal Analysis and Response to Events System (ANTARES; [22]), and LASAIR [23].

The processing done by the astronomical brokers needs to be rapid, considering the large number of alerts emitted every night. It also needs to be accurate since observation time is limited and performing follow-up observations on objects that are not of scientific interest is undesirable. Commonly, the classification consists of supervised machine learning algorithms whose performance is limited by the number of existing annotations of astronomical objects. These annotations are usually obtained either using the knowledge of experts or by cross-matching the observed objects with previously annotated catalogs that have observed the same regions of the sky.

Despite dealing with vast amounts of data coming from the telescopes, the labeled data available to be used as training sets for the classifiers are not abundant. On top of that, the

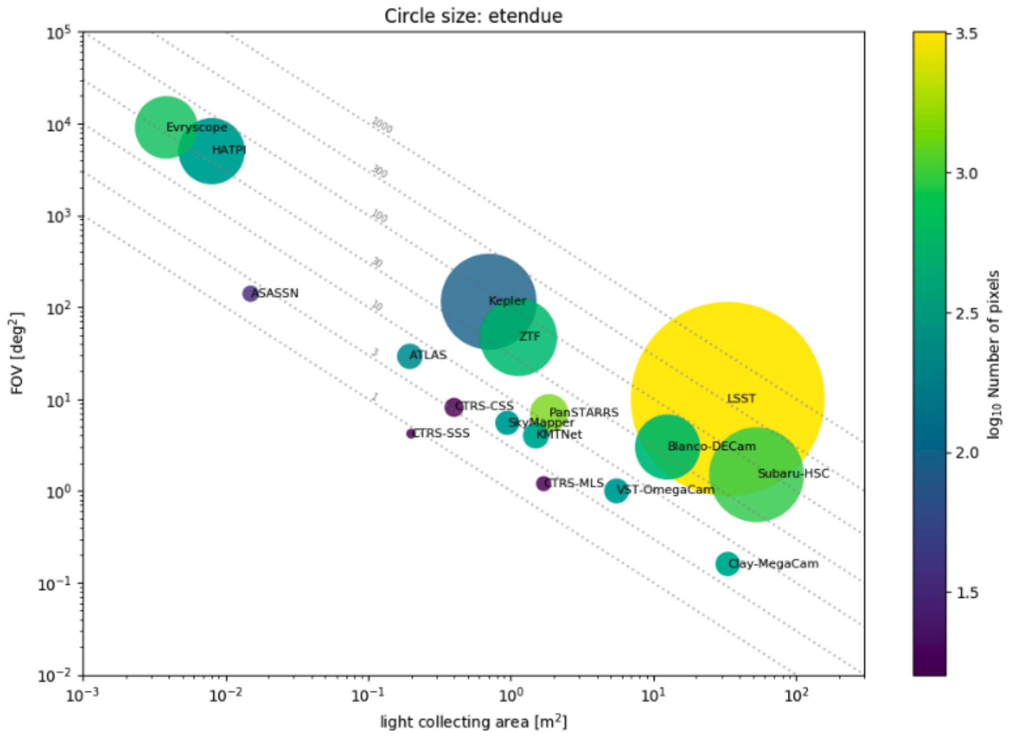


Figure 2.1: Field of view (FOV) in [deg^2] vs light collecting area in [m^2] for different telescopes. The product of these two quantities (displayed as the area of the circles) is known as etendue, which measures the volume observed in a single exposure by a telescope. The figure was obtained from [21].

distribution of the annotated astronomical datasets usually is very distant from being uniform and presents a big imbalance, placing an extra difficulty when trying to learn inductive rules with standard machine learning algorithms. The urge for accurate classification that demands large and diverse annotated datasets motivates our data augmentation framework to mitigate the problem of data imbalance and data scarcity.

2.1.2 Astronomical Glossary

This section provides definitions for the core concepts related to the astronomical background of this thesis.

Astronomical object

Often used interchangeably with astronomical body, an astronomical object is a natural phenomenon or physical body that occurs or exists in the universe, emits its own light or reflects that of other objects, and can potentially be observed with 'telescopes with enough resolution and light gathering. Examples of astronomical objects are galaxies, stars, asteroids, and planets.

Irregular sampling

Unlike most phenomena that can be represented as time-series, astronomical time-series (light curves) are irregularly sampled, i.e., the time intervals between two consecutive observations are not constant, producing variable time gaps between observations that can be in the order of seconds or the order of months. These irregularities happen because telescopes cannot observe the same source at constant time intervals due to the survey's general science objectives or changes in atmospheric conditions.

Charged-coupled Device

One of the critical components of a telescope, charge-coupled devices (CCDs) are a widely used technology for light detection in scientific and non-scientific applications. It is a highly sensitive detector composed of multiple light-sensitive sensors. Photons that have traveled through the atmosphere, lens, filters, and mirror hit the detectors and are translated to electronic charges proportional to the intensity of the light received. These charges are then converted into a digital image (matrix of integer numbers) in Analog to Digital Units (ADUs) or counts that resemble the sky's observed regions. An extensive description of these processes can be found in [24].

Flux

After the CCD acquires the digital images, the counts associated with the detection are aggregated. Considering additional information such as time of exposure, gain of the instruments, and observing conditions, the total counts can be transformed into flux, which accounts for the energy of a detected object per time unit, divided by the light-collecting area.

Magnitude

Once the flux has been obtained, it is possible to measure the perceived brightness of the object registered by a detector placed on earth by computing the object's magnitude m according to:

$$m = -2.5 \log_{10} \left(\frac{F_s}{F_{ref}} \right) \tag{2.1}$$

where F_s is the previously explained flux of the object, and F_{ref} corresponds to a reference stellar object's flux with a known magnitude measured by a different instrument, used to calibrate the observations and take into account absorption and color diffraction phenomena produced by the atmosphere.

It is worth mentioning that equation 2.1 implies that magnitudes have a reverse logarithmic scale, i.e., lower magnitudes imply brighter objects. According to this, plots that include magnitudes in the y-axis are often displayed inverted y-axis. Another interesting consequence of this scale is that differences of 1.0 in magnitude should imply a factor of ~ 2.5 in brightness or flux.

In general and unless stated otherwise, the word magnitude refers to *apparent* magnitude, which is how the intrinsic luminosity of the stellar objects is perceived from the earth. Distance from the telescope to the object, interactions of the object's radiation with the atmosphere, and possible absorption and scattering of its radiation by stellar dust can affect the apparent magnitude's value. Its counterpart, the *absolute* magnitude, is not affected by these phenomena. It plays an important role when estimating distances to different stellar objects whose absolute magnitude can be derived from other properties such as their periodic behavior.

Variable star

A variable star is a star whose magnitude fluctuates over time. The causes for its variation can be intrinsic if the star suffers changes in its physical properties that alter its luminosity or extrinsic if the start is obstructed by others, limiting the amount of light that the telescopes can perceive.

Light curves

A light curve is a common way of characterizing astronomical objects. It is an irregularly-sampled time-series that displays the evolution of an astronomical object's brightness with respect to time. The magnitude represents the brightness of a stellar object. The observation times are commonly represented by their Modified Julian Date (MJD), i.e., the number of days since a reference start date that corresponds to midnight of November 17, 1858, Universal Time (UT). Astronomical surveys can provide observations in multiple frequency bands, in which case a single stellar object can be associated with more than one light curve.

2.1.3 Astronomical Objects of Interest

In this section, we describe the astronomical objects relevant to this work. The wide variety of existing astronomical objects has been grouped by ALeRCE [21] into a hierarchical taxonomy that can be seen in Figure 2.2. This taxonomy is organized in a hierarchical structure following the light curve variability and their different physical properties. This taxonomy intends to unify the existing works that perform the classification of astronomical light curves, and it also allows for the addition of new subgroups as the quality of the data grows [25].

Considering that each additional class that we include in our experiments increases the complexity of the problem, the scope of this thesis is limited to periodic variable stars. Working with periodic stars is also convenient because they usually present recognizable patterns when mapped to their phase space, a process that is detailed in Section 2.1.3.

Periodic objects

An astronomical object is called periodic if its brightness over time presents a repeated pattern. In this thesis, we consider a subset of periodic variable stars that is composed by:

- **Eclipsing Binaries**: extrinsic variable stars composed by a pair of stars that orbit around each other. The magnitude captured with the telescopes suffers fluctuations when the orbit plane and the line of sight are aligned, reducing the object's magnitude when one of the stars obstructs the line of sight of its binary companion.

- **RR Lyrae**: intrinsic variable stars characterized by their short periods ranging from a couple of hours to a day and magnitudes ranging from 0.3 to 2. They appear in very old

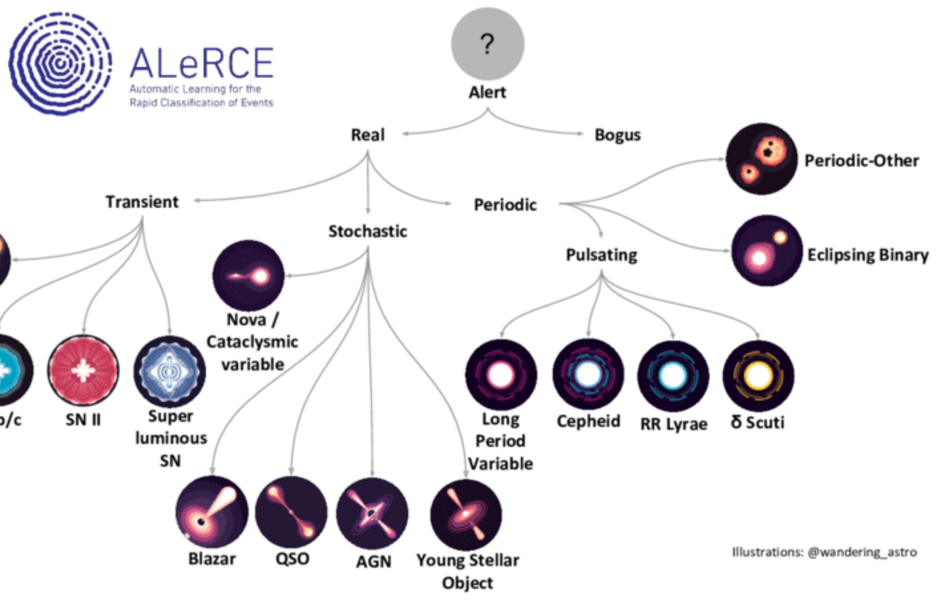


Figure 2.2: Alerce Taxonomy. The figure was obtained from [21].

stellar populations in the galactic halos and thick disks of the galaxy and have typical ages of around 10 billion years.

- **Long Period Variables**: intrinsic variable stars that are distinguishable for their extended periods ranging from weeks to several years.
- **Cepheids**: intrinsic variable stars that can be found further than RR Lyraes due to their larger brightness. Their typical periods can range from 1 day to over 200 days, and they live in the thin disk of the galaxy along with the young stellar population.
- **Delta Scuti**: intrinsic variable stars, fainter than Cepheids and with shorter pulsation periods, but they follow the same period-luminosity relation, and the boundary period to distinguish both classes is a matter of convention.

Period folding

Since the desired characteristic shapes of periodic light curves are only visible in the phase space, we map the observation times into this bounded space using the period information, an operation known as folding. Denoting the light curve period as T and the observation time as t, the folding procedure is performed by converting t into ϕ_t according to:

$$\phi_T \equiv t \pmod T \tag{2.2}$$
$$\phi_t = \frac{\phi_T}{T} \tag{2.3}$$

where the congruence symbol $\equiv$ in Equation 2.2 refers to the extension of the modulo operator with modulus T to real numbers. An illustration of this process for a RR Lyrae observed by

ZTF is shown in Figure 2.3.

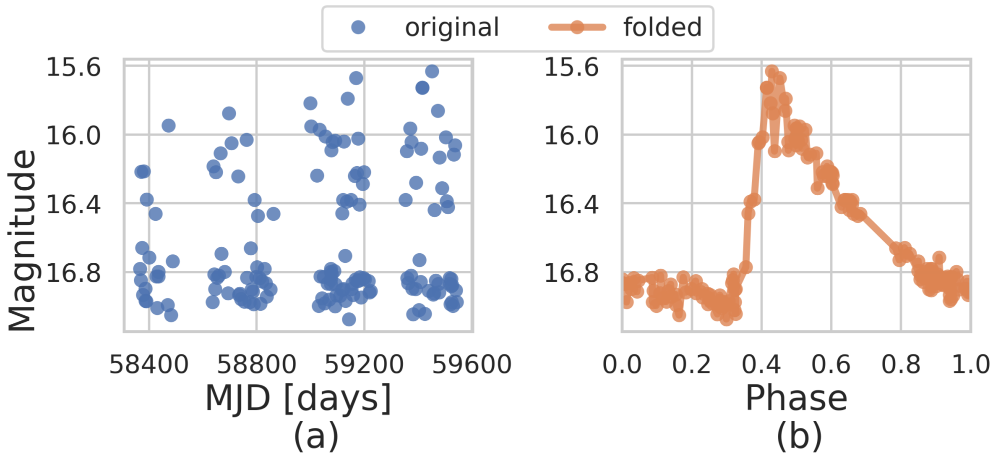


Figure 2.3: (a) Original RR Lyrae visualized in time space. (b) Period-folded RR Lyrae visualized in phase space.

2.1.4 Importance of the Classification of Astronomical Objects

Quickly and accurately classifying astronomical alerts provides an immense scientific value. For example, the orbit of Eclipsing Binary systems can be used to estimate their mass, density, luminosity, and distance. The period-luminosity relation of RR Lyrae permits using them as standard candles to measure distances to old stellar systems living in the halo and thick disk and acquire knowledge about the earliest history of our galaxy. Finally, Cepheids can also be used as distance indicators to nearby galaxies such as the Magellanic Clouds and the Andromeda Galaxy. All of this reinforces that accurate classification of these astronomical objects could help better understand the structure and formation of our galaxy and its neighbors.

2.2 Machine Learning Background

Broadly speaking, a machine learning algorithm is an algorithm that is able to *learn* from data. An algorithm is said to *learn* from data if its performance at the desired task improves as the algorithm is exposed to more data [26].

2.2.1 Datasets

In general, the data available for the algorithms to learn is organized in a group called *dataset*, which should be understood as a collection of examples used for training. Finally, a training example is a collection of *features* that we want the machine learning algorithm to process. Even though the term *feature* is usually understood as the result of complex operations applied to the raw data, the data itself could be the feature of interest. Examples of this are pixel values of an image, voltage measured in medical signals at a specific time, or stock price in a given day for a time-series.

In this work, we use the terms *sample* and *example* interchangeably, understanding that training examples are realizations of a sampling process coming from an underlying data distribution, and embracing their stochastic nature.

Usually, datasets used for studying and bench-marking machine learning algorithms are designed with desirable properties that facilitate working with them. Apart from significant pre-processing and filtering steps, these datasets are collected (or generated) to have an approximately uniform class distribution. However, this is not the case when we analyze datasets in the real world, and this brings up one of the main motivations of this thesis: imbalanced datasets.

Imbalanced datasets

Let $\mathcal{D} = \{x_i, y_i\}_{i=1}^{n}$ be a dataset where x_i is a real example and $y_i \in Y = \{1, 2, ..., c\}$ a class label associated to 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 datasets are composed of one or more classes (majority classes) that severely outrepresent other existing classes (minority classes) [2].

Given an imbalanced dataset $\mathcal{D}$, 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 dataset, its balanced counterpart $\mathcal{D}^u$.

2.2.2 Classification

Classification corresponds to one of the most popular tasks in machine learning. When doing classification, algorithms are asked to predict which of the n existing categories or classes a particular input belongs to. To perform this prediction, the model can output directly the expected class $1, ..., n$, or a bijective mapping of it, such as a one-hot encoded version of the class. Another possible form of the output is a probability distribution over the classes.

In classification problems, the performance of an algorithm can be measured with different evaluation metrics depending on the context. In this work, we employ one of the most popular metrics in multi-class classification: the accuracy classification score.

Classification accuracy

This metric calculates how often the model's predictions match the correct class or label. Given a set of n predicted integer classes $\hat{y}_1, ..., \hat{y}_n$ and their corresponding labels $y_1, ..., y_n$, the accuracy classification score for the set of samples is computed as:

$$acc = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{\{y_i\}}(\hat{y}_i) \tag{2.4}$$

where $\mathbf{1}_A(x)$ is the indicator function defined as:

$$\mathbf{1}_A(x) = \begin{cases} 1 & \text{if } \ x \in A \ , \\ 0 & \text{if } \ x \notin A \ . \end{cases} \tag{2.5}$$

2.2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are one of the critical components of the most significant advances in machine learning in the last years. They are particular types of neural networks that were designed inspired by the structure of the animal visual cortex, where each neuron in the cerebral cortex responds to stimuli in specific regions of the entire visual field, called receptive fields [27, 28]. CNN architectures are not new in the field of pattern recognition; they were already used in the nineties to solve pattern recognition tasks [29]. However, their use has become widely popular since they were utilized in deep architectures in the ImageNet image classification challenge in 2012 [30].

Due to the convolution operation's nature, which shares the weight of the kernels applied to the input features, the output of a convolution (often called *feature maps*) is considered to have translation-equivariant properties. Consequently, performing convolution operations along a particular dimension is intended to identify specific patterns that could be present in any location along this dimension. In the case of spatially correlated data such as images, these dimensions usually correspond to the image's height and width. In the case of temporally correlated data, these dimensions could correspond to the temporal dimension.

In the following section, and without intending to describe convolution operations completely, we provide a glimpse of how convolutions and deconvolutions work in a simplified two-dimensional case.

Convolutions

The convolution operation consists of sliding a filter or *kernel* w over an input x in order to produce an output y. An example of this is illustrated in Figure 2.4. At each position of the kernel, an individual output $y_{i,j}$ (green matrix) is computed by performing the sum of the products between the single components of the kernel w (shaded matrix) and the input x (blue matrix) according to:

$$y_{i,j} = \sum_{m,n} x_{i+m,j+n} \cdot w_{m,n} \tag{2.6}$$

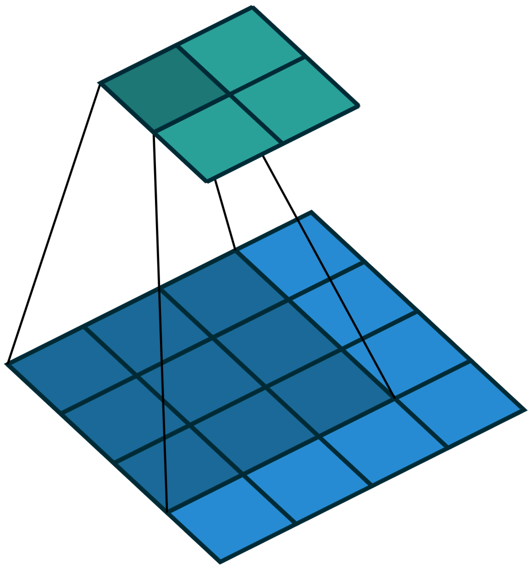


Figure 2.4: Diagram of a convolution operation. The 4x4 blue matrix represents the input. The 3x3 shaded matrix represents the kernel. The 2x2 green matrix represents the output. This figure was obtained from [31]

It is worth mentioning that Equation 2.6 does not precisely describe the convolution operation. Instead, it illustrates the cross-correlation, which is usually implemented as a substitute for the convolution operation in deep learning frameworks. These two terms are often used interchangeably since their difference is subtle, and it consists of whether the kernel is flipped to perform the multiplication or not.

To illustrate how a convolution operation works in this simplified case, a concrete example is analyzed. If the input and kernel in Figure 2.4 had values corresponding to:

$$x = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}; w = \begin{bmatrix} a & b & c \\ \mathrm{d} & \mathrm{e} & f \\ g & h & i \end{bmatrix}$$

The result of computing the convolution between them according to Equation 2.6 gives the output:

$$y = \begin{bmatrix} a+c+\mathrm{e}+g+i & b+\mathrm{d}+f+h \\ b+\mathrm{d}+f+h & a+c+\mathrm{e}+g+i \end{bmatrix}$$

.

Considering that convolutions are usually performed in modern hardware such as graphics processing units (GPUs) and that these devices can benefit considerably from performing matrix operations, it is helpful to count with an equivalent of the convolution in matrix form. To do so, it is necessary to transform w into its sparse version :

$$\mathbf{W} = \begin{bmatrix} a & b & c & 0 & \mathrm{d} & \mathrm{e} & f & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 \\ 0 & a & b & c & 0 & \mathrm{d} & \mathrm{e} & f & 0 & g & h & i & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a & b & c & 0 & \mathrm{d} & \mathrm{e} & f & 0 & g & h & i & 0 \\ 0 & 0 & 0 & 0 & 0 & a & b & c & 0 & \mathrm{d} & \mathrm{e} & f & 0 & g & h & i \end{bmatrix}$$

.

With the sparse matrix $\mathbf{W}$ and two additional reshaping operations, it is possible to replace the convolution operation of Equation 2.6 with a simple multiplication. Let the superscript ^{f} denote the flattened version of a given matrix. These two additional reshaping operations correspond to a flattening operation of the input x before computing the product and a resizing operation of the result y^f after calculating the product. In summary, the sparse kernel $\mathbf{W}$ can be multiplied with a flattened version of the input (x^f) to obtain a flattened version of the output y^f, from which we can recover the original output y.

An important observation of this product is that it performs a mapping $x^f \in \mathbb{R}^{16} \mapsto y^f \in \mathbb{R}^4$. Furthermore, if performing an inverse mapping was needed (from output space to input space), the same operation replacing $\mathbf{W}$ by its transpose would suffice, which motivates calling such operation as **transposed convolutions**.

While the regular product is used in the forward pass to perform the standard convolution, its transposed version is necessary to propagate the gradients in the backward pass during training.

Deconvolution

The use of transposed convolutions to perform the "inverse mapping" of a standard convolution has motivated the use of the term **deconvolution** when referring to these layers. Though this term is widely used and is also employed in this work to talk about transposed convolutions, its use should be careful since it can quickly lead to confusion. A transposed convolution **does not** correspond to a deconvolution's mathematical definition; i.e., it is not the inverse operation of convolution.

Figure 2.5 illustrates an example analogous to the direct case analyzed in Figure 2.4 to understand how deconvolution works. An effortless way to understand the dimensions of deconvolution operations is by remembering the idea of inverse mapping. The dimensions of the deconvolution's input (blue matrix in Figure 2.5) match the output of the corresponding direct convolution (green matrix in 2.4). In this sense, Figures 2.4 and 2.5 are almost upside-down versions of each other.

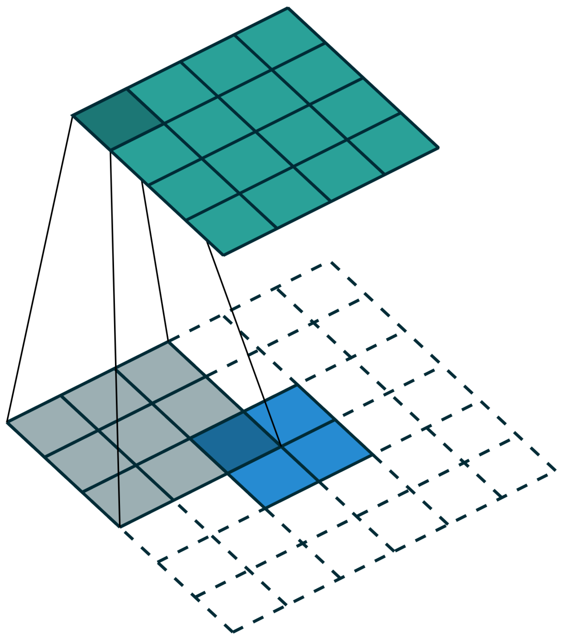


Figure 2.5: Diagram of a deconvolution operation. The 2x2 blue matrix represents the input. The 3x3 shaded matrix represents the kernel. The 4x4 green matrix represents the output. The dashed squares represent zero-padded portion of the input. This figure was obtained from [31]

Furthermore, going into the details of transposing the sparse matrix $\mathbf{W}$, it can be seen that multiplying the input of the deconvolution by $\mathbf{W^T}$ is equivalent to zero-padding the input followed by a direct convolution, as Figure 2.5 shows. This zero-padding step followed by a convolution used to compute the deconvolution also obeys the connectivity patterns of its corresponding direct convolution. In the direct convolution of Figure 2.4, any pixel in the corners of the input only contributes to the corresponding corner pixel of the output without contributing to any other pixel, similarly to when zero padding is applied to the deconvolution, as shown in Figure 2.5.

2.2.4 Generative Modeling

The field of generative modeling studies how to approximate data distributions. These approximations are usually parametric models that can summarize the information present in a given dataset. Given a parametric family of model distributions $\mathcal{P}$, the goal is to learn optimal parameters θ of the model that can minimize some notion of distance between the original data distribution P_r and the model distribution P_θ.

Types of Generative Models

Generative models can be classified into two large groups depending on how the model attempts to approximate the desired data distribution.

On the one hand, explicit density models aim to construct a closed-form density $p_\theta(x)$ that can be used directly to represent the likelihood and learn their parameters θ by maximizing the likelihood. These models can be further divided into models whose formulations

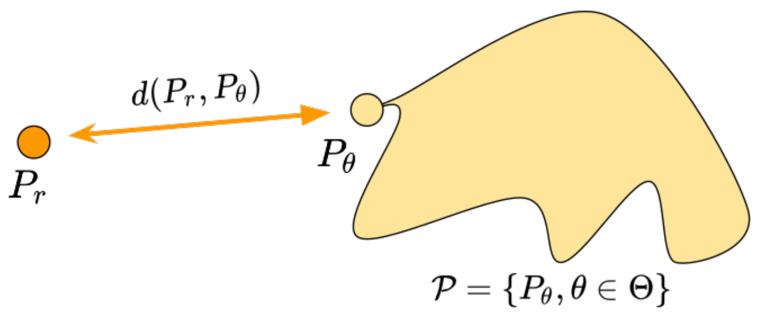


Figure 2.6: Illustration of the generative modeling scenario. The real and generated distributions are signaled with the points P_r and P_θ. The yellow shape on the right symbolizes the family of model distributions $\mathcal{P}$.

lead to tractable densities, such as Fully Visible Belief Networks (NADE [32], PixelRNN [33], PixelCNN [34]), or models that involve intractable densities and require approximation techniques. These approximations could be variational such as Variational Auto Encoders (VAE; [35]), or Markov Chain approximations, such as Deep Boltzmann Machines (DBM; [36]).

On the other hand, implicit density models do not intend to explicitly construct a closed-form density over the space where the data lies. These models, instead, interact with the probability distributions less directly by drawing samples from them. This sampling process can consist of multiple steps in the case of Markov Chain-based models such as Deep Generative Stochastic Networks (GSN; [37]), or it can consist of a single step as in the algorithm of our interest: GANs.

It is important to note that the provided classification of generative models is valid to describe the essence of the original algorithms presented in it. However, some interesting mixes between implicit and explicit algorithms could happen. Implicit models such as GANs could be used to define explicit densities, which does not contradict the fact that the training algorithm of the GAN itself uniquely uses the model's ability to draw samples from a distribution.

Likelihood-free Learning

As the name suggests, the term likelihood-free learning refers to models that do not attempt to maximize a likelihood function, being able to learn without directly depending on such a procedure. In generative modeling, multiple reasons could support the choice of a likelihood-free model. Probably the most substantial one is the evidence of the likelihood of being generally uninformative about the quality of generated samples and vice versa [38]. Moreover, using such an approach inevitably leads to the density estimation problem, which assumes that the original data distribution admits a density function in the first place.

In addition, tractable techniques for this estimation could become extremely complex when the data lies in highly-dimensional spaces. Finally, in the presence of intractable density estimation, variational approximations attempt to maximize a lower bound of the model's likelihood without necessarily maximizing the original likelihood, and the intractability of the densities does not permit providing a confident measure of how close the provided solution is to the optimal one. All these reasons motivate using an implicit model that is not directly dependent on a likelihood function.

Manifold Learning

In machine learning, it is common to find ideas based on the concept of manifolds. Without intending to provide a formal mathematical definition of the concept, we will understand the term manifold as a connected set of points that can be approximated well by considering only a small number of dimensions embedded in a higher-dimensional space [26]. For example, a sphere (the surface of a three-dimensional ball) is a two-dimensional manifold embedded in a three-dimensional space.

Manifolds often appear in machine learning algorithms that rely on the **manifold assumption**. This assumption is based on the idea that data probability distributions in $\mathbb{R}^n$ are highly concentrated in some regions. At the same time, most of the remaining areas correspond to areas with invalid inputs. Consequently, a machine learning algorithm only needs to look for variations across a collection of manifolds containing small subsets of points instead of considering the entire $\mathbb{R}^n$ space. The manifold assumption will be particularly relevant for understanding one of this work's core concepts: Generative Adversarial Networks.

2.2.5 Generative Adversarial Networks

As a side note, in the previous sections the model distribution has correctly been denoted by P_θ because it is a model distribution parameterized by θ. For clarifying purposes, we will assume that model distributions are parameterized by some parameters θ and simply refer to the model distribution as P_g, where the sub-index $_g$ stands for **generated** distribution, in contrast with the **real** distribution P_r.

A Two-Sample Test Perspective

With the concepts introduced in Section 2.2.4, it is possible to formulate the Generative Adversarial Networks framework following a classical two-sample test formulation. Because of the undesirable properties of explicit density estimation mentioned in Section 2.2.4, we narrow the scope to implicitly modeling the original data distribution P_r.

Assuming that we have access to the samples from the real distribution P_r and a generative model that defines a generated distribution P_g from which we can sample efficiently, we now have two different sets of samples: $S_r = \{x_r \sim P_r\}$ and $S_g = \{x_g \sim P_g\}$. To assess the quality of the generated samples, it is necessary to decide whether these two sets of samples could come from the same distribution or not, considering the null and alternative hypotheses $H_0 : P_r = P_g$ and $H_1 : P_r \neq P_g$ respectively.

To complete the two-sample test, we need a test statistic T that can compute some sort of difference between the set of samples S_r and S_g. From the generative model's perspective, we would like the samples in S_g to be similar to those in S_r. Thus the goal of the generative model is to minimize the two-sample test statistic T so H_0 cannot be rejected.

In a low-dimensional case where our distributions are real-valued Gaussians, T could simply be the difference between the sample means or sample variances between S_r and S_g. However, finding a suitable test statistic for high-dimensional data is challenging. Therefore, we could benefit significantly from learning it instead of exhaustively looking for appropriate candidates. To do this, we will use a model called discriminator.

It is worth mentioning that performing this two-sample test is inherently likelihood-free since it only computes T based on the samples x_r and x_g and not their corresponding densities. With the notion of a two-sample test in a generative modeling environment, it is now possible to provide a clear explanation of the GAN framework.

The Discriminator

The discriminator is any model with learnable parameters that provides a notion of distance between the samples from S_r and S_g. Let V denote this distance.

The distance V is analogous to the test statistic T. It is computed by considering the output of D (denoted as $\hat{y}$ in Figure 2.7) when evaluating the samples x_r and x_g individually. The goal of D is to maximize V in support of the alternative hypothesis $P_r \neq P_g$, which justifies using the letter V to denote this notion of distance: it is the *value function* that D will maximize to push the distributions apart as far as possible.

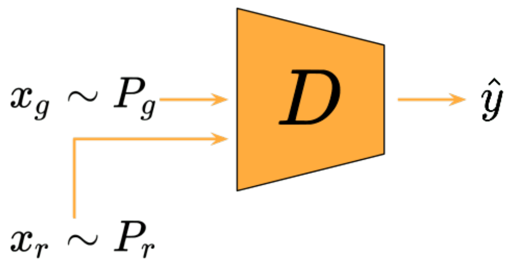


Figure 2.7: Diagram of the input and output of the discriminator. x_g and x_r correspond to real and generated samples, respectively, and $\hat{y}$ is the output of the discriminator computed both inputs independently.

So far, we have mentioned the existence of a generative model that permits efficient sampling, but no additional details have been provided. The responsible for allowing the sampling from our generated data distribution will be the generator.

The Generator

The generator is a model that performs a mapping between a latent space (denoted by z in Figure 2.8) and the real sample space, intending to generate realistic samples x_g that resemble those in the distribution P_r. The goal of G, in opposition to D, is to minimize the value function V in support of the null hypothesis $P_r = P_g$, bringing P_g as close to P_r as possible.

The generated samples x_g correspond to the samples drawn from our generated distribution P_g. In that sense, G itself performs the action of sampling from the distribution. For G to mimic the action of sampling from a probability distribution, some source of stochasticity is needed. This stochasticity required for the sampling process is obtained from the latent variable z, sampled from a multivariate Gaussian distribution.

The manifold assumption previously described in Section 2.2.4 is an unstated assumption under generating samples from this random latent space. We assume that G makes it possible to recover the high dimensional distribution in which the data lies, requiring only a low dimensional Gaussian space as a starting point.

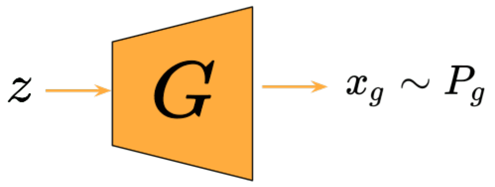


Figure 2.8: Diagram of the input and output of the generator. The input latent variable is denoted by z, and x_g corresponds to the generated sample.

The Two Player Game

The GAN framework consists of a game between two networks. Given an input dataset of real samples $S_r = \{x_r \sim P_r\}$, the generator network (G) aims to generate fake samples $S_g = \{x_g \sim P_g\}$. In contrast, the discriminator network (D) tries to distinguish between real 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 realistic samples, while on the other hand D is trained to predict whether a given sample comes from the input dataset or was generated by G. At the end of the training, G will generate samples similar to those in the input dataset, and the D will be unable to tell apart generated from real samples.

The objective of the two players can be written together in the form of a min-max game, a well-known problem in the field of game theory, according to:

$$\min_{\theta_G} \max_{\theta_D} V(\theta_G, \theta_D) \tag{2.7}$$

The formulation presented in Equation 2.7 is also known as a zero-sum game, which comes from both players attempting to maximize objective functions (V for D, and $-V$ for G) that sum up to zero.

To migrate from the game theory formulation presented in 2.7 to a machine learning-like formulation, our models must have objective functions that they aim to minimize. These functions, typically called loss or cost functions, correspond to:

$$L_D = -V(\theta_G, \theta_G) \tag{2.8}$$
$$L_G = V(\theta_G, \theta_G) \tag{2.9}$$

In the original GAN formulation [6], these losses were defined by assigning the real/fake classification task to D, which naturally converts L_D into a binary cross-entropy loss function given by:

$$L_D = -\mathop{\mathbb{E}}_{x_r \sim P_r}[\log D(x_r)] - \mathop{\mathbb{E}}_{x_g \sim P_g}[\log(1 - D(x_g))] \tag{2.10}$$

$$L_G = \mathop{\mathbb{E}}_{x_r \sim P_r}[\log D(x_r)] + \mathop{\mathbb{E}}_{x_g \sim P_g}[\log(1 - D(x_g))] \tag{2.11}$$

The single training steps for each network are explained in detail in the following sections.

Training the Discriminator

Figure 2.9 details the entire process for updating D's parameters. After evaluating individually real and fake samples generated by G, it is possible to compute D's loss function L_D. The computation of this loss involves comparing D's output $\hat{y}$ and a target y which indicates the actual distribution each sample came from, either real or generated. Once the loss has been computed, the gradient descent step can be performed to modify uniquely D's parameters θ_D.

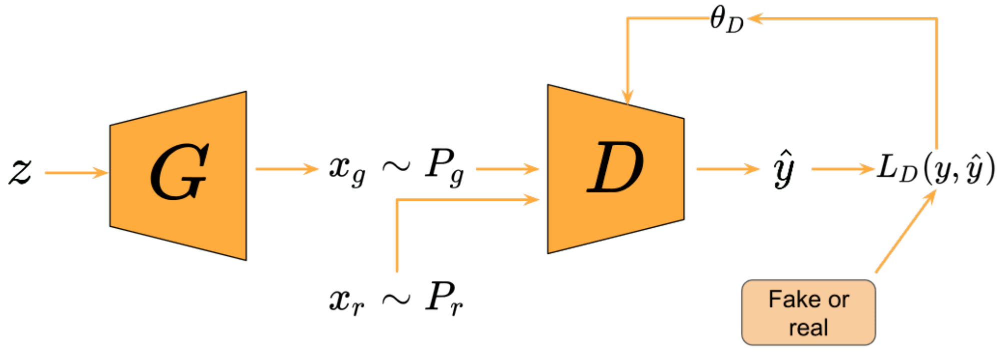


Figure 2.9: Details of updating D's parameters θ_D. x_g and x_r correspond to real and generated samples, respectively, and $\hat{y}$ is the output of the discriminator computed on both inputs independently. L_D is the loss described in Equation 2.8

Training the Generator

Figure 2.10 provides details for updating G's parameters. Once G produces the fake samples, they are evaluated by D. The comparison between D's output and the previously mentioned target y permits the computation of the loss function L_G, which is necessary to perform the gradient descent step and update G's parameters θ_G. It is worth noting that even though the loss in Equation 2.11 includes the term $\mathbb{E}_{x_r \sim P_r}[\log D(x_r)]$ according to the min-max formulation, this term vanishes when computing the gradients with respect to θ_G. Therefore, the only feedback that G needs from D comes from the evaluation of the fake samples.

2.2.6 Conditional Generation

Following the formulation described in Section 2.2.5, a generative model that successfully approximates the real data distribution would be able to arbitrarily generate samples x_g that look indistinguishable from the real samples x_r. However, if we were interested in a specific attribute of the generated samples, it would be necessary to generate a large number

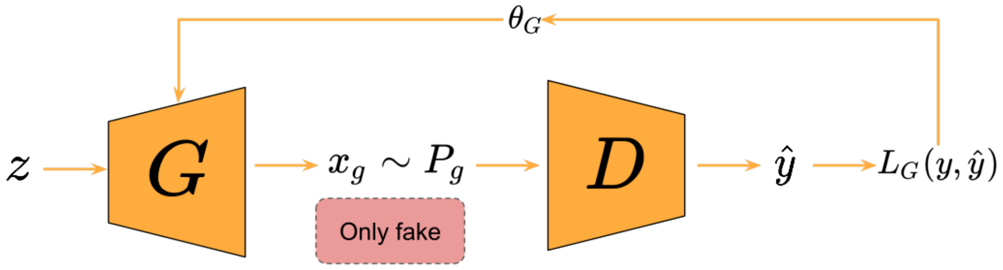


Figure 2.10: Details of updating G's parameters θ_G. x_g correspond to the generated samples and $\hat{y}$ is the output of the discriminator computed on the generated samples. L_G is the loss described in Equation 2.9

of samples to apply then some filtering step to gather the samples that capture the desired attribute.

Though possible, this filtering step can be quite challenging if the desired attribute cannot be obtained straightforwardly from the data, requiring a lot of visual inspection for large datasets or additional hours of computation if the filtering requires additional models. The conditional generation formulation comes to solve this problem, allowing for the generation of samples that belong to desired conditional distribution

Conditional GANs

In the context of GANs, the first conditional model corresponded to the Conditional Generative Adversarial Nets (C-GANs; [39]), which enabled the conditional generation of samples by introducing a subtle modification to the original GAN framework. To condition on the desired attribute, it is enough to add it to the inputs of G and D, forming a new joint representation.

Let $\bar{z}$ denote the desired attributes of interest. Figure 2.11 shows how $\bar{z}$ is combined with the original latent space z to produce z', the new input of G. Analogously, Figure 2.12 shows how $\bar{z}$ is combined with real and generated samples to produce the inputs of D. We intentionally denote D's conditional attributes by the same letter $\bar{z}$ to emphasize that its representations should contain the same attributes used to generate x_g. These attributes are shared between real and generated samples, suggesting that we should first pick an attribute $\bar{z}$ from the real data set during training, and then generate samples conditioned on this attribute. Finally, real and generated samples can be processed by D.

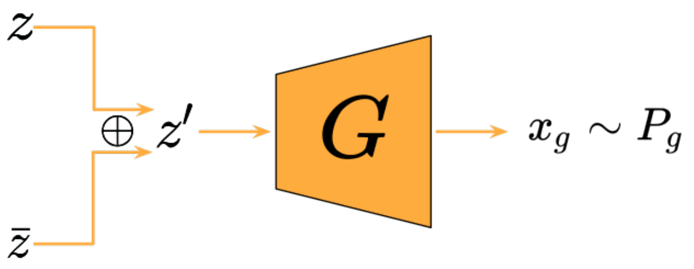


Figure 2.11: Diagram of the modified input for the C-GAN's generator. The input latent variable is denoted by z, the conditional attributes by $\bar{z}$, and the generated samples by x_g.

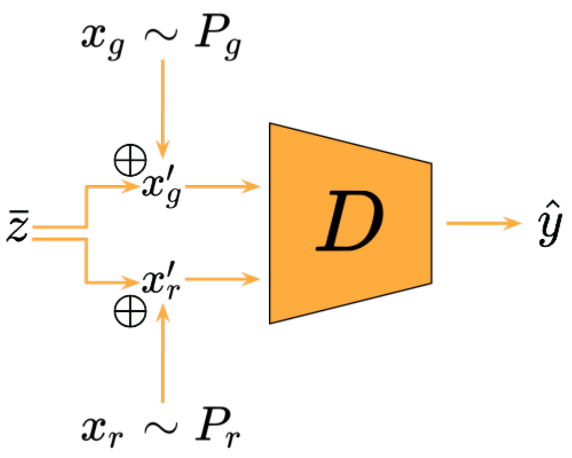


Figure 2.12: Diagram of the modified input for the C-GAN's discriminator. The conditional attributes are denoted by $\bar{z}$, x_g and x_r correspond to real and generated samples respectively, and $\hat{y}$ is the output of the discriminator computed on both inputs independently.

We show a concatenation operation denoted by $\oplus$ to create the joint representation in both cases, but this is just an option among different approaches to combine these representations. Regarding $\bar{z}$ and what it includes, a widely used conditional attribute is the class or category of the samples, which allows the model to perform class-conditional generation. However, this framework is quite flexible in the sense that the attribute $\bar{z}$ of interest does not need to be the class, and it could include one or many other features of interest.

Auxiliary Classifier GANs

An alternative to C-GANs specifically proposed for class-conditional generation is the Auxiliary Classifier GANs (AC-GANs; [40]). This approach tackles the conditional generation problem with a classification task. Instead of providing the conditional attribute to both networks, it modifies D to contain an auxiliary decoder network that outputs the class label of its input samples. This situation is described in Figure 2.13, where D now outputs two different values simultaneously. The original output, previously denoted as $\hat{y}$ in Figures 2.7, 2.9, and 2.12, is now denoted by D_{rg} for clarity purposes, indicating that it attempts to distinguish real from generated samples. The additional output that attempts to predict the different classes of both generated and real samples is denoted by D_y.

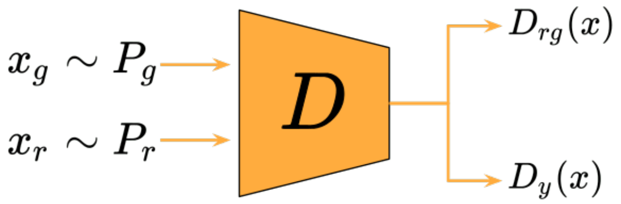


Figure 2.13: Diagram of the AC-GAN's discriminator, where x_g and x_r correspond to real and generated samples, respectively, D_{rg} corresponds to the original output, and D_y corresponds to the class prediction. Both of the outputs are computed simultaneously for each input.

These two outputs can be used in two different cross-entropy losses. First, D_{rg} is used in the original binary cross-entropy to separate real from generated (L_D in Equation 2.10), and a multi-class cross-entropy (denoted by H_r and H_g for real and generated samples respectively) to separate the different classes presented in the data set. These two losses can be linearly combined to provide the new loss functions $\widetilde{L_D}$ and $\widetilde{L_G}$ according to:

$$\begin{aligned}
\widetilde{L_D} &= -\mathop{\mathbb{E}}_{x_r \sim P_r}[\log D_{rg}(x_r)] - \mathop{\mathbb{E}}_{x_g \sim P_g}[\log(1 - D_{rg}(x_g))] \\
&\quad + \xi(\mathop{\mathbb{E}}_{x_r \sim P_r}[H_r] + \mathop{\mathbb{E}}_{x_g \sim P_g}[H_g]) \qquad (2.12) \\
\widetilde{L_G} &= \mathop{\mathbb{E}}_{x_g \sim P_g}[\log(1 - D_{rg}(x_g))] + \xi \mathop{\mathbb{E}}_{x_g \sim P_g}[H_g] \qquad (2.13)
\end{aligned}$$

An interesting fact about the additional terms added in Equations 2.12 and 2.13 is that their presence inevitably breaks the zero-sum formulation since both networks do not play adversarial roles anymore; both of them attempt to help solve the multi-class classification problem. Specifically, these additional terms guide D to perform the multi-class classification itself, and G to generate samples that can be correctly classified by its corresponding D at any given moment of the training. It should also be noted that the first term of Equation 2.11 was omitted in Equation 2.13 for its previously mentioned irrelevance.

2.2.7 State of the art in GANs

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 conditional-generation models ([39]; [40]; [41]), models that stabilize the erratic behavior of the original GANs ([42]; [43]; [44]), and models that generate samples with an impressively high quality and resolution ([45]; [46]; [47]; [48]) among many other models and applications. An extensive description of GAN models in computer vision is provided in [49].

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 the C-RNN-GAN ([50]), which proposed adding recurrent neural networks to the GAN's generator and discriminator to handle the temporal evolution of the time-series. This work was followed by the RC-GAN ([51]) which added label-conditional generation and a focus the downstream medical task.

More recently, [52] introduced a jointly trained embedding network that combines the unsupervised GAN framework with a supervised auto-regressive model to capture the conditional temporal dynamics of the time-series.

Lately, [53] 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.

Despite new complex models with elaborated architectures or more extravagant formulations of the losses that we may consider using, they all miss one key aspect: dealing with irregularly-spaced data. This missing key component makes them unsuitable for our problem.

We intend to focus on this work in going further than just generation, planning to provide helpful metrics to leverage the use of generative models for the classification of astronomical light curves. Hence, while we could continually update the generative model up to the last model claiming to be the state-of-the-art in generation, our interest is also to provide orthogonal techniques to the incremental improvements that newer models could add.

Considering this, we build our generative model based on the only existing GAN model - to our knowledge- designed to deal with astronomical time-series: the Time-Conditional Generative Adversarial Network.

Time-Conditional Generative Adversarial Network

This work, also known as T-CGAN ([11]), proposes a method to generate irregularly spaced time series. It extends the C-GAN framework explained in Section 2.2.6 by using the time-series timestamps as an attribute to perform conditional generation. This attempts to understand the time-series temporal behavior and generate time-series values that are consequent with the provided timestamps.

The architectures used for D and G are convolutional and deconvolutional neural networks, respectively. These architectures are adequate because all the datasets used were subsampled to a fixed length. In addition, the networks are trained based on the original GAN framework described in Section 2.2.5, and no stopping criterion is defined for the training of GAN.

Even though we preserve the use of convolutional networks and fixed-length time series, there are many aspects that we intend to improve. Firstly, this work does not use real astronomical data and only considers binary toy datasets. Secondly, it does not include conditional generation with physical parameters of interest, nor does it performs conditional multi-class generation. Finally, and similarly to the previously described models, it does not tackle the problem of model selection for a downstream task.

The following section describes one of the essential modifications our methodology includes to the TC-GAN: The Wasserstein GAN.

Wasserstein GAN

Although the GAN framework could be cataloged as intuitive and straightforward, training a GAN in practice can be difficult. Some of these difficulties are described in Section 2.2.8. The Wasserstein-GAN (WGAN, [42]) can overcome some of these training difficulties, being a widely used GAN model known for its training stability. This GAN proposes replacing the notion of distance from the original framework with the Wasserstein-1 distance $W(P_r, P_g)$, also known as the Earth Mover distance. Informally, this distance measures the minimum cost of transporting all the mass in the distribution A to the distribution B; which can be formally stated as :

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x_r, x_g) \sim \gamma}[||x_r - x_g||] \tag{2.14}$$

where $\gamma(x_r, x_g)$ denotes any joint distribution whose marginals are P_r and P_g, respectively, and $\Pi(P_r, P_g)$ represents the set of these joint distributions.

Since the mathematical formulation in Equation 2.14 can be a little rough at first sight, we attempt to provide a friendly interpretation. Given any transport plan or mapping that one can use to transform P_r and P_g, the expectation computes the total cost of performing such action. The W distance corresponds then to infimum, i.e., the greatest lower bound among all these transport plans.

The W distance has desirable properties that make it suitable to be a loss function. This distance is continuous everywhere and differentiable almost everywhere, a property that the original losses are said to be lacking [42]. However, it also has some drawbacks; it is highly intractable due to the infimum across all the possible transport plans. For this reason, the W distance is approximated using the Kantorovich-Rubinstein duality ([54]) according to:

$$W(P_r, P_g) = \sup_{\|f\| \leq 1} \mathbb{E}_{x_r \sim P_r}[f(x_r)] - \mathbb{E}_{x_g \sim P_g}[f(x_g)] \tag{2.15}$$

where the supremum is searched over the space of all 1-Lipschitz functions. Now, if we have a parameterized family of functions $\{D_{\theta_D}\}$ that satisfy the Lipschitz condition, we could consider solving the alternative problem:

$$\max_{\theta_D} \mathbb{E}_{x_r \sim P_r}[D(x_r)] - \mathbb{E}_{x_g \sim P_g}[D(x_g)] \tag{2.16}$$

Computing the value in Equation 2.16 would be equivalent to computing $W(P_r, P_g)$ up to a multiplicative constant under the assumption that the supremum of Equation 2.15 can be reached for some value θ_D. This expression can then be used to measure the dissimilarity or distance between the two distributions. The Lipschitz condition on D is enforced rustically by performing a weight clipping step on θ_D after each gradient update. Finally, the losses can be stated as follows:

$$\begin{aligned} L_D &= \mathop{\mathbb{E}}_{x_g \sim P_g}[D(x_g)]) - \mathop{\mathbb{E}}_{x_r \sim P_r}[D(x_r)] & (2.17) \\ L_G &= -\mathop{\mathbb{E}}_{x_g \sim P_g}[D(x_g)] & (2.18) \end{aligned}$$

Wasserstein GAN with Gradient Penalty

An upgraded version of the WGAN is the WGAN with Gradient Penalty (WGAN-GP, [43]). This model adds a regularization term to the original WGAN loss to satisfy the Lipschitz condition on D instead of weight-clipping. The WGAN-GP objective that is minimized by D during the training process is:

$$L_D = \mathop{\mathbb{E}}_{x_g \sim P_g}[D(x_g)]) - \mathop{\mathbb{E}}_{x_r \sim P_r}[D(x_r)] + \lambda \mathop{\mathbb{E}}_{\hat{x} \sim P_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2] \tag{2.19}$$

where L_G is not specified since it remains the same as Equation 2.18, $P_{\hat{x}}$ 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.

2.2.8 Training difficulties: A not so fun game

Unfortunately, GAN training can be a challenging task, and it is known to be highly unstable and many times does not reach convergence [55]. These difficulties arise because the adversarial game between the two networks results in a non-convex optimization problem. To be more specific, each player tries to reduce their cost, but none of them has total control since it can still be affected by the other player. In this way, each player can indefinitely make infinitesimal small gradient steps to reduce their own cost at the expense of the other player, going into circular orbits rather than arriving at a common equilibrium point.

Mode Collapse

One of the most famous GAN problems is when G learns to produce extremely good samples with minimal diversity. These samples, repeatedly produced by G, can eventually trick D and trap into a local minimum, inducing the failure of the training. This phenomenon is known as mode collapse since G learns to collapse the entire probability distribution into minimal portions.

Overfitting in GANs

As described in [56], overfitting in GANs occurs when training on small datasets. 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 [56] 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.

2.2.9 Evaluation of GANs

The losses described in Equations 2.18, 2.17 successfully describe the adversarial problem and quantify the distance between P_r and P_g. However, 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 dataset. 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 [57].

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.

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 [58] and later used in [51], [59], [60] and [61]. For clarity, we choose to preserve the names in [51]: 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.

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 pre-trained classifier ([62]; [63]; [64]; [65]; [57]). The classifier is generally a convolutional neural network (CNN) such as the Inception-v3 [66], 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*. As mentioned in Section 2.2.4, 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.

The Fréchet Inception Distance (FID) This metric proposed by [63] 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:

$$FID = \underbrace{||\mu_r - \mu_g||}_{\text{(a)}}^2 + \mathrm{Tr}(\underbrace{\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}}_{\text{(b)}}) \tag{2.20}$$

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 [67].

Precision and Recall This work proposed separating fidelity and diversity into two relative-density-based metrics: **precision** and **recall** [64]. 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 2.20.

Improved Precision and Recall Motivated by the failure at identifying models with poor variability, this work proposed improved precision and recall metrics (**P&R**; [65]). These metrics are computed by estimating the manifolds $\Phi \in \{\Phi_r, \Phi_g\}$ according to:

$$\hat{\Phi} = \bigcup_{\phi \in \mathbf{\Phi}} B(\phi, NND_k(\phi)) \tag{2.21}$$

where $\mathbf{\Phi} \in \{\mathbf{\Phi}_r, \mathbf{\Phi}_g\}$ 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 k-th nearest neighbor within the corresponding manifold. In the presence of outliers, the KNN approach results in an over-estimation of the manifolds due to the large distances between samples.

Let $\hat{\Phi}_r$ and $\hat{\Phi}_g$ be the approximations of the real and generated manifolds described in Equation 2.21, the **P&R** metrics can be computed according to:

$$\begin{aligned} \mathbf{P}_{(\mathbf{\Phi}_r, \mathbf{\Phi}_g)} &= \frac{1}{|\mathbf{\Phi}_g|} \sum_{\phi_g \in \mathbf{\Phi}_g} \mathbf{1}_{\hat{\Phi}_r}(\phi_g) && (2.22) \\ \mathbf{R}_{(\mathbf{\Phi}_r, \mathbf{\Phi}_g)} &= \frac{1}{|\mathbf{\Phi}_r|} \sum_{\phi_r \in \mathbf{\Phi}_r} \mathbf{1}_{\hat{\Phi}_g}(\phi_r) && (2.23) \end{aligned}$$

To completely understand how to compute these metrics, Figure 2.15 shows a two-dimensional example with the estimated manifolds for $K = 2$. Due to the symmetry of Equations 2.22 and 2.23, we will only show how to compute **P**. For every generated feature sample ϕ_g, we need to check if it belongs to at least one of the red balls $B(\phi_r, NND_2(\phi_r))$. Since all ϕ_g satisfy this condition, we obtain $\mathbf{P} = \frac{1}{4}(1 + 1 + 1 + 1) = 1$.

Density and Coverage This work proposed **density** and **coverage** (**D&C**; [57]) motivated by the vulnerability of **P&R** to outliers. While **P** measures fidelity depending on the

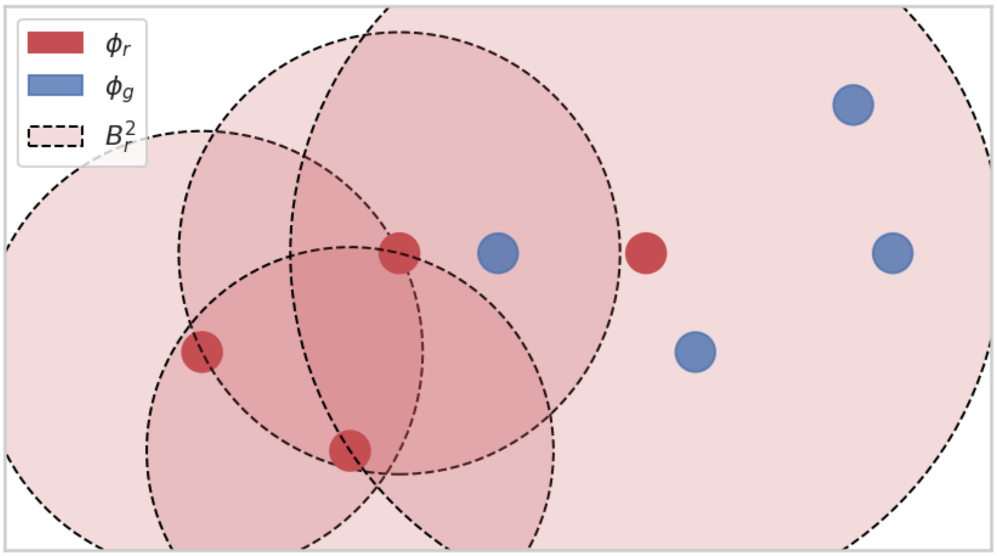


Figure 2.14

Figure 2.15: Two-dimensional scenario of the real and generated manifold approximates. 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 generated feature samples ϕ_g are shown in blue.

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**.

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 2.21, we compute the **D** metric according to:

$$\mathbf{D}_{(\mathbf{\Phi}_r,\mathbf{\Phi}_g)} = \frac{1}{k|\mathbf{\Phi}_g|} \sum_{\phi_g \in \mathbf{\Phi}_g} \sum_{\phi_r \in \mathbf{\Phi}_r} \mathbf{1}_{B_r^k}(\phi_g) \tag{2.24}$$

where $\mathbf{1}_A(x)$ is the same indicator function defined in Equation 2.5. Returning to the example of Figure 2.15, we compute **D** by counting the number of red balls $B(\phi_r, NND_k(\phi_r))$ that enclose each ϕ_g, resulting in $\mathbf{D} = \frac{1}{2 \cdot 4}(2+1+1+1) = \frac{5}{8} = 0.625$. This represents clearly the improvements of **D** upon **P**, assigning a lower value to a region with a low concentration of real samples, which in the case of **P** resulted in an overestimated value due to the presence of one outlier sample.

We found that **P** and **C** saturate quickly in our practical case, 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 over-estimation issue as outliers. Consequently, we decide to use **D** and **R** as our fidelity and diversity metrics (and disregard showing how to compute **C**).

2.2.10 Additional GAN techniques

Exponential Moving Averages

Among the reasons for non-convergence in adversarial algorithms, we find cycling behaviors around optimal solutions ([68]; [69]) and slow outward spiraling ([70]). To tackle this cycling behavior without intervening in the adversarial game between G and D, performing exponential moving averages over G's parameters during training is proposed [55]. This technique helps alleviate the cyclic behavior of the algorithm by shrinking the amplitude of the oscillations around optimal solutions.

The averaging operation is performed as:

$$\bar{\theta}_G^{(t)} = \bar{\theta}_G^{(t-1)} + (1-\delta)\theta_G^{(t)} \tag{2.25}$$

where $\bar{\theta}_G^t$ represents the exponentially averaged version of the parameter θ_G at iteration t, and δ modulates the effective time windows considered into the average. As δ gets closer to 1, the effective time window considered for computing the averages is larger.

It should be clarified that the averaging operation is only computed over G, and it is not used during training to compute any gradient. Instead, it is only used for inference.

2.2.11 Data Augmentation

Data augmentation refers to a set of techniques applied to a dataset used to create new samples that are slightly different from the existing ones to increase the number of samples in the dataset. It is frequently used to prevent overfitting, and it helps improve the performance of machine learning models for various applications [3]. Classic examples of this in the field of images are rotations, translations, crops, and flips, among others.

Time-series methods

In the time domain, data augmentation techniques are less standardized. Traditional techniques in time-series correspond to non-parametric transformations such as jittering, scaling, window-slicing, and window-warping [71]. Parametric techniques can also be applied in data augmentation, such as the parametric model-based augmentation for transient phenomena proposed in [72].

Between these time-series methods, we employ the window-warping transformation for our baseline experiments. The reason for this choice is explained in Section 3.3.8.

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 $t_1 \leq t_w \leq t_2$. 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 length of the time-series.

Chapter 3

Methodology

This chapter provides a detailed description of the proposed methodology. First, an overview of the datasets and their pre-processing steps is provided. Then a complete description of the framework is given, including the details of the classifier and the generative models, a preliminary experiment that motivates the two novel contributions of the work (resampling block and $\mathcal{G}$-score), and the baselines utilized for comparison.

3.1 Datasets

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.

The Catalina Surveys Data Release-1 This catalog described in [15], captured with the 8.2 deg^2 field-of-view camera mounted on the CSS 27-inch Schmidt telescope, provides $\sim$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 3.1.

The Zwicky Transient Facility This survey, known by its acronym ZTF [19] provides a public multi-band 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 [21], 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 ([25], [73]) have already used ZTF data processed by the ALeRCE broker

Table 3.1: Adopted classes distribution for the Catalina Surveys Data Release-1. The original class acronyms as described in [15] are shown in (·).

New class	Original class
EBSD/D	Contact eclipsing binary (EW)
	Semi-detached eclipsing binary (β Lyrae)
RRL	Fundamental mode RR Lyrae (RRab)
	First over-tone mode RR Lyrae (RRc)
	Multi-mode 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)

to train different machine learning algorithms. More details about the data processing can be found in [21].

After pre-processing both datasets following the steps detailed in Section 3.2, we obtain the definitive versions of the datasets that will be used in our experiments, from now on referred to as the "Catalina" and the "ZTF" datasets. The class distributions of the pre-processed datasets are shown in Table 3.2.

Table 3.2: Class distributions of the pre-processed datasets.

Catalina		**ZTF**	
Class	Nº samples	Class	Nº samples
EBSD/D	28980	EB	31477
RRL	7533	RRL	18729
EBC	4500	LPV	5245
LPV	483	DSCT	507
DSCT	241	CEP	471
CEP	182		

3.2 Data pre-processing

To use the data described in Section 3.1, some pre-processing steps need to be applied. The pre-processing consists of four main steps: period folding, outlier filtering, time sampling, and median centering.

Period folding

The first step of the pre-processing consists of mapping the observation times to phase space, operation that was described in Section 2.1.3.

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.

Outlier filtering

Considering that some of the light curves in the datasets can include a significant amount of noise, we filter out anomalous observations within each curve of both datasets. 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 dataset, 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 surrounding observations. This comparison is performed using the *z-score*[1] 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 $z_{score} > 3$, and repeating two 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 3.1b. 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 [25], we use $mag_{thr} = 30$.

Time sub-sampling

To bring the problem to a more straightforward domain, we set the length of the light curves to a predefined value for each dataset. 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 datasets contain irregularly sampled light curves, and we perform the sub-sampling step after the period folding step, choosing an observation implies selecting a magnitude with its corresponding observation phase. Both magnitudes and phases are part

[1]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 $z = \frac{x-\mu}{\sigma}$

of the input of our models, as will be detailed in Section 3.3. Figure 3.1b shows an example of the time sub-sampling step.

The light curve length is set to 100 observations for the Catalina dataset, whereas that of the ZTF dataset 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 $41k$ and $56k$ samples in the Catalina and ZTF datasets, respectively, whose class distribution is shown in Table 3.2.

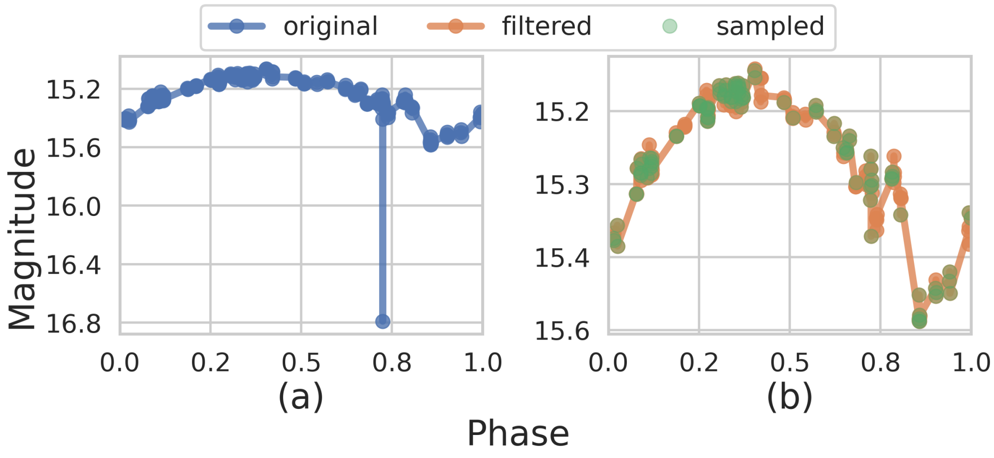


Figure 3.1: (a) Original cepheid from the Catalina dataset. (b) Filtered and sub-sampled versions of the original cepheid.

Median centering

The last step to get the data ready for data generation is centering it around 0 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.

3.3 The Proposed Framework

3.3.1 General Description

We propose a conditional generation approach that extends the T-CGAN [11], adding the class and 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 3.3.2.

A summary of the proposed methodology, that details the partitions of datasets for the models and metrics is provided in Figure 3.2.

We start by partitioning the pre-processed dataset $\mathcal{D}$ into $\mathcal{D}_{train}$, $\mathcal{D}_{val}$ and $\mathcal{D}_{test}$, the *train*, *validation*, and *test* sets. Each class in $\mathcal{D}_{val}$ and $\mathcal{D}_{test}$ contains 20% of the total number of samples of the smallest class in $\mathcal{D}$. To train the GAN and the classifier we use and D^u_{train}, a uniformly balanced version of the original $\mathcal{D}_{train}$ obtained through the resampling block that will be explained in Section 3.3.6.

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

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

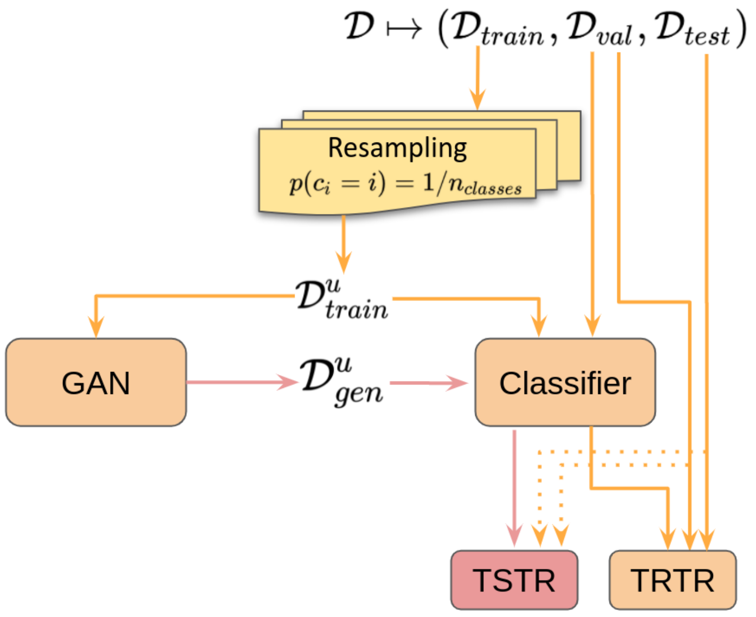

$\mathcal{D} \mapsto (\mathcal{D}_{train}, \mathcal{D}_{val}, \mathcal{D}_{test})$
Resampling
$p(c_i = i) = 1/n_{classes}$
$\mathcal{D}^u_{train}$
GAN
$\mathcal{D}^u_{gen}$
Classifier
TSTR
TRTR

Figure 3.2: Diagram of the methodology

3.3.2 Data structure details

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 dataset $\mathcal{D}_{train}$ 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 datasets, respectively, obeying roughly the proportion between the light curve lengths of the datasets. Following a C-GAN-like approach [39], 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 $\mathbf{a}$ and $\mathbf{c}$ be tensor versions of a and c, and L and N denote the light curve length and number of classes of a dataset; we define $\mathbf{a} \in \mathbb{R}^L$ as a vector with value a in all its components, and $\mathbf{c} \in \mathbb{R}^{L \times N}$ as a one-hot encoding of c, composed by $\mathbf{0}$'s and $\mathbf{1}$'s vectors, where $\{\mathbf{0}, \mathbf{1}\} \in \mathbb{R}^L$. The tensor version of $\bar{z}$ is $\bar{\mathbf{z}} = [\phi_t, \mathbf{a}, \mathbf{c}] \in \mathbb{R}^{L \times 2+N}$. The concatenation of $\bar{\mathbf{z}}$ and real or generated magnitudes will be the input of D, with dimensions $L \times 3 + N$.

3.3.3 Classifier details

To reduce the variance of the experiments, the classifier consists of an ensemble of 5 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 [11]. The input is forwarded through a set of convolution blocks that halve the temporal dimension, followed by dense layers. The network is trained with a batch size of 256 and using Adam optimizer [74] with $\alpha = 0.0001, \beta_1 = 0.9, \beta_2 = 0.999$. Table 3.3 shows the detailed architecture of the base classifier. To compute all the feature-based metrics explained in Section 2.2.9, we use the output of the last convolution block of this base classifier, trained on each of the datasets separately.

To save computation time and avoid overfitting, we employ an early stopping criterion that stops training after seeing no improvements in the validation losses 5 times in a row. Validation is done every 100 iterations. In addition, after seeing no improvements in the validation losses 3 times in a row, it restores the previous best model so far and halves the learning rate.

3.3.4 GAN details

In addition to the original WGAN-GP formulation explained in Section 2.2.7, we include additional regularization terms to Equation 2.18 and 2.19. Following the AC-GAN-like approach described in Section 2.2.6, the output of D has two components: $D_{rg} \in \mathbb{R}$ and $D_y \in \mathbb{R}^N$. Therefore, a multi-class cross-entropy regularization of real and generated samples is added 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 [45]. To the generator loss, we only add the multi-class cross-entropy regularization of generated samples. Consequently, the losses minimized in the proposed framework are:

Table 3.3: Classifier architecture. L, and N correspond to the light curve length and number of classes respectively and they vary depending on the selected dataset as mentioned in Section 3.1. 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}.

Input	$x \in \mathbb{R}^L$ $\phi_t \in \mathbb{R}^L$
Conv. Block	$2 \to 32$
Conv. Block	$32 \to 64$
Conv. Block	$64 \to 128$
Conv. Block	$128 \to 64$
Conv. Block	$64 \to 64$
Dense	$\lceil L/32 \rceil \times 64 \to 100$
BN, ReLU, Dropout	$100 \to 100$
Dense, Softmax	$100 \to N$

Convolution Block ($p_s = 2, k_s = 3, c_{in}, c_{out}$)	
Block Input	$l_i \times c_{in}$
1-D Convolution, BN	$l_i \times c_{in} \to l_i \times c_{out}$
Max-pooling, ReLU	$l_i \times c_{out} \to \lceil l_i/2 \rceil \times c_{out}$

$$\widetilde{L_D} = L_D + \xi(\mathop{\mathbb{E}}_{x_r \sim P_r}[H_r] + \mathop{\mathbb{E}}_{x_g \sim P_g}[H_g]) + \varepsilon \mathop{\mathbb{E}}_{x_r \sim P_r}[D_{rg}(x_r)^2] \tag{3.1}$$

$$\widetilde{L_G} = L_G + \xi \mathop{\mathbb{E}}_{x_g \sim P_g}[H_g] \tag{3.2}$$

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 $\varepsilon = 1$ control the strength of each regularization term.

We perform $n_{disc} = 5$ discriminator iterations per generator iteration, and train for 400K generator iterations using Adam optimizer with $\alpha = 0.0001, \beta_1 = 0.5, \beta_2 = 0.9$. At training time, we compute the Exponential Moving Average (EMA) with decay $\delta = 0.999$ for the generator weights, to be used when generating samples for evaluation. A full description of the GAN architecture is shown in Table 3.4.

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.

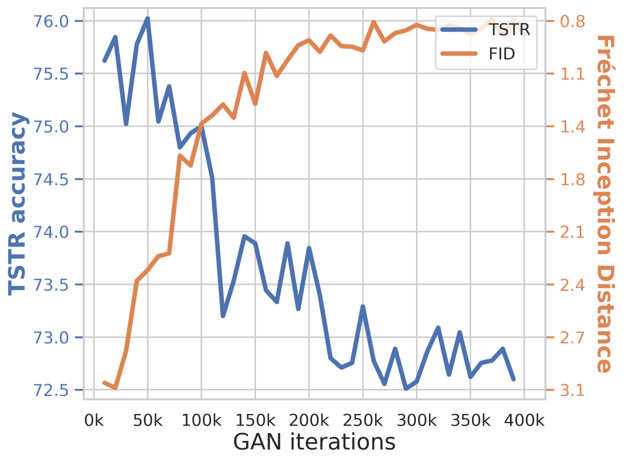
Table 3.4: GAN architecture. $\ell, L,$ and N correspond to the latent space dimensionality, light curve length and number of classes respectively, which depend on the selected dataset as mentioned in Sections 3.1 and 3.3. The fixed block parameters s and k_s stand for stride and kernel size respectively, 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_{in} and c_{out}.

(a) Generator

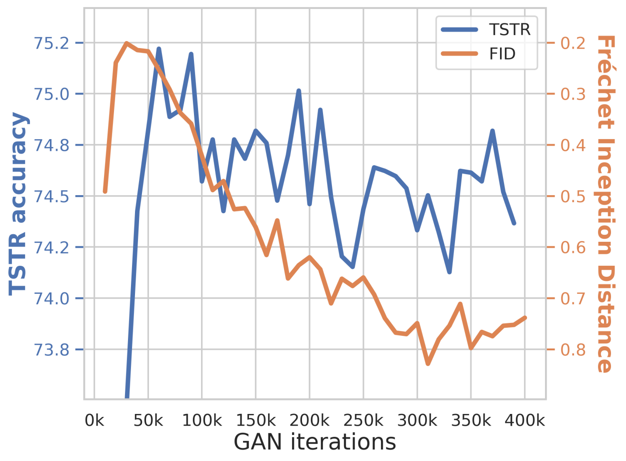
Input	$z \in \mathbb{R}^{\ell}$ $\bar{z} \in \mathbb{R}^{L+1+N}$
Dense, ReLU	$\ell + (L+1+N) \rightarrow 4 \times 1024$
Deconv. Block	$1024 \rightarrow 512$
Deconv. Block	$512 \rightarrow 256$
Deconv. Block	$256 \rightarrow 128$
Deconv. Block	$128 \rightarrow 64$
Deconv. Block	$64 \rightarrow 1$
Tanh $\cdot s$	$L \times 1$
Deconvolution Block $(s = 2, k_s = 5, c_{in}, c_{out})$	
Block Input	$l_i \times c_{in}$
1-D Deconvolution	$l_i \times c_{in} \rightarrow 2l_i \times c_{out}$
ReLU	$2l_i \times c_{out}$

(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 $(s = 2, k_s = 5, c_{in}, c_{out})$	
Block Input	$l_i \times c_{in}$
1-D Convolution	$l_i \times c_{in} \rightarrow \lceil l_i/2 \rceil \times c_{out}$
LeakyReLU	$\lceil l_i/2 \rceil \times c_{out}$



(a) Catalina



(b) ZTF

Figure 3.3: Evolution of the validation TSTR accuracy and FID over the course of GAN training for the different datasets. Both scores were computed every $10k$ iterations of a single GAN model. The computation of the FID was done with $50k$ generated samples divided into 10 batches and the entire real dataset, as suggested in [63].

3.3.5 Preliminary experiment: The u-GAN

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

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

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 3.3 shows the evolution of the validation TSTR accuracies and FID scores every $10k$ iterations. Since computing TSTR accuracies involves training multiple classifiers, evaluating this score more frequently is unfeasible.

The preliminary experiment shown in Figure 3.3 raises two major concerns that will be addressed in the following sections:

a) 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 2.2.8.

b) 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 3.3a.

The behavior detailed in a) can be understood as follows: in a balanced dataset such as $\mathcal{D}^u_{train}$, 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 $\mathcal{D}_{train}$. As the imbalance grows, 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 $10k$ iterations. Hence, the best model selected by this metric could be sub-optimal if the decay occurs suddenly, which motivates the proposed **resampling block** explained in Section 3.3.6.

The discrepancy described in b), although undesirable, is not surprising; it was also reported in [61], and it is completely plausible considering the limitations of FID related to mode dropping and mode inventing mentioned in Section 2.2.9. 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 datasets, and motivates the proposed $\mathcal{G}$ **-score** for model selection explained in Section 3.3.7.

3.3.6 Resampling block

Motivated by the rapid GAN overfitting shown in Figure 3.3, we propose a resampling operation that can successfully delay the occurrence of this behavior. A updated version of Figure 3.2 that includes the use of the resampling block is presented in Figure 3.4.

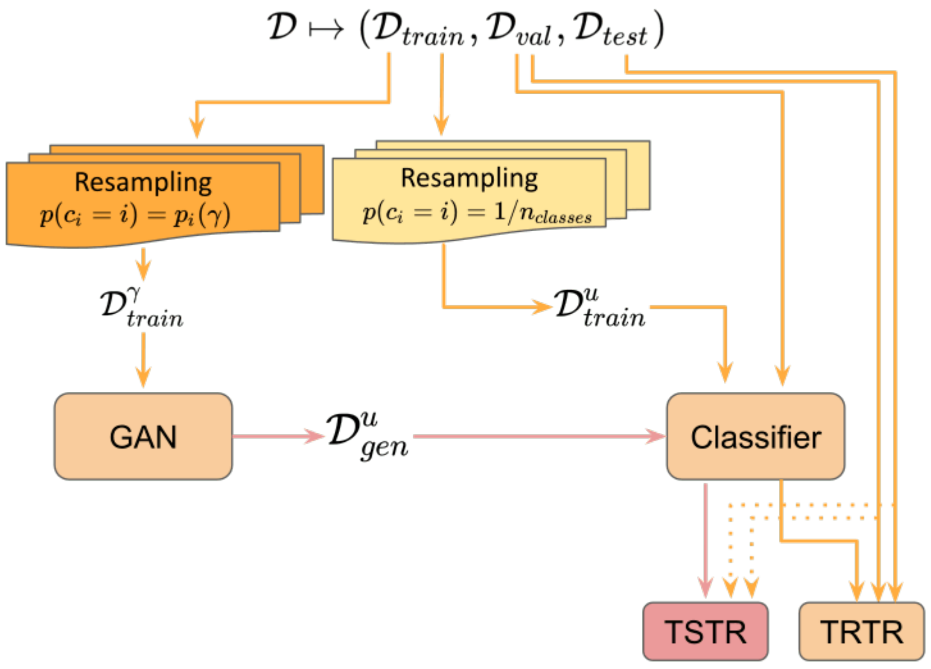

$\mathcal{D} \mapsto (\mathcal{D}_{train}, \mathcal{D}_{val}, \mathcal{D}_{test})$
Resampling
$p(c_i = i) = p_i(\gamma)$
Resampling
$p(c_i = i) = 1/n_{classes}$
$\mathcal{D}^{\gamma}_{train}$
$\mathcal{D}^{u}_{train}$
GAN
$\mathcal{D}^{u}_{gen}$
Classifier
TSTR
TRTR

Figure 3.4: Diagram of the methodology including the γ-resampling block

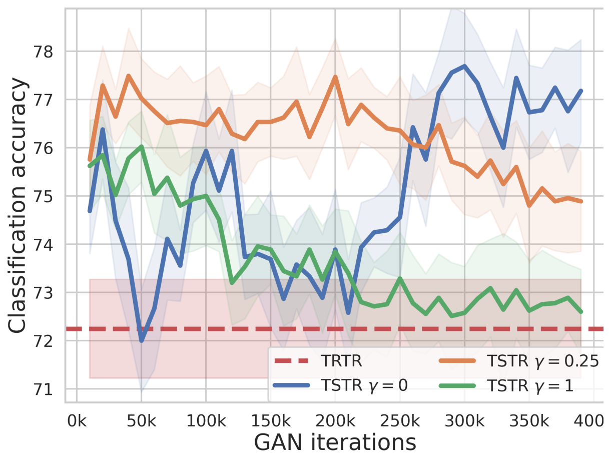
The resampling operation consists of continuously drawing samples from the N classes of a dataset $\mathcal{D}$, to modify its class distribution. Let S be the number of samples of $\mathcal{D}$. We start by splitting $\mathcal{D}$ into N sub-datasets $\{\mathcal{D}_i\}_{i=1}^{N}$ of size $\{S_i\}_{i=1}^{N}$, where each dataset $\mathcal{D}_i$ only contains samples from the i-th class. From each sub-dataset, 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 p_i 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 non-uniform distributions. To illustrate this clearly, we describe two edge cases. On the one hand, we could leave the original class distribution unbalanced by setting $p_i = S_i/S$, 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 $p_i = 1/N$, which is how we get $\mathcal{D}^{u}_{train}$ from $\mathcal{D}_{train}$.

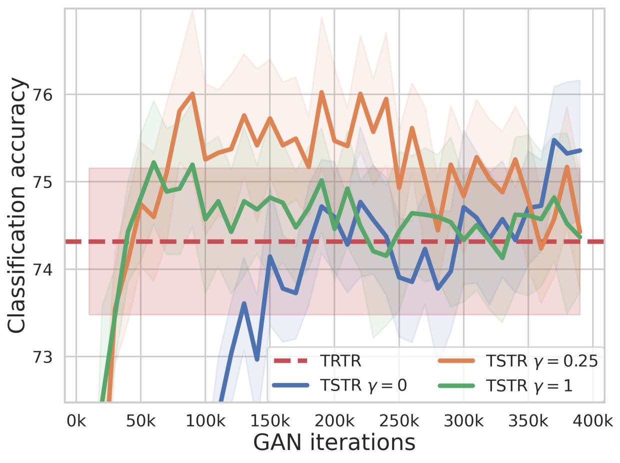
Apart from these two scenarios, we could also generate any dataset $\mathcal{D}^{\gamma}$ whose class distribution lies "in between" that of $\mathcal{D}$ and $\mathcal{D}^{u}$, created by linearly interpolating between the aforementioned probabilities:

$$p_i = \gamma \left(\frac{1}{N}\right) + (1-\gamma)\frac{S_i}{S}, \text{ where } 0 < \gamma < 1 \tag{3.3}$$

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 3.5. 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 (1 every 230 samples will be cepheids of the Catalina dataset, roughly 1 cepheid every 4 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 3.5 for $\gamma = 0.25$. A model trained with $\mathcal{D}^\gamma$ will be referred to as the "γ-GAN".



(a) Catalina



(b) ZTF

Figure 3.5: Evolution of the validation TSTR accuracy over the course of GAN training for different values of γ. The figure shows mean $\pm$ 1 standard deviation over 15 independent runs of the classifier and a single GAN model. The computation of both metrics was done every 10k GAN iterations

3.3.7 Model selection: The $\mathcal{G}$-score

As mentioned in Section 3.3.5, the behavior of TSTR accuracies shown in Figure 3.3 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).

The natural option for this metric would be FID, but as also shown in Figure 3.3a, 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 [57], 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$(\mathbf{D})$ and recall$(\mathbf{R})$. Figure 3.6 shows the results of computing the per-class density and recall metric for the Catalina dataset.

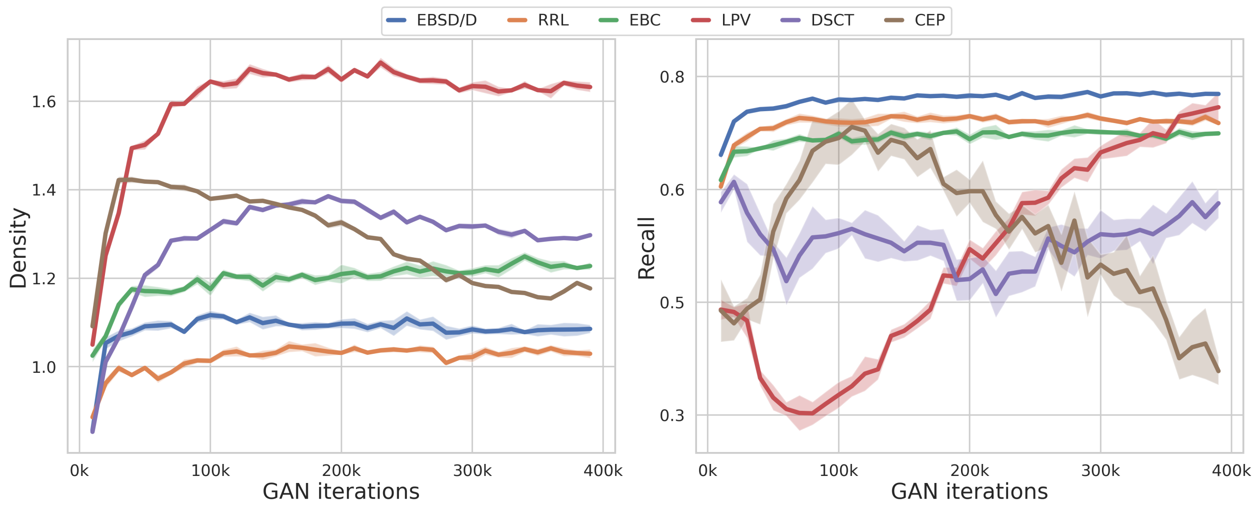


Figure 3.6: Class density($\mathbf{D}$) and recall($\mathbf{R}$) metrics of the Catalina dataset. Each curve shows mean $\pm$ 1 standard deviation over 5 computations of the metrics, for a single GAN model.

The fact that $\mathbf{D}$ values are not bounded by one is consistent with the formula presented in Equation 2.24 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 over-estimation of the real manifold mentioned in Section 2.2.9, due to sparse feature spaces. An illustration of this situation is shown in Figure 3.7, where the sparsity in the real distribution causes that the generated samples in average belong to more than $K = 2$ balls, leading to $\mathbf{D} = \frac{1}{4}(\frac{2}{2} + \frac{3}{2} + \frac{4}{2} + \frac{3}{2}) = 1.5$. Additionally, if we reduce the sparsity of the real distribution by removing the furthest sample (bottom left), we get $\mathbf{D} = \frac{1}{4}(\frac{1}{2} + \frac{2}{2} + \frac{3}{2} + \frac{2}{2}) = 1$.

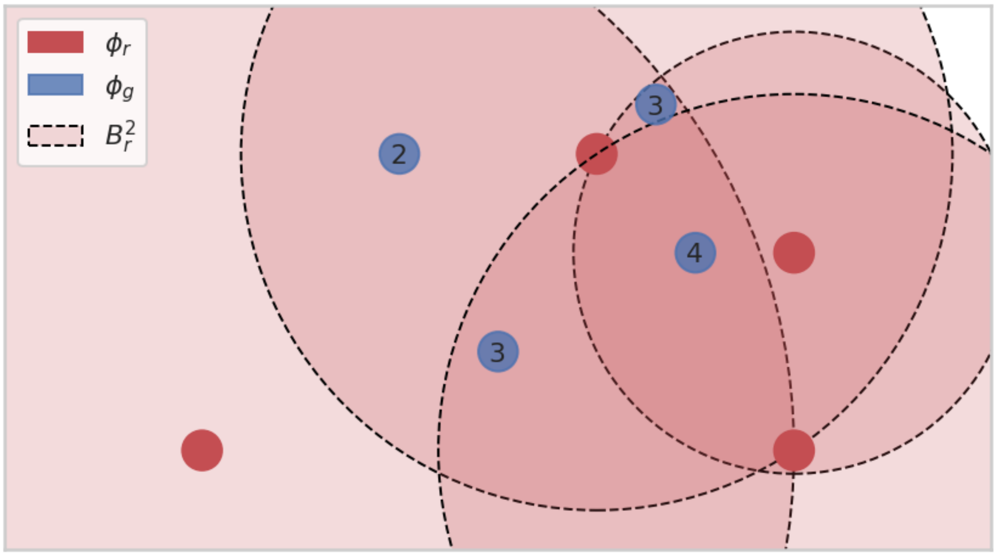


Figure 3.7: Two-dimensional scenario that illustrates a case in which $\mathbf{D}$ is not bounded by 1. The dashed lines show the regions B_r^2: circles around the real feature samples ϕ_r, with radii equal to the distance to their second nearest neighbors. The numbers inside each sample ϕ_g denote the number of circles that enclose the sample.

Since $\mathbf{R}$ is bounded between 0 and 1 by definition, the unbounded behavior of $\mathbf{D}$ is undesirable because it privileges $\mathbf{D}$ over $\mathbf{R}$ in any mean we compute between them. In addition, we find that $\mathbf{D}$ also presents a clear bias towards the less populated classes. To overcome these problems, we perform a per-class min-max normalization to $\mathbf{D}$ and $\mathbf{R}$ as follows:

$$\mathbf{D}'_i = \frac{\mathbf{D}_i - \mathbf{D}_i^{min}}{\mathbf{D}_i^{max} - \mathbf{D}_i^{min}} \tag{3.4}$$

$$\mathbf{R}'_i = \frac{\mathbf{R}_i - \mathbf{R}_i^{min}}{\mathbf{R}_i^{max} - \mathbf{R}_i^{min}} \tag{3.5}$$

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

After the class scores are normalized, we combine them in an equally weighted F-score described in Equation 3.6. 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 3.7.

$$F_i = \frac{2\mathbf{D}'_i\mathbf{R}'_i}{\mathbf{D}'_i + \mathbf{R}'_i} \tag{3.6}$$

$$\mathcal{G}\text{-score} = \frac{1}{N}\sum_i F_i \tag{3.7}$$

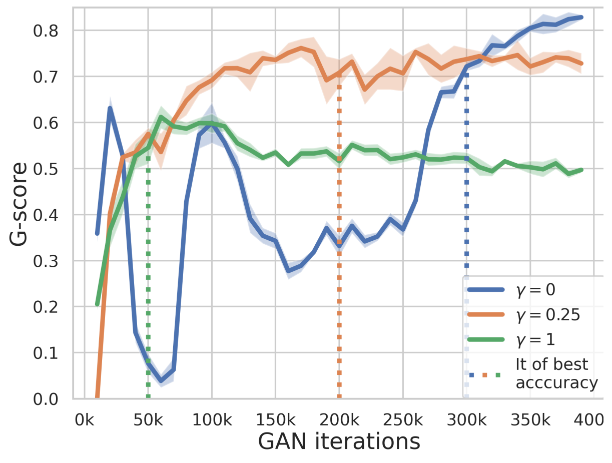
We prefer the balanced F-score over the F-score of the class means intending to weight equally majority and minority classes, as suggested by [75].

The results of computing the $\mathcal{G}$-score for multiple GANs trained with different values of γ are shown in Figure 3.8. As it can be seen, the $\mathcal{G}$-score curves and validation accuracy curves from Figure 3.5 seem to have a high correlation, which becomes more evident when analyzing the $\gamma = 0$ curve for the Catalina dataset.

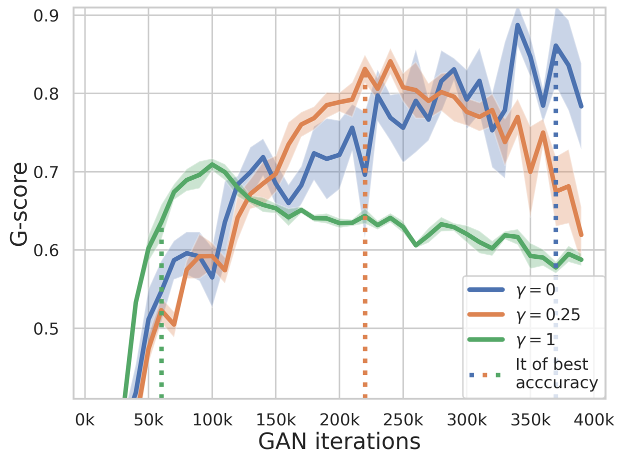
The importance of performing the normalization step to compute the $\mathcal{G}$-score can be evidenced in Figure 3.9, where this step has been omitted. Not performing this normalization could lead to unfortunate results, which becomes evident when analyzing $\gamma = 0.25$ and $\gamma = 1$ curves for the Catalina dataset. These curves suggest that sample quality improves if we continue training when in fact, Figure 3.5(a) shows that the classification accuracies at the last iterations are far from optimal.

3.3.8 Baselines

To evaluate our generated datasets 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 datasets. It is worth mentioning that the training sets used to compute the scores are all **balanced datasets**, either GAN-generated (TSTR) or real-augmented (TRTR).

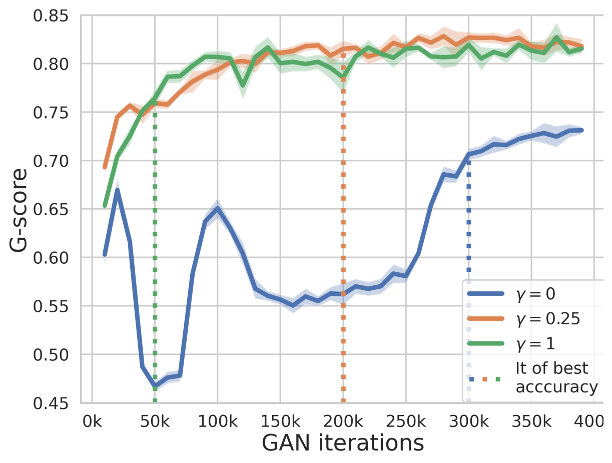


(a) Catalina

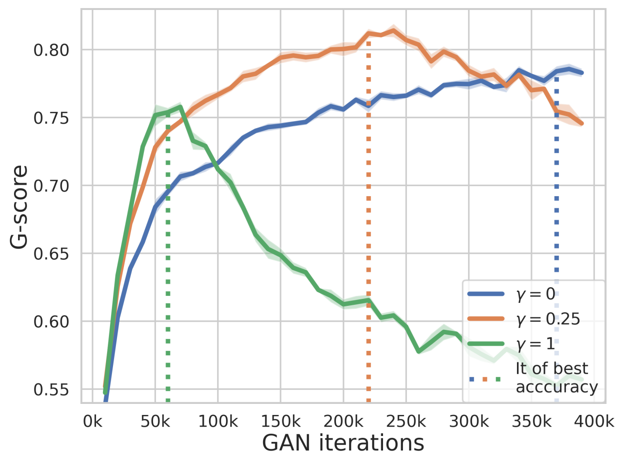


(b) ZTF

Figure 3.8: Evolution of the $\mathcal{G}$-score for different γ values over the course of GAN training for the different datasets. Each curve shows mean $\pm$ 1 standard deviation over 5 computations of the metrics, for a single GAN model.



(a) Catalina



(b) ZTF

Figure 3.9: Evolution of the unnormalized $\mathcal{G}$-score for different γ values over the course of GAN training for the different datasets. Each curve shows mean $\pm$ 1 standard deviation over 5 computations of the metrics, for a single GAN model.

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 pre-processed time-series with a fixed number of observations. Consequently, our augmentation methods consist of oversampling and different window-warping-based operations.

Oversampling

The oversampling augmentation corresponds to generating the balanced dataset $\mathcal{D}^u_{train}$ by repeating samples from the original dataset $\mathcal{D}_{train}$, using the resampling block described in Section 3.3.6.

Window-warping

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**.

Soft window-warping

We preserve the core idea of window-warping by designing expansions and contractions that do not increase the length of the time-series. 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 $f(t_1) \geq t_1$, $f(t_2) \leq t_2$. 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:

$$\begin{aligned} f(t_w) &= a + b \cdot \tanh\left(k(t_w - c)\right) \\ a &= c = (t_1 + t_2)/2 \\ b &= (t_2 - t_1)/2 \end{aligned} \tag{3.8}$$

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 modulates the strength of the expansions or contractions by modifying the saturation degree of the $\tanh(\cdot)$, producing expansions when saturated and contractions otherwise. Since we do not intend to benefit expansions over contractions or vice-versa, and considering the exponential nature of the $\tanh(\cdot)$, k is sampled from a log-uniform distribution.

An example of performing such sampling is shown in Figure 3.10 where k was chosen from a log-spaced array in the interval $\left[\frac{1}{4b}, \frac{4}{b}\right]$, with $t_1 = 5$ and $t_2 = 10$. As it can be seen, the values of k produce that the number of tangents producing contractions is comparable to the ones producing expansions.

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

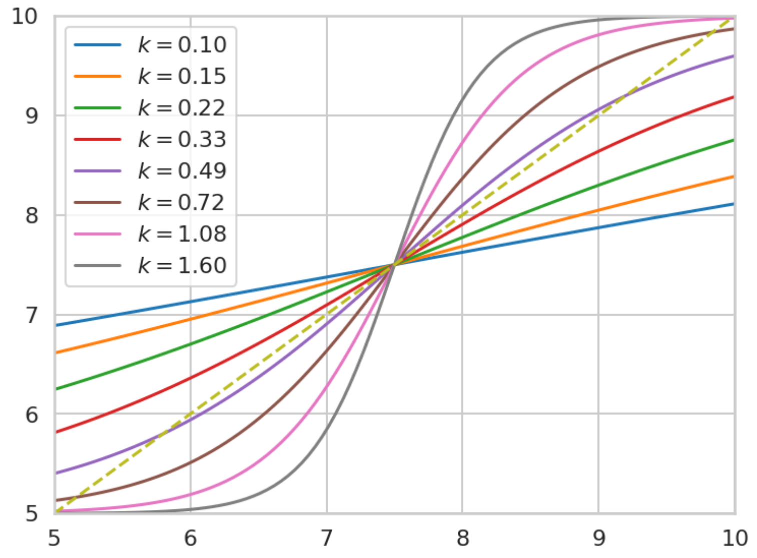


Figure 3.10: Behavior of $f(t) = a + b \cdot \tanh(k(t - c))$ with multiple k chosen form a log-spaced array in the interval $[t_1, t_2] = [5, 10]$. The values of a, b and c where determined according to Equation 3.9

.

monotonous, t_1, t_2 are the minimum and maximum values in the window respectively. Hence, the natural extension to the signal axis is:

$$\begin{aligned} f(x_w) &= a + b \cdot \tanh\left(k(x_w - c)\right) \\ m_1 &= \min_{t \in t_w} x(t) \\ m_2 &= \max_{t \in t_w} x(t) \\ a &= c = (m_1 + m_2)/2 \\ b &= (m_2 - m_1)/2 \end{aligned} \tag{3.9}$$

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 3.11. The result of simultaneously applying these two transformations will be referred to as **soft mixed-warping**.

3.3.9 Data Augmentation experiment: mixing generated and real data

Apart from evaluating the proposed generative models by generating purely synthetic datasets to be used as training sets for the classifiers (TSTR score), it is interesting how this score would behave if we somehow combine the generated data of the best model with real data. This experiment, whose results are shown in Section 4.1, considers the following approach: we add a resampling block to the methodology Figure 3.4, which will now sample from $\mathcal{D}^u_{train}$ with a probability of p_{real}, and from $\mathcal{D}^u_{gen}$ with $1 - p_{real}$.

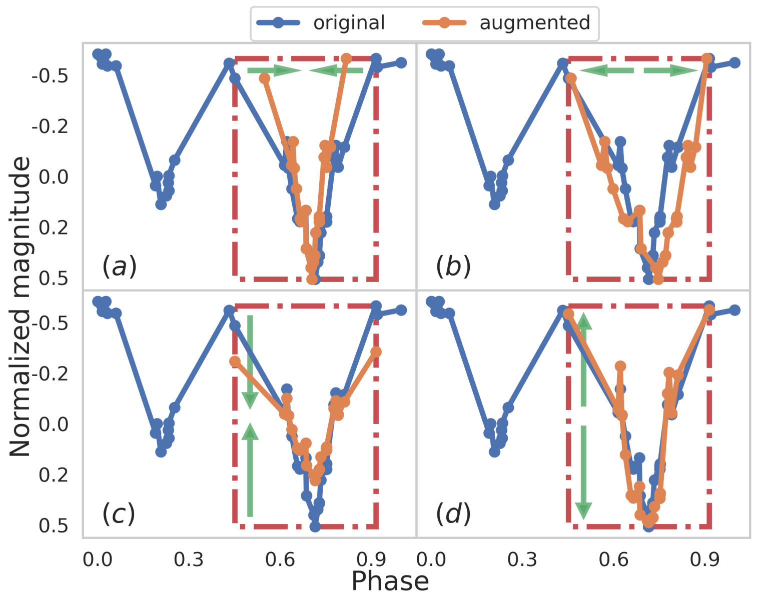


Figure 3.11: Examples of the soft window-warping transformations for an eclipsing binary of the ZTF dataset. (a) Soft time-warping contraction. (b) Soft time-warping expansion. (c) Soft magnitude-warping contraction. (d) Soft magnitude-warping expansion.

3.3.10 Details for performing multiple experiments

When reporting experiments that require statistically significant differences, such as Table 4.1, it is necessary to run multiple experiments.

Considering that our framework involves many sources of variability when running multiple experiments, it is necessary to reduce unnecessary variability sources that could violate the **Ceteris Paribus** principle and contaminate the results.

We first fix the partition of $\mathcal{D}$ into $\mathcal{D}_{train}$, $\mathcal{D}_{val}$, and $\mathcal{D}_{test}$ of Figures 3.2 and 3.4 to accomplish this. Secondly, though very stable due to the good-natured properties of the WGAN mentioned in Section 2.2.7, we also remove the stochasticity of training multiple trials of the same GAN model to evaluate the classification results.

Apart from these two processes, everything in Figures 3.2 and 3.4 is performed independently for every trial; this includes using a GAN model to generate different training sets for every trial, which remains a source of variability. Additionally, there is also variability in how the datasets are resampled according to the probabilistic approach to select samples from different classes and the shuffle operations involved within each class dataset.

As a side note, training multiple GANs trials would significantly increase computation time because training the GANs is computationally expensive, and because multiple classifiers would need to be trained for every single GAN trial.

Chapter 4

Results and Discussion

This chapter provides the results of this work and their corresponding analysis. First, the results obtained by using our methodology are exposed in Section 4.1, including a visualization of the generated light curves for the ZTF dataset and a comparison of the classification performance of the different models and baselines described in Chapter 3. In addition, a detailed analysis of the results and interpretation of them is provided in Section 4.2.

4.1 Results

4.1.1 Generated Samples

This section explores the generated samples by visual inspection, a mandatory step when training generative models. Even though we only show this inspection for the best performing model trained on the ZTF dataset, it is highly recommendable to extensively visualize the generated samples to get a real insight into the sample quality of the models. Figure 4.1 shows some GAN-generated light curves for the ZTF dataset. The conditional vector $\bar{z}$ used to generate these samples considers phases, amplitude, and class of the real data shown in the first two columns. The curves in these two columns were intentionally selected to have similar amplitudes close the median amplitude of their corresponding class.

According to the conditional parameters, and as it can be seen, most of the generated samples preserve the desired 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 4.6), 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 4.2.1.

To further illustrate the abilities of our model to condition on different amplitudes, Figure 4.2 shows a variation of this conditional parameter. The curves in the first two columns were selected to have amplitudes close to the low and high ends of the amplitude distribution of the corresponding class. The generated samples are conditioned on progressively increasing amplitudes, moving from the left to the rightmost column (see some of the amplitudes explicitly shown in the top row). As we can see, the generated samples follow the increasing

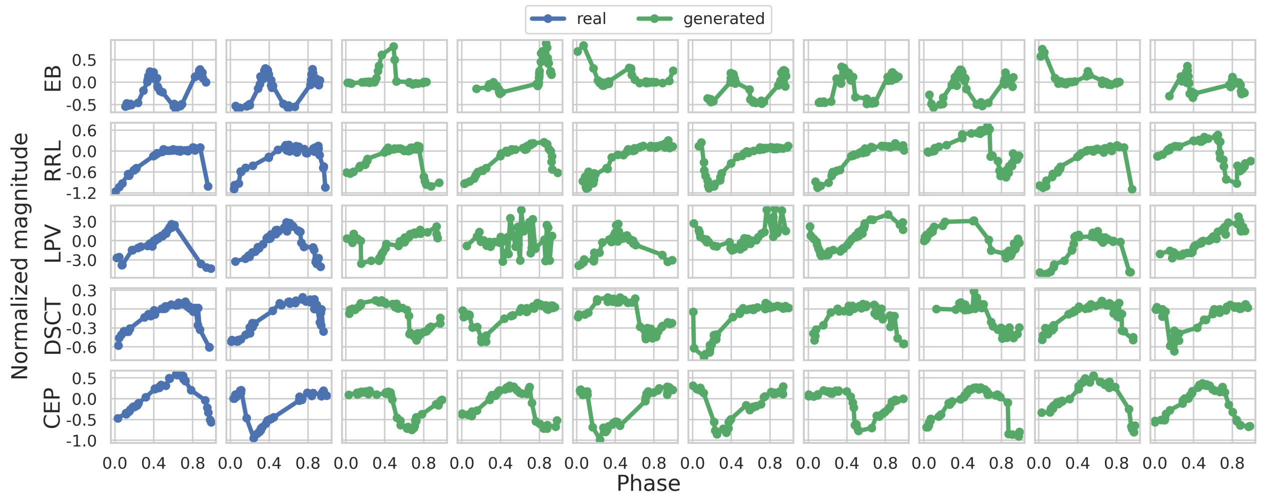


Figure 4.1: Real and generated light curves of the ZTF dataset. Each row represent a different class. The synthetic curves in green were conditioned on the attributes (phases, class and amplitude) of the corresponding real curves in blue.

amplitude pattern they were conditioned on, allowing for generating datasets with diverse amplitude distributions.

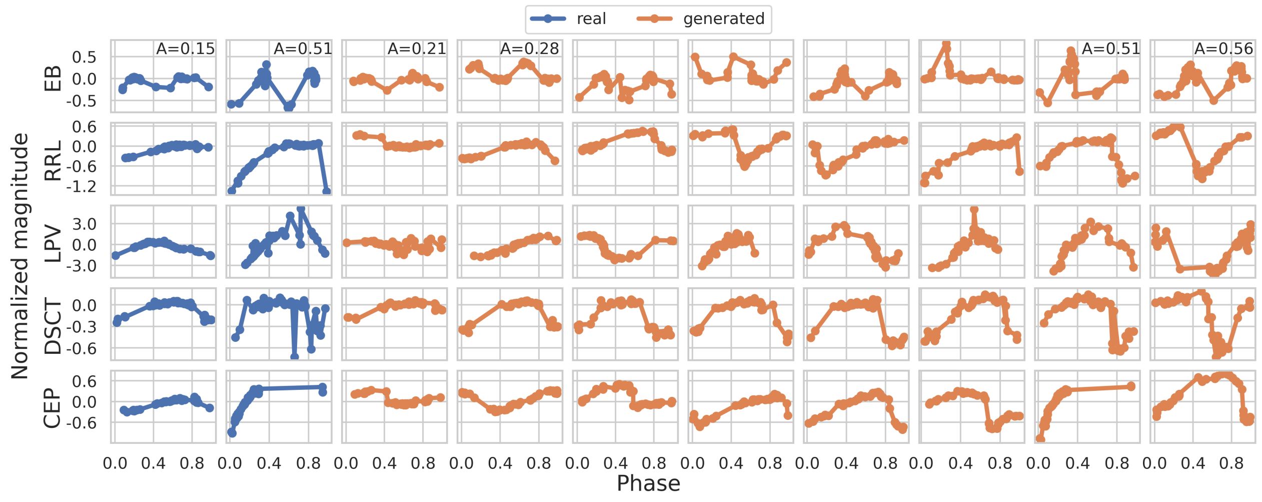


Figure 4.2: Real and generated light curves of the ZTF dataset with different amplitudes specified for the top row as A. Each row represents a different class. The real curves in blue were selected to have amplitudes in the low and high end of the class. The synthetic curves in green were conditioned on the phases and class of the corresponding real curves in blue, and on amplitudes that cover the entire range of the class amplitudes.

4.1.2 Classification

The test classification accuracies obtained by using different training sets are shown in Table 4.1. The first four rows (rows A-D) show TRTR classification results when training on real data that has been augmented with the random transformations described in Section 3.3.8. The soft-warping transformations (rows B-D) are applied to the dataset previously balanced by oversampling. The following four rows (rows E-H) show TSTR classification results when training purely 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. In addition, row I shows the classification accuracy obtained by mixing synthetic data from the models in row H with real data, as explained in Section 3.3.9.

As Table 4.1 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 (u or γ-GAN) 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 datasets, 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 datasets). Furthermore, for each dataset, 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 dataset by the γ-GAN model.

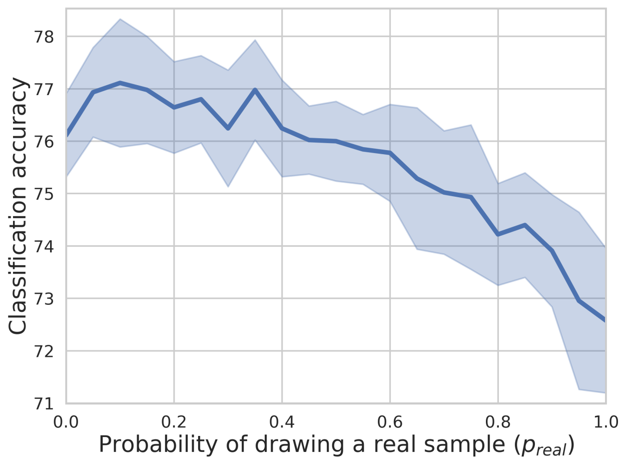
Interestingly, the combination of the proposed γ-GAN + $\mathcal{G}$-score obtains the best classification accuracies among models that **use only synthetic data**, statistically outperforming all existing methods for ZTF dataset, and all but one (γ-GAN + val. accuracy in row G) for the case of the Catalina dataset.

Finally, before understanding the results shown in row I, it is necessary to introduce Figure 4.3, which shows the validation classification accuracies for different combinations of real and synthetic. The model in row I, includes synthetic data from the previous best model (γ-GAN + $\mathcal{G}$-score in row H) and real data, mixed with the probability p_{real} that gave the maximum validation accuracy in Figure 4.3. Although classification accuracies of this model numerically outperform all previous existing models, the p-values with respect to row H show that the improvement is not significantly better than using only synthetic data

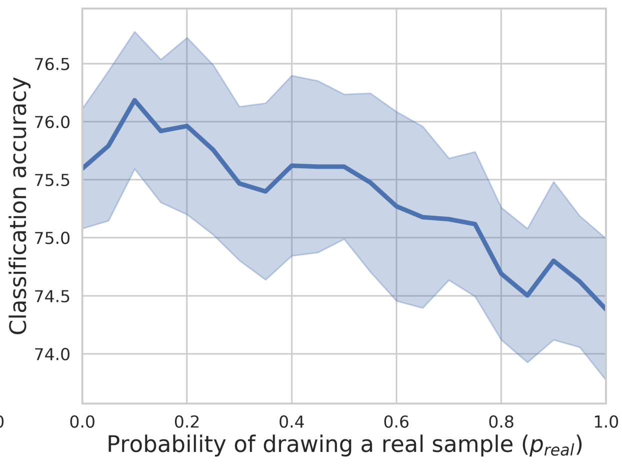
For completeness and to compare the predictions between the models, the confusion matrices of the oversampling baseline (row A) and the best performing model (row I) are provided in Figures 4.4 and 4.5. Interestingly, if we compare the two models, the percentage of correct predictions always improves, except for the RR Lyrae class in both datasets.

Table 4.1: Classification accuracy of the different augmentation methods on test datasets. 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. All the models were trained on a single GPU (NVIDIA GeForce GTX 1080 Ti), requiring approximately 17.3 hours for a GAN model and 3.5 minutes for an ensemble of classifiers.

			Catalina					ZTF				
		Method	Accuracy [%]	p value				Accuracy [%]	p value			
				A					A			
TRTR	A	Oversampling	73.44±1.22					72.61±0.69				
TRTR	B	Soft time-warping	74.06±1.04	.145				72.69±0.99	.786			
TRTR	C	Soft mag-warping	73.64±1.79	.723				72.45±0.70	.533			
TRTR	D	Soft mixed-warping	73.82±1.50	.452				72.53±0.69	.753			
TSTR		u-GAN		A	E	F	G		A	E	F	G
TSTR	E	Val Acc	75.97±0.94	<.001				74.17±0.62	<.001			
TSTR	F	$\mathcal{G}$-score	76.28±0.74	<.001	.324			73.79±0.50	<.001	.075		
TSTR		**γ-GAN**										
TSTR	G	Val Acc	76.86±1.09	<.001	.024	.102		74.37±0.51	<.001	.342	.003	
TSTR	H	$\mathcal{G}$-score	**76.97 ± 0.79**	<.001	.004	.041	.752	**74.94 ± 0.44**	<.001	<.001	<.001	.002
		γ-GAN + Real		A	E	F	H		A	E	F	H
	I	$\mathcal{G}$-score	**77.06 ± 0.75**	<.001	.001	.008	.754	**75.31 ± 0.57**	<.001	<.001	<.001	.056

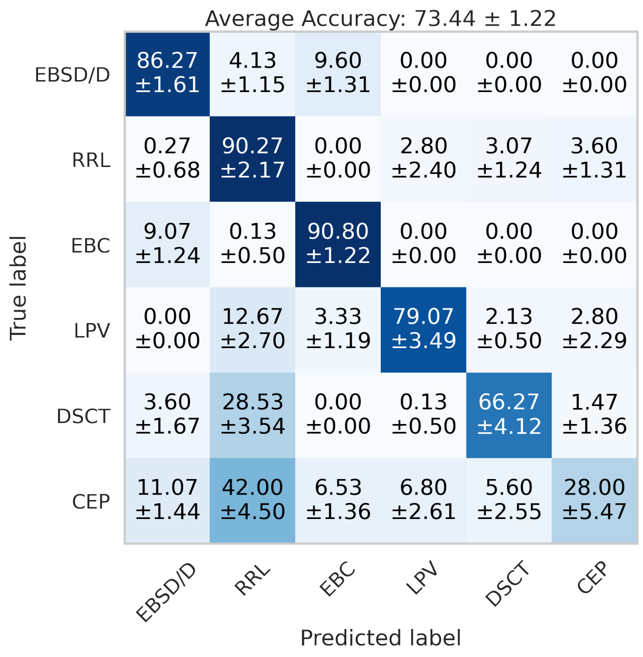


(a) Catalina



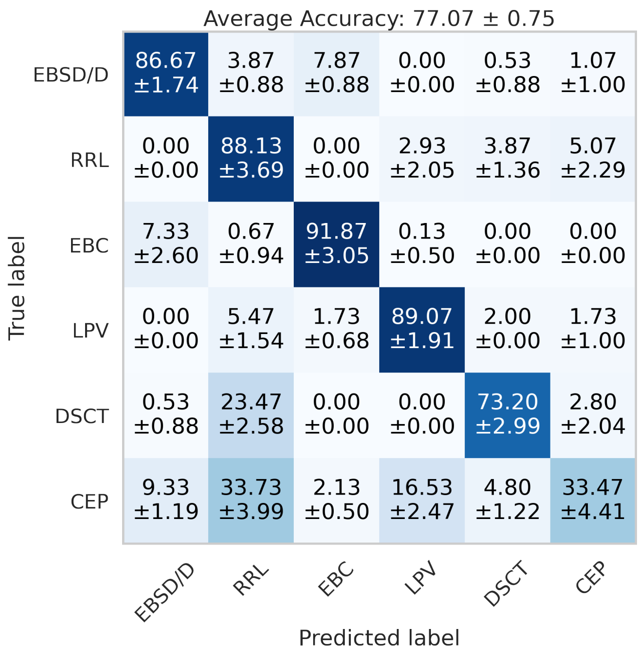
(b) ZTF

Figure 4.3: Validation classification accuracy for the data augmentation experiment described in Section 3.3.9. The figure shows mean ± 1 standard deviation over 15 independent runs of the classifier and a single GAN model, for different values of p_{real}.


Average Accuracy: 73.44 ± 1.22

True label \ Predicted label	EBSD/D	RRL	EBC	LPV	DSCT	CEP
EBSD/D	86.27 ±1.61	4.13 ±1.15	9.60 ±1.31	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
RRL	0.27 ±0.68	90.27 ±2.17	0.00 ±0.00	2.80 ±2.40	3.07 ±1.24	3.60 ±1.31
EBC	9.07 ±1.24	0.13 ±0.50	90.80 ±1.22	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
LPV	0.00 ±0.00	12.67 ±2.70	3.33 ±1.19	79.07 ±3.49	2.13 ±0.50	2.80 ±2.29
DSCT	3.60 ±1.67	28.53 ±3.54	0.00 ±0.00	0.13 ±0.50	66.27 ±4.12	1.47 ±1.36
CEP	11.07 ±1.44	42.00 ±4.50	6.53 ±1.36	6.80 ±2.61	5.60 ±2.55	28.00 ±5.47

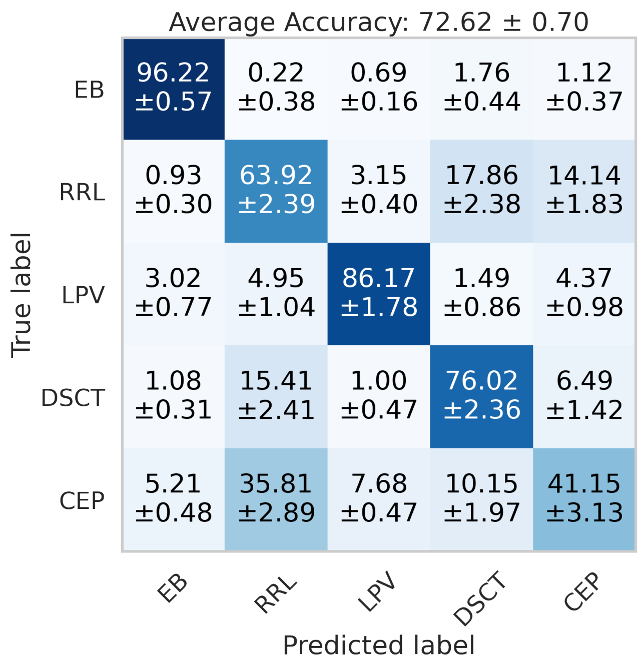
(a) Real data only


Average Accuracy: 77.07 ± 0.75

True label \ Predicted label	EBSD/D	RRL	EBC	LPV	DSCT	CEP
EBSD/D	86.67 ±1.74	3.87 ±0.88	7.87 ±0.88	0.00 ±0.00	0.53 ±0.88	1.07 ±1.00
RRL	0.00 ±0.00	88.13 ±3.69	0.00 ±0.00	2.93 ±2.05	3.87 ±1.36	5.07 ±2.29
EBC	7.33 ±2.60	0.67 ±0.94	91.87 ±3.05	0.13 ±0.50	0.00 ±0.00	0.00 ±0.00
LPV	0.00 ±0.00	5.47 ±1.54	1.73 ±0.68	89.07 ±1.91	2.00 ±0.00	1.73 ±1.00
DSCT	0.53 ±0.88	23.47 ±2.58	0.00 ±0.00	0.00 ±0.00	73.20 ±2.99	2.80 ±2.04
CEP	9.33 ±1.19	33.73 ±3.99	2.13 ±0.50	16.53 ±2.47	4.80 ±1.22	33.47 ±4.41

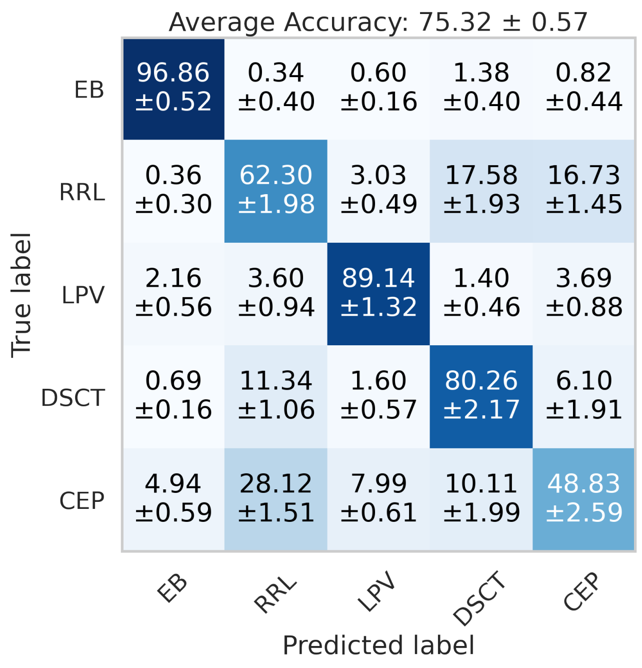
(b) Real and synthetic data

Figure 4.4: Test classification results for the Catalina dataset. (a) Classifier of row A in Table 4.1. (b) Classifier of row I in Table 4.1. We report the mean and standard deviation calculated over 15 independent runs


Average Accuracy: 72.62 ± 0.70

True label \ Predicted label	EB	RRL	LPV	DSCT	CEP
EB	96.22 ±0.57	0.22 ±0.38	0.69 ±0.16	1.76 ±0.44	1.12 ±0.37
RRL	0.93 ±0.30	63.92 ±2.39	3.15 ±0.40	17.86 ±2.38	14.14 ±1.83
LPV	3.02 ±0.77	4.95 ±1.04	86.17 ±1.78	1.49 ±0.86	4.37 ±0.98
DSCT	1.08 ±0.31	15.41 ±2.41	1.00 ±0.47	76.02 ±2.36	6.49 ±1.42
CEP	5.21 ±0.48	35.81 ±2.89	7.68 ±0.47	10.15 ±1.97	41.15 ±3.13

(a) Real data only


Average Accuracy: 75.32 ± 0.57

True label \ Predicted label	EB	RRL	LPV	DSCT	CEP
EB	96.86 ±0.52	0.34 ±0.40	0.60 ±0.16	1.38 ±0.40	0.82 ±0.44
RRL	0.36 ±0.30	62.30 ±1.98	3.03 ±0.49	17.58 ±1.93	16.73 ±1.45
LPV	2.16 ±0.56	3.60 ±0.94	89.14 ±1.32	1.40 ±0.46	3.69 ±0.88
DSCT	0.69 ±0.16	11.34 ±1.06	1.60 ±0.57	80.26 ±2.17	6.10 ±1.91
CEP	4.94 ±0.59	28.12 ±1.51	7.99 ±0.61	10.11 ±1.99	48.83 ±2.59

(b) Real and synthetic data

Figure 4.5: Test classification results for the ZTF dataset. (a) Classifier of row A in Table 4.1. (b) Classifier of row I in Table 4.1. We report the mean and standard deviation calculated over 15 independent runs

4.2 Discussion

4.2.1 Quality of generated samples

An example of individual samples that may not satisfy desired properties could be the top right light curve in Figure 4.2. This eclipsing binary, supposed to have an amplitude of 0.56, clearly has a lower amplitude than the real eclipsing binary in the second column of the same figure, which has an amplitude of 0.51. This indicates that even though performing decently for most of the amplitudes, the model fails at conditionally generating samples with amplitudes close to the distribution's limits. This could be explained by the lack of samples belonging to these distribution regions or the lack of penalization of the models for such cases. In fact, we did try adding an explicit amplitude error regularization, but it caused the models to perform terribly, obtaining classification accuracies close to 40%.

Regarding the samples of Figure 4.1, although the generated samples look realistic in general, there can be samples that present noisy artifacts, making them not the best candidates for the classes they intend to represent. While these artifact samples could be easily avoided by applying truncation techniques on the latent space of G, it would not be informative about the quality of the individual samples, impeding us from learning what makes a sample look realistic.

A metric that could help us measure individual sample quality is the *realism score* [65], 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 $\mathbf{\Phi}_r = \{\phi_r\}$, the similarity between ϕ_g and the real manifold Φ_r is calculated as:

$$\mathcal{R}(\phi_g, \Phi_r) = \max_{\phi_r \in \mathbf{\Phi_r}} \left\{ \frac{\|NND_k(\phi_r)\|_2}{\|\phi_r - \phi_g\|_2} \right\} \tag{4.1}$$

where $NND_k(\phi)$ is the distance from ϕ to its k-th nearest neighbor within the corresponding manifold. Equation 4.1 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 dataset by *realism score* is shown in Figure 4.6.

Because it can successfully identify artifacts that could be filtered out of the dataset, 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 that applying this filtering to our generated datasets only deteriorates the results as it can be seen in Figure 4.7. We hypothesize that these artifacts, although undesirable, are not crucial when defining the decision boundaries of the problem, but they could act as regularizers. Moreover, strongly filtered datasets cause a drop in the classification accuracy, probably caused by their over-constrained diversity the resulting datasets.

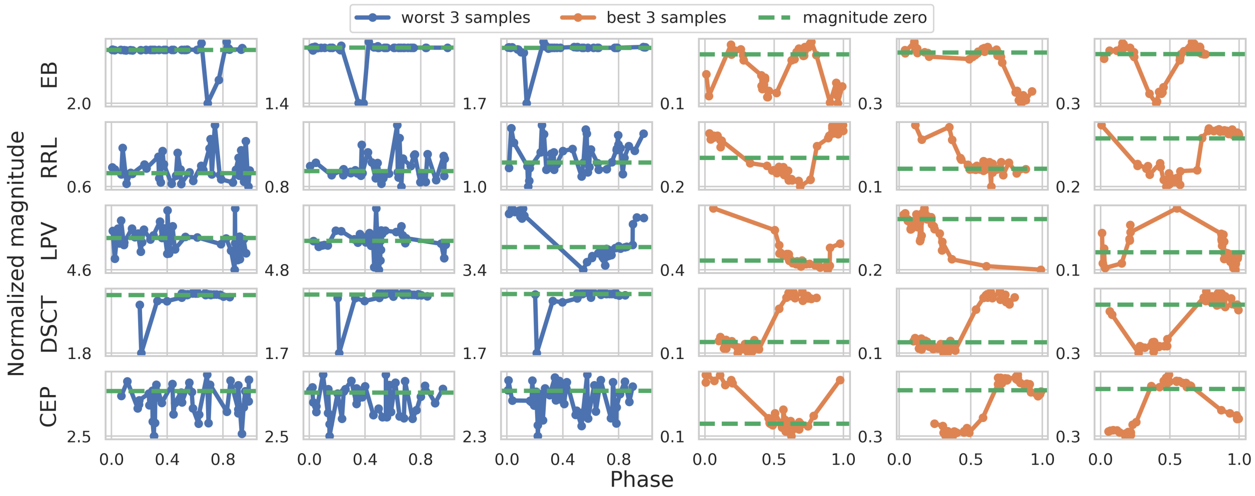
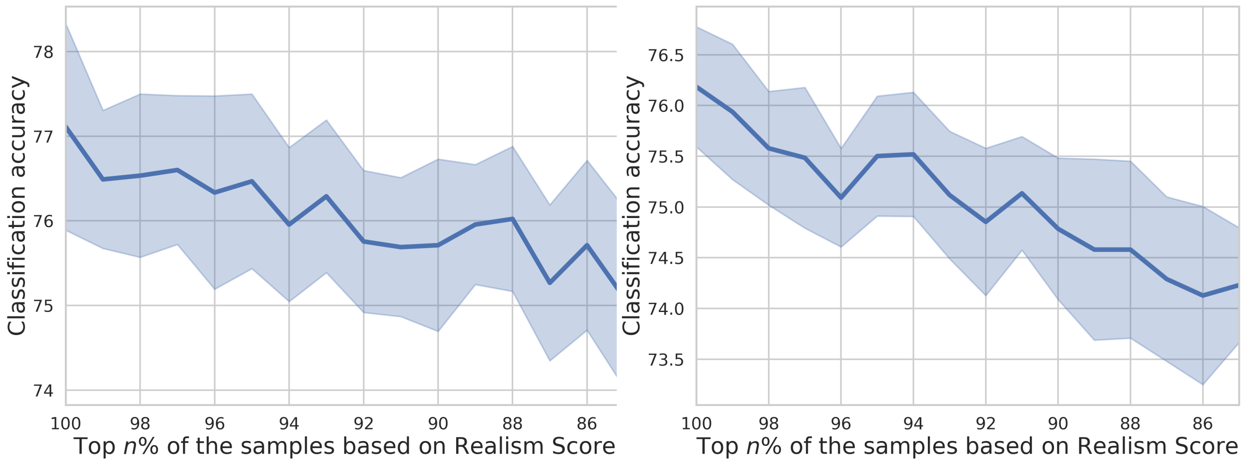


Figure 4.6: Realism score ranking of the ZTF generated light curves. To rank the samples, we first generated a replication of the D^{u}_{train}, computed their realism score, and then selected the best and worst samples from the sorted realism scores.



(a) Catalina

(b) ZTF

Figure 4.7: Validation classification accuracy for the realism score-filtering. The classifiers were trained on real data augmented with synthetic data previously filtered based on different realism score thresholds. The figure shows mean $\pm$ 1 standard deviation over 15 independent runs of the classifier and a single GAN model.

4.2.2 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 dataset 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 3.5 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 3.5b. 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 six 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 $\mathcal{G}$-score compares $\mathcal{D}^u_{gen}$ to the entire training set $\mathcal{D}_{train}$, the validation accuracy score is computed on the small dataset $\mathcal{D}_{val}$ 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.

Data augmentation The results in Figure 4.3 are quite illustrative of the fact that using synthetic data can be beneficial for the training of the classifiers. Moreover, there is a shared pattern in both datasets: the performance of the models increases to a point and then drops as we add more and more real data at the expense of synthetic data. The trade-off between real and synthetic data is interesting, and it could be understood if our generative models provide a good -but not perfect- approximation of the real data distribution. Hence, combining the data generated by these models with the original data they try to recreate could still provide incremental benefits for classification.

Confusion matrices The results of the confusion matrices shown in Figures 4.4 and 4.5 can help us understand where our generated data is beneficial. First, it is interesting to see how adding synthetic data to the most populated class in both datasets adds only a marginal benefit. Second, classes that benefit from synthetic data are coincidentally the three less populated classes of each dataset. These two facts *could* suggest that the great advantage that our models add is the diversity of the generated samples, which is beneficial for classes with limited diversity and irrelevant for diverse classes.

4.2.3 Alternative to $\mathcal{G}$-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**&**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*[1] of the raw macro-density and macro-recall. For example, in the case of the Catalina dataset, doing so would imply evaluating approximately 1/4 of total candidates.

The disposition of the optima for the Catalina dataset is shown in Figure 4.8. Interestingly, the model selected with the validation accuracy criterion is in the sub-optimal region which supports the idea of overfitting explained in Section 4.2.2. 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**&**R** scores and then evaluate these candidates based on the validation TSTR score to choose an operating point.

[1]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.

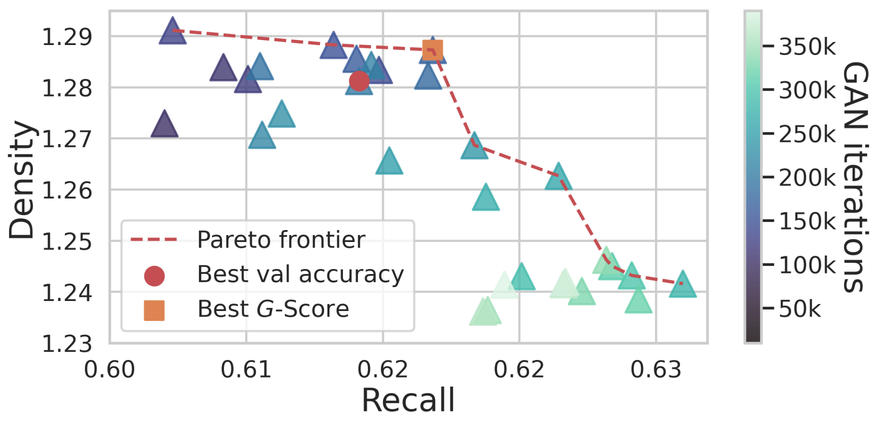


Figure 4.8: Macro density and recall metrics for the Catalina dataset. Each point corresponds to the average of 5 independent computation of density and recall for a single GAN model.

Chapter 5

Conclusions

In this thesis, 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 datasets. The designed generative model, consisting of a conditional WGAN, is able to deal with irregularly sampled time-series and perform conditional generating on other features of interest, such as the class and amplitude of the time-series.

Motivated by the rapid overfitting of our generative model in the presence of imbalanced datasets, we proposed a resampling technique (γ-resampling) to mitigate this behavior. Also, inspired by the incapability of the standard metric (FID) to measure this overfitting, we proposed 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, with less computation time than the original TSTR classification accuracy.

To robustly assess the quality of the generated samples and their impact on the classification task, we designed a light curve classifier based on ensembles of neural networks that can provide statistical consistency when being trained multiple times.

In order to compare our model with other classic data augmentation techniques for time-series such as window-warping, we proposed a modification of this technique that is more suitable for our scenario (soft window-warping).

Using the proposed methodology, we can generate diverse synthetic datasets of irregularly sampled time-series that capture the properties of the original training sets and leverage their diversity to outperform classifiers trained uniquely on real data. Our experiments explored training the classifiers with only generated synthetic datasets, only real datasets, and a combination of both datasets to use our generative models as data augmentation methods.

The proposed model could be extended to work with classifiers that are currently operating in real-time such as the ALeRCE light curve classifier [25], boosting its performance on the ZTF stream and eventually on the future LSST, contributing to understanding the tridimensional structure and formation of our galaxy and its neighbors.

As an additional note, the work presented in this thesis has been summarized in a paper [76] that was accepted and published in The Astrophysical Journal (ApJ).

5.1 Future Work

Although effective in this simplified setup, the proposed methodology could be improved in many aspects. For example, the concatenation operation used to add the conditional parameters to D is sub-optimal since it repeats unnecessary information, as detailed in Section 3.3.2. It may also not scale well if there were more conditional parameters of interest. A more thoughtful way of doing this would be using embedding layers to project these parameters into an intermediate representation, which could be still added to the input of the discriminator in a C-GAN-like approach or at later stages in a Projection Discriminator-like approach [41].

Another exciting improvement to the model could be 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 [52] or [53].

Regarding the values used for conditional generation, we used the class-conditional parameter to generate datasets 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 conditional distribution of these parameters, and other physical parameters that may be relevant to include.

Similarly, related to sampling methods for generation, all our synthetic datasets were generated by sampling the latent vector z from a multivariate Gaussian distribution. Evaluating different sampling methods, such as those presented in [65], and inspecting how they affect the qualitative and quantitative results, could be an exciting path to follow.

From an astronomical perspective, we can mention two interesting improvements. First, extending this generation framework to non-periodic sources would be an exciting next step that should be accomplished almost effortlessly. Second, modifying the model to deal with multi-band light curves could be tackled by adding the desired band to the conditional parameters of interest, and it would be essential in applying this framework to data from the new-generation telescopes.

Finally, a piece of advice that we would like to transmit to anyone interested in using generative models for a downstream task is to learn as much as possible about the downstream task itself. Specifically, if the desired downstream task is classification, and the classifiers that want to be improved are well-known feature-based algorithms, attempting to generate time-series, is probably not the best idea. Instead, the generative models should intend to be as specific to the downstream task as possible, which for this case would mean learning the distribution of the features that will be used then for the classification problem instead of tackling it indirectly by generating time-series.

Bibliography

[1] R. Caruana, "Learning from imbalanced data: Rank metrics and extra tasks," *Proc. Am. Assoc. for Artificial Intelligence (AAAI) Conf*, pp. 51–57, 2000.

[2] H. He and E. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, pp. 1263–1284, 9 2009.

[3] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao *et al.*, "Time series data augmentation for deep learning: A survey," *Proceedings of the 30th Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4653–4660, 8 2021.

[4] B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," *PLOS One*, vol. 16, pp. 1–32, 7 2021.

[5] V. Sampath, I. Maurtua, J. J. A. Martín, and A. Gutierrez, "A survey on generative adversarial networks for imbalance problems in computer vision tasks," *Journal of Big Data*, vol. 8, pp. 1–59, 2021.

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley *et al.*, "Generative adversarial nets," *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, pp. 2672–2680, 2014.

[7] X. Zhu, Y. Liu, J. Li, T. Wan, and Z. Qin, "Emotion classification with data augmentation using generative adversarial networks," *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 349–360, 2018.

[8] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger *et al.*, "GAN-based synthetic medical image augmentation for increased cnn performance in liver lesion classification," *Neurocomputing*, vol. 321, pp. 321–331, 2018.

[9] H. Salehinejad, S. Valaee, T. Dowdell, E. Colak, and J. Barfett, "Generalization of deep neural networks for chest pathology classification in x-rays using generative adversarial networks," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 990–994, 2018.

[10] Y. Huang, Y. Jin, Y. Li, and Z. Lin, "Towards imbalanced image classification: A generative adversarial network ensemble learning method," *IEEE Access*, vol. 8, pp. 88 399–88 409, 2020.

[11] G. Ramponi, P. Protopapas, M. Brambilla, and R. Janssen, "T-CGAN: Conditional generative adversarial network for data augmentation in noisy time series with irregular sampling," *arXiv e-prints*, p. arXiv:1811.08295, 2018.

[12] Z. Zhang, J. Han, K. Qian, C. Janott, Y. Guo *et al.*, "Snore-GANs: Improving automatic snore sound classification with synthesized data," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, pp. 300–310, 1 2020.

[13] J. P. Gardner, J. C. Mather, M. Clampin, R. Doyon, M. A. Greenhouse *et al.*, "The James Webb Space Telescope," *Space Science Reviews*, vol. 123, pp. 485–606, 4 2006.

[14] J. Martínez-Palomera, F. Förster, P. Protopapas, J. C. Maureira, P. Lira *et al.*, "The high cadence transit survey (hits): Compilation and characterization of light-curve catalogs," *The Astronomical Journal*, vol. 156, p. 186, 10 2018.

[15] A. J. Drake, M. J. Graham, S. G. Djorgovski, M. Catelan, A. A. Mahabal *et al.*, "The catalina surveys periodic variable star catalog," *The Astrophysical Journal Supplement Series*, vol. 213, 6 2014.

[16] C. Alcock, R. A. Allsman, D. R. Alves, T. S. Axelrod, A. C. Becker *et al.*, "The macho project: Microlensing detection efficiency," *The Astrophysical Journal Supplement Series*, vol. 136, pp. 439–462, 10 2001.

[17] G. Pojmański, "The all sky automated survey," *Astronomische Nachrichten*, vol. 325, pp. 553–555, 10 2004.

[18] J. L. Tonry, L. Denneau, A. N. Heinze, B. Stalder, K. W. Smith *et al.*, "Atlas: A high-cadence all-sky survey system," *Publications of the Astronomical Society of the Pacific*, vol. 130, p. 064505, 6 2018.

[19] E. C. Bellm, S. R. Kulkarni, M. J. Graham, R. Dekany, R. M. Smith *et al.*, "The Zwicky Transient Facility: System overview, performance, and first results," *Publications of the Astronomical Society of the Pacific*, vol. 131, p. 18002, 12 2018.

[20] Željko Ivezić, S. M. Kahn, J. A. Tyson, B. Abel, E. Acosta *et al.*, "LSST: From science drivers to reference design and anticipated data products," *The Astrophysical Journal*, vol. 873, p. 111, 3 2019.

[21] F. Förster, G. Cabrera-Vives, E. Castillo-Navarrete, P. A. Estévez, P. Sánchez-Sáez *et al.*, "The automatic learning for the rapid classification of events (alerce) alert broker," *The Astronomical Journal*, vol. 161, p. 242, 4 2021.

[22] G. Narayan, T. Zaidi, M. D. Soraisam, Z. Wang, M. Lochner *et al.*, "Machine-learning-based brokers for real-time classification of the LSST alert stream," *The Astrophysical Journal Supplement Series*, vol. 236, p. 9, 5 2018.

[23] K. W. Smith, R. D. Williams, D. R. Young, A. Ibsen, S. J. Smartt *et al.*, "Lasair: The transient alert broker for LSST:uk," *Research Notes of the AAS*, vol. 3, p. 26, 1 2019.

[24] R. Carrasco-Davis, "Image sequence simulation and deep learning for astronomical object classification," *Master's Thesis, University of Chile*, 2019.

[25] P. Sánchez-Sáez, I. Reyes, C. Valenzuela, F. Förster, S. Eyheramendy *et al.*, "Alert classification for the alerce broker system: The light curve classifier," *The Astronomical Journal*, vol. 161, p. 141, 2 2021.

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[27] D. H. Hubel, "The visual cortex of the brain," *Scientific American*, vol. 209, pp. 54–63, 11 1963.

[28] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional arquitecture in two nonstriate visual areas (18 and 19) of the cat," *Journal of Neurophysiology*, vol. 28, pp. 229–289, 3 1965.

[29] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," *Lecture Notes in Computer Science*, vol. 1681, pp. 319–345, 1999.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[31] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *rXiv e-prints*, p. arXiv:1603.07285, 3 2016.

[32] B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle, "Neural autoregressive distribution estimation," *Journal of Machine Learning Research*, vol. 17, pp. 7184–7220, 1 2016.

[33] A. V. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48, pp. 1747–1756, 4 2016.

[34] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves *et al.*, "Conditional image generation with pixelcnn decoders," *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 4797–4805, 2016.

[35] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *2nd International Conference on Learning Representations (ICLR), Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[36] R. Salakhutdinov and H. Larochelle, "Efficient learning of deep boltzmann machines," *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9, pp. 693–700, 4 2010.

[37] Y. Bengio, Éric Thibodeau-Laufer, G. Alain, and J. Yosinski, "Deep generative stochastic networks trainable by backprop," *Proceedings of the 31st International Conference on*

International Conference on Machine Learning - Volume 32, p. II–226–II–234, 2014.

[38] L. Theis, A. van den Oord, and M. Bethge, "A note on the evaluation of generative models," *International Conference on Learning Representations (ICLR)*, 2016.

[39] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv e-prints*, p. arXiv:1411.1784, 2014.

[40] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, pp. 2642–2651, 3 2017.

[41] T. Miyato and M. Koyama, "cGANs with projection discriminator," *International Conference on Learning Representations (ICLR)*, p. arXiv:1802.05637, 2018.

[42] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, pp. 214–223, 3 2017.

[43] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein GANs," *Advances in Neural Information Processing Systems*, vol. 30, pp. 5769–5779, 2017.

[44] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *International Conference on Learning Representations (ICLR)*, 2018.

[45] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," *International Conference on Learning Representations (ICLR)*, 2018.

[46] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," *International Conference on Learning Representations (ICLR)*, 2019.

[47] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2019.

[48] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen *et al.*, "Analyzing and improving the image quality of styleGAN," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2020.

[49] Z. Wang, Q. She, and T. E. Ward, "Generative adversarial networks in computer vision: A survey and taxonomy," *ACM Computing Surveys*, vol. 54, pp. 1–38, 3 2022.

[50] O. Mogren, "C-RNN-GAN: A continuous recurrent neural network with adversarial training," *Constructive Machine Learning Workshop (CML) at NIPS*, 2016.

[51] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional GANs," *arXiv e-prints*, p. arXiv:1706.02633, 2017.

[52] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[53] H. Ni, L. Szpruch, M. Wiese, S. Liao, and B. Xiao, "Conditional sig-wasserstein GANs for time series generation," *SSRN Electronic Journal*, 2020.

[54] C. Villani, *Optimal Transport.* Springer Berlin Heidelberg, 2009, vol. 338.

[55] Y. Yazici, C.-S. Foo, S. Winkler, K.-H. Yap, G. Piliouras *et al.*, "The unusual effectiveness of averaging in GAN training," *International Conference on Learning Representations (ICLR)*, vol. abs/1806.04498, 2019.

[56] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen *et al.*, "Training generative adversarial networks with limited data," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 104–12 114, 2020.

[57] M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo, "Reliable fidelity and diversity metrics for generative models," *Proceedings of the 37th International Conference on Machine Learning*, vol. 119, pp. 7176–7185, 3 2020.

[58] J. Yang, A. Kannan, D. Batra, and D. Parikh, "Lr-gan: Layered recursive generative adversarial networks for image generation," *International Conference on Learning Representations (ICLR)*, 2017.

[59] S. Santurkar, L. Schmidt, and A. Madry, "A classification-based study of covariate shift in GAN distributions," *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, pp. 4480–4489, 3 2018.

[60] K. Shmelkov, C. Schmid, and K. Alahari, "How good is my GAN?" *Proceedings of the European Conference on Computer Vision (ECCV)*, 9 2018.

[61] S. Ravuri and O. Vinyals, "Classification accuracy score for conditional generative models," *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 12 268–12 279, 2019.

[62] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford *et al.*, "Improved techniques for training GANs," *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 2234–2242, 2016.

[63] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6629–6640, 2017.

[64] M. S. M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly, "Assessing generative models via precision and recall," *Proceedings of the 32nd International Conference on*

Neural Information Processing Systems, pp. 5234–5243, 2018.

[65] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila, "Improved precision and recall metric for assessing generative models," *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019.

[66] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 6 2016.

[67] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying MMD GANs," *International Conference on Learning Representations (ICLR)*, 2018.

[68] P. Mertikopoulos, C. Papadimitriou, and G. Piliouras, "Cycles in adversarial regularized learning," *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2703–2717, 1 2018.

[69] C. Papadimitriou and G. Piliouras, "From nash equilibria to chain recurrent sets: An algorithmic solution concept for game theory," *Entropy*, vol. 20, p. 782, 10 2018.

[70] J. P. Bailey and G. Piliouras, "Multiplicative weights update in zero-sum games," *Proceedings of the 2018 ACM Conference on Economics and Computation*, pp. 321–338, 6 2018.

[71] A. L. Guennec, S. Malinowski, and R. Tavenard, "Data augmentation for time series classification using convolutional neural networks," *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, 9 2016.

[72] O. Pimentel, P. A. Estévez, and F. Förster, "Deep attention-based supernovae classification of multi-band light-curves," *arXiv e-prints*, p. arXiv:2201.08482, 2022.

[73] R. Carrasco-Davis, E. Reyes, C. Valenzuela, F. Förster, P. A. Estévez *et al.*, "Alert classification for the alerce broker system: The real-time stamp classifier," *The Astronomical Journal*, vol. 162, p. 231, 12 2021.

[74] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations (ICLR), San Diego, CA, USA, Conference Track Proceedings*, 2015.

[75] J. Opitz and S. Burst, "Macro f1 and macro f1," *arXiv e-prints*, p. arXiv:1911.03347, 2019.

[76] G. García-Jara, P. Protopapas, and P. A. Estévez, "Improving astronomical time-series classification via data augmentation with generative adversarial networks," *The Astrophysical Journal*, vol. 935, p. 23, 8 2022.