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

CLASIFICACIÓN PROFUNDA DE CURVAS DE LUZ MULTI-BANDA BASADA EN ATENCIÓN

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

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

ÓSCAR ALBERTO PIMENTEL FUENTES

PROFESOR GUÍA:
PABLO ESTÉVEZ VALENCIA

PROFESOR CO-GUÍA:
FRANCISCO FÖRSTER BURÓN

MIEMBROS DE LA COMISIÓN:
PABLO HUIJSE HEISE
CLAUDIO PEREZ FLORES

Este trabajo ha sido parcialmente financiado por el proyecto Millennium Institute of Astrophysics IC12009 y el proyecto FONDECYT regular #1220829

SANTIAGO DE CHILE
2022

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE MAGISTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA,
MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: ÓSCAR ALBERTO PIMENTEL FUENTES
FECHA: 2022
PROF. GUÍA: PABLO ESTÉVEZ VALENCIA

CLASIFICACIÓN PROFUNDA DE CURVAS DE LUZ MULTI-BANDA BASADA EN ATENCIÓN

En estudios astronómicos, como Zwicky Transient Facility (ZTF), las supernovas (SNe) son objetos poco comunes en comparación con otras clases de eventos astronómicos. Junto con esta escasez de datos, el procesamiento de curvas de luz multi-banda es una tarea desafiante debido a la cadencia altamente irregular, largos intervalos de tiempo, mediciones perdidas, baja cantidad de observaciones, etc. Estos problemas son particularmente perjudiciales para el análisis de eventos transitorios usando curvas de luz, especialmente SNe. En este trabajo, se ofrecen tres contribuciones principales. 1) Basándose en una modulación temporal y en mecanismos de atención, se propone un modelo llamado TimeModAttn que busca clasificar curvas de luz multi-banda de diferentes tipos de SNe. 2) Se propone un modelo para la generación sintética de curvas de luz multi-banda de SNe basado en un Modelo Paramétrico de SNe (SPM). Esto busca poder aumentar el número de muestras y diversidad de cadencia. El modelo TimeModAttn propuesto superó a un clasificador Random Forest, incrementando el balanced-F_1score desde $\approx .525$ hasta $\approx .596$. TimeModAttn también superó a otros modelos de aprendizaje profundo, basados en Redes Neuronales Recurrentes (RNNs), en dos escenarios: clasificación tardía y clasificación temprana. 3) Se realizaron experimentos de interpretabilidad buscando validar el funcionamiento del modelo TimeModAttn.

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE MAGISTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA,
MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: ÓSCAR ALBERTO PIMENTEL FUENTES
FECHA: 2022
PROF. GUÍA: PABLO ESTÉVEZ VALENCIA

DEEP ATTENTION-BASED SUPERNOVAE CLASSIFICATION OF MULTI-BAND LIGHT-CURVES

In astronomical surveys, such as the Zwicky Transient Facility (ZTF), supernovae (SNe) are relatively uncommon objects compared to other classes of variable events. Along with this scarcity, the processing of multi-band light-curves is a challenging task due to the highly irregular cadence, long time gaps, missing-values, low number of observations, etc. These issues are particularly detrimental for the analysis of transient events with SN-like light-curves. In this work, we offer three main contributions. 1) Based on temporal modulation and attention mechanisms, we propose a Deep Attention model called TimeModAttn to classify multi-band light-curves of different SN types, avoiding photometric or hand-crafted feature computations, missing-values assumptions, and explicit imputation and interpolation methods. 2) We propose a model for the synthetic generation of SN multi-band light-curves based on the Supernova Parametric Model (SPM). This allows us to increase the number of samples and the diversity of the cadence. The proposed TimeModAttn model outperformed a Random Forest classifier, increasing the balanced-F_1score from $\approx .525$ to $\approx .596$. TimeModAttn also outperformed other Deep Learning models, based on Recurrent Neural Networks (RNNs), in two scenarios: late-classification and early-classification. 3) We conducted interpretability experiments to evaluate and validate proposed model.

«To learn something new, you need to try new things and not be afraid to be wrong.»
Roy T. Bennett.

Acknowledgements

Many things were learned during this far too long process, but I think that my most personal and important lesson is the following: how critical is to have meaningful human connections, to surround yourself with people who care about you beyond your academic grade, your work, or even your achievements. Unfortunately, this lesson was learned the hard way and accompanied by great suffering.

It is not my intention to extend the acknowledgements too long, so I can only present here a brief dedication, thanking some of the great people who accompanied me during this insane challenge which probably was the hardest of my life (I lost count of the number of times I threw up because of the uncontrollable anxiety this caused me). Please be aware that I keep with me any help or emotional support you may have ever offered me during this tough process. Frankly, without you, this would not have been possible. I appreciate it with all my heart.

I'm extremely grateful to the group of people who accompanied me during this thesis process, who have been remarkably generous with their time, technical and emotional support:

Thanks to my beloved family, for always being there for me throughout my life: Francisco (my dad), Sandra (my mom), Francisca (my sister), and Mila (my dog). Without you, this would have been impossible.

Thanks to my thesis committee. Thanks to my professors Pablo Estevez and Francisco Förster, for always being attentive and willing offer me they exceptional help, guidance, and wisdom.

Thanks to my ever-there schoolmates and life-friends: Lolito, Rocuz, Cristobal, Jorquera, Dieguito, Fatias, Marito, and Nico. Also, thanks to Fabian (Dr. Fausto) for the all the good food and book recommendations.

Thanks to all those wonderful people I was fortunate enough to meet during my college years: Alfredo, Cata, Cristobal, Dieguito, Fefi, Mauro, Vicho, Javier, Nico, and Vichotote. I would like to personally thank Alonso, for helping me so many times in my worst moments.

Thanks to my always irreverent and sarcastic friends from the electrical engineering department, for all those bad laughs and monkey memes: Renete, Claudaña, Constantino Fernandez, Nico (Medivh), and Saldaña (Cromi). Also, thanks to Carolina and Valentina for understanding my suffering.

Thanks to my colleagues in the computational intelligence laboratory (LabIC), for always willing to help me and offer me feedback on my work: Nicolás Tapia, Germán García, Esteban Reyes, Mauricio Romero, Nicolás Astorga, Ignacio Reyes, Rodrigo Carrasco, Pablo Montero, and Rosario Molina. Thanks to Loreto for always solving my doubts about the MsC program. I also thank Hugo and Luis for helping me to reboot my PC so many times.

Thanks to the Corgos team and the Survivors team. Last but not least, I thank to the always reliable Stitches for all those good hooks.

Additional Acknowledgements

The authors acknowledge support from the National Agency of Research and Development's Millennium Science Initiative through grant IC12009, awarded to the Millennium Institute of Astrophysics (OP, PE, FF) and from the National Agency for Research and Development (ANID) grants: BASAL Center of Mathematical Modelling AFP-170001, ACE210010, FB210005 (FF), and FONDECYT Regular #1200710 (FF) and #1220829 (PE). We thank the ALeRCE broker for collecting and providing the data used for this thesis. We thank Pablo Montero and Nicolás Astorga for the useful discussions.

Table of Content

Table of Tables

Table of Figures

Chapter 1

Introduction

1.1. Motivation

The supernovae are transient astronomical events resulting from a powerful and bright stellar explosion, which have captured the curiosity of astronomers since ancient times. In the past, it was believed that a supernova was a very bright new star in the firmament, which is why it was coined with the name «super-novae» or «new» from Latin. However, it is now known that a supernova is an explosion that marks the end of the life cycle of a star. The brightness caused by this explosion can be so extreme that it can even be compared to the brightness emitted by the galaxy that hosts the supernova event (Fig. 1.1 shows an example of a supernova along with its host galaxy).

The study of transient astronomical events, specifically supernovae (SNe), has played a critical role in astronomy. Type Ia SNe (thermonuclear SNe) are standardizable candles and have become important tools for cosmological distance determinations (Wright & Li, 2018), leading to the discovery of the accelerated expansion of the universe (Schmidt et al., 1998; Riess et al., 1998) and its precise characterization with projects such as the Dark Energy Survey (DES; Sánchez, 2006; Dark Energy Survey Collaboration et al., 2016). At the same time, the study of Type Ib/c and Type II supernovae (core-collapse SNe) has helped astronomers to understand the evolution and explosion mechanisms of stars, including insights into the formation of stellar mass black holes (Sukhbold & Adams, 2020).

These expanded opportunities to study the cosmos are a consequence of the constant efforts to develop new telescopes that collect massive amounts of data every night, creating a new Big Data paradigm for astronomy. High-volume data collection is managed by astronomical surveys such as the Zwicky Transient Facility survey (ZTF; Bellm et al., 2019) and experiments such as the High Cadence Transient Survey (HiTS; Förster et al., 2016). These surveys are preparing us for the Vera C. Rubin Observatory and its Legacy Survey of Space and Time (LSST; Ivezić et al., 2019). The LSST survey is expected to gather approximately 15 terabytes of raw data per night by observing up to 37 billion astronomical objects in 10 years, including several millions of SNe (Ivezić et al., 2019).

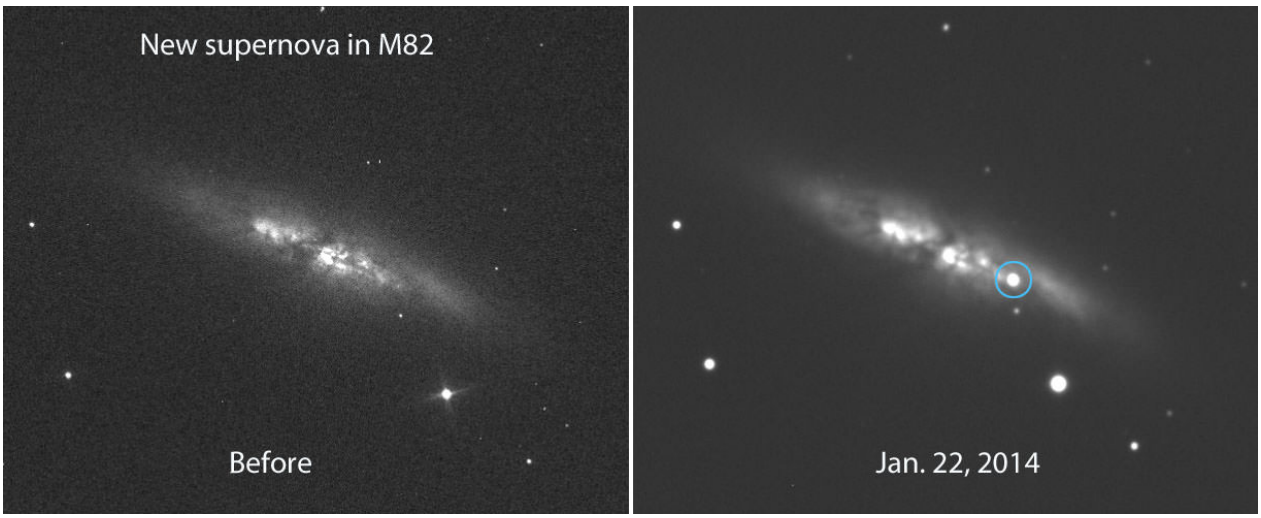

Figure 1.1: Example of the appearance of a SN in the host galaxy M82 (marked with a blue circle). Original figure extracted from `https://www.universetoday.com/108386/bright-new-supernova-blows-up-in-nearby-m82-the-cigar-galaxy/`.

1.1.1. Previous Works

Historically, SNe have been studied and classified into different types through optical spectroscopy. However, the use of this technique requires an immense investment of time and human effort. Given this limitation, only a marginal proportion of the SN candidates, reported from high-volume data streams, are being effectively studied and followed-up.

The Big Data paradigm challenge has motivated the scientific community to search for alternative methods for classification other than spectroscopic observations. In particular, several methods have been proposed to classify different types of SNe using the discovery images and light-curves. Most existing methods are based on features extracted from the light-curves by using parametric models (Karpenka et al., 2012; Noebauer et al., 2017; Lochner et al., 2016; Villar et al., 2019), PCA and Kernel PCA reductions (Ishida & de Souza, 2013; Lochner et al., 2016), Wavelet based features (Varughese et al., 2015; Lochner et al., 2016), Gaussian processes light-curve augmentation (Boone, 2019), and different hand-crafted features (Villar et al., 2019).

A successful example for the processing of discovery images and light-curves is the Automatic Learning for the Rapid Classification of Events broker (ALeRCE; Förster et al., 2021; Sánchez-Sáez et al., 2021; Carrasco-Davis et al., 2021). The ALeRCE broker considered a vast collection of features[1] based on prior astrophysical expert knowledge (Sánchez-Sáez et al., 2021), including a SN parametric model. These extracted features are used to classify SNe (or other astronomical events) along with classical Machine Learning models, such as the Balanced Random Forest (BRF), Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), and Gradient Boosting.

A notable difficulty is the intrinsic scarcity in the number of empirical SN light-curves, especially for certain SN types such as the Superluminous SNe (SLSN), which also leads to

[1] `http://alerce.science/features/`.

a high class imbalance within the SN classes. These difficulties have motivated the release of several simulated SN light-curve datasets as part of data classification challenges, such as the Supernova Photometric Classification Challenge (SPCC; Kessler et al., 2010) and the Photometric LSST Astronomical Time-Series Classification Challenge (PLAsTiCC; The PLAsTiCC team et al., 2018).

Several Deep Learning models have been motivated by these challenges. In Charnock & Moss (2017), SN light-curve classifiers based on Recurrent Neural Networks (RNNs) were proposed using models such as the Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM). To deal with the multi-band missing-values, a light-curve imputation was performed using random values between the last and next valid light-curve observations. In Moss (2018), a Phased-LSTM model was used to include the time information as a new memory gate, computing averages between the last and next observations to deal with missing-values.

The PELICAN project (Pasquet et al., 2019) proposed an autoencoder (encoder-decoder) architecture based on the use of Convolutional Neural Networks (CNNs) that are adapted to process time series, such as SN multi-band light-curves. This work dealt with the irregular cadence by using a missing-value assumption, where a set of additional loss functions were proposed to attenuate the overfitting risk associated with zero mask values. In Brunel et al. (2019), an adapted CNN based model was also proposed for the processing of SN light-curves along with a promising Siamese network architecture.

The RAPID project (Muthukrishna et al., 2019) used a GRU model to classify different transient and SN types, including a new pseudo-class to characterize the SN pre-explosion region. The irregular cadence and multi-band misalignments were treated using a grid linear interpolation. In Möller & de Boissière (2020) a Bayesian RNN model was developed, where the time difference information between the current and last observation was included to describe the irregular cadence information as model input.

In addition, notable efforts have been made for the classification of other astronomical light-curves, such as variable stars and stochastic events. Deep Learning encoder and autoencoder models (encoder-decoder), based on RNN models (Naul et al., 2018; Jamal & Bloom, 2020; Tachibana et al., 2020; Donoso-Oliva et al., 2021) and Temporal CNN (TCNN) models (Jamal & Bloom, 2020; Zhang & Bloom, 2021), have been proposed for the automatic feature extraction from light-curves. Moreover, the direct processing of image-stamp sequences has been also proposed using Recurrent CNNs (RCNNs) (Carrasco-Davis et al., 2019; Gómez et al., 2020).

As a competitive alternative to RNNs, CNNs, and TCNNs, light-curve classification models based on attention mechanisms have started to emerge. In Ibsen & Mann (2020), a GRU model was jointly used with a self-attention mechanism to improve the early-classification performance. Recently, in Allam & McEwen (2021), a model to classify light-curves using an adapted Transformer model was developed, where a Gaussian process interpolation method was used to deal with the irregular cadence. However, both works used simulated light-curves from the PLAsTiCC dataset in a completely supervised learning scheme and heavily relying on light-curve interpolation methods.

Additional details about several of the aforementioned works, along with other not men-

tioned works, can be found in Appendix §A.1.

1.1.2. General Problem and Associated Difficulties

A brief summary of some of the difficulties reported in the literature w.r.t. processing SN multi-band light-curve datasets is presented below:

- **Scarcity of data**: unlike periodic events or variable stars, the transient SN event has a limited time of existence in the cosmos. This, along with the intrinsic rarity and unpredictability of this event, ultimately produces a general scarcity of SN light-curves that are captured by the astronomical surveys. In general, it has been already reported that the associated number of SN light-curves in these datasets are not properly suitable for the Deep Learning standards (Charnock & Moss, 2017).
- **Imbalance of SN types**: there are SN types whose probability of occurrence in the cosmos is higher than others. In particular, it is common to find, in astronomical surveys, a high number of light-curves associated with the Ia type, but a low number of light-curves of the core-collapse SN type. This SN type imbalance can be highly detrimental for some automatic classification algorithms as it usually induces a bias on the types with the highest number of samples: the model can ensure a good performance by correctly characterizing only the most abundant or majority class (Hosenie et al., 2020).

Fig. 1.2 shows an example of a SN multi-band light-curve. The study and processing of SN multi-band light-curves is also confronted with important challenges, such as the following ones:

- **Irregular cadence**: in the terrestrial survey context, there is an intrinsic irregularity in the continuous observation of the cosmos (photometric follow-up) due to factors such as unfavorable atmospheric conditions for the measurements, general unavailability of the telescope and the used optical bands, rotation of the earth, position of the sun and the moon, etc. Notice that these factors have an external nature to the astronomical object to be followed-up; therefore, the irregular sampling (cadence) does not show an especial correlation w.r.t. the original behavior of the astronomical event. In turn, this irregular cadence behavior also produces an extra set of especial challenges and difficulties:
  - **Time gaps without observations**: due to the irregular cadence, it is common to find light-curves that present long periods of time without any observations, which may even exceed 100 days in extreme cases.
  - **Missing observations between bands**: it is not always possible to have simultaneous observations for all the optical filters in the telescope. Therefore, observations associated with one or multiple bands may be lost (missing-values problem)[2].

[2]This phenomena can also be considered as a temporal multi-band misalignment problem.

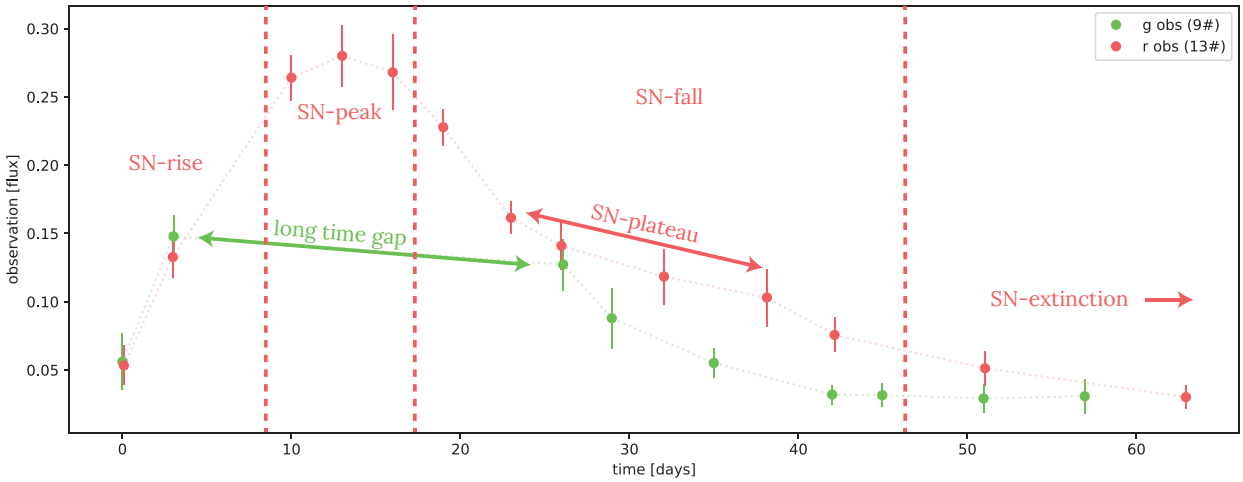

Figure 1.2: Example of a SN multi-band light-curve. Challenges, such as the irregular cadence, variable-length, multi-band missing-values, and long time gaps, can be observed in this light-curve. Relevant SN regions are also noted: SN-rise, SN-peak, SN-fall, SN-plateau, and SN-extinction. It is worth mentioning that, in general, not all the SN light-curves are so well-behaved, especially in case of empirical surveys such as the ZTF survey.

  - **Variable-length**: in general, the empirical available light-curves have an arbitrary number of observations for each of the optical photometric bands, i.e., they are variable-length multi-variable time series. In some extreme cases, by considering the transient behavior of the SN event, this problem may even produce light-curves with few or even zero number of observations in one photometric band.
  - **Irregular time offset**: due to the irregular cadence, it is possible to start capturing the photometric information of SNe after its brightness peak has already occurred (time offset). Having no information from the SN-peak, or the very first days of the SN, can be highly detrimental for a correct characterization of the SN light-curve.

- **Disturbances by external factors**: the behavior of the apparent maximum brightness of a light-curve can be affected by external factors, e.g., weather conditions, sky brightness, distance of the SN captured w.r.t. the observer (planet Earth), etc. This difficulty can be worked out, in some degree, along with variables obtained from spectrometry techniques, such as the redshift. Unfortunately, consistently having access to these type of spectral variables is highly unlikely in the new paradigm of Big Data in astronomy.

- **Intrinsic misclassifications between SN types**: the classification between types of SN can be a challenging task due to some intrinsic similarities in the nature of these events. For instance, as previously reported in the literature (Moss, 2018; Villar et al., 2019; Sánchez-Sáez et al., 2021), a common confusion between the Ia and Ibc SN types can be usually found among Machine Learning classification algorithms. This confusion may be related with intrinsic similarities of the mechanisms that explain the SN-peak: the diffusion of energy deposited by radioactive ^{56}Ni (Arnett, 2008).

- **Heteroscedastic noise**: the uncertainties associated with brightness observations on

a light-curve are heteroscedastic, i.e., the dispersion behavior does not follow a fixed trend. Moreover, this behavior can be different for each photometric band, presenting bands with a higher associated intrinsic noise than others.

By taking into account the intrinsic characteristics associated with the handling of empirical SN multi-band light-curves, the methodology of this work is developed in chapter §3. We propose methods and strategies that aim to correctly deal with the problems and difficulties aforementioned, especially with important challenges such as the general scarcity of SN light-curves from non-simulated SN datasets, the high class imbalance, and the high sampling irregularity (irregular cadence) presented in the SN multi-band light-curves.

1.2. Hypotheses

1. Models based on temporal modulation and attention mechanisms are able to automatically extract meaningful features and temporal dependencies from the SN multi-band light-curves. These features are sufficiently expressive to allow the model to solve two main tasks: 1) A SN multi-band light-curve reconstruction task. 2) A categorical discrimination task between different types of SNe.
2. We hypothesize that models based on temporal modulation and attention mechanisms can solve the proposed classification task equal to or better than two other alternatives previously implemented in the literature: 1) Models based on astrophysical and hand-crafted feature engineering. 2) Models based on other type of Deep Learning methods such as RNNs.
3. The use of synthetic light-curves allows the models to explore information that may not be correctly captured given the highly irregular sampling (irregular cadence) that arises when working with empirical SN light-curves. Then, the use of synthetic SN light-curves results in an equal or superior classification performance than the use of only empirical SN light-curves for both: 1) Models based on astrophysical and hand-crafted feature engineering. 2) Models based on other type of Deep Learning methods, e.g., RNNs.

1.3. General Objective

To propose and implement a model based on time modulation and attention mechanisms for the classification of SN multi-band light-curves from the ZTF survey, and to develop a method for the generation of synthetic SN multi-band light-curves by using a parametric model of SNe.

1.4. Specific Objectives

1. To propose and implement a Deep Learning classifier, based on temporal modulation and attention mechanisms, to discriminate among different SN types by automatically

extracting meaningful features from the SN multi-band light-curves. To implement Deep Learning training strategies that allow us to deal with the high class imbalance and data scarcity: imbalance learning and data-augmentation techniques.

2. To compare the performance of the proposed Deep Learning model w.r.t. classical methodologies based on the engineering of astrophysical and hand-crafted features extracted from the SN multi-band light-curves. To compare the performance of the proposed model w.r.t. other Deep Learning alternatives used in the literature (e.g., RNNs). These comparisons are studied in two scenarios: the late-classification and the early-classification scenarios.
3. To implement interpretability experiments, supported by the interpretative intrinsic capacity of the temporal modulation and attention mechanisms, that allow us to inspect, evaluate, and validate the automatic decisions exposed by the proposed model when processing the SN multi-band light-curves.
4. To develop a method for the synthetic generation of SN multi-band light-curves based on a parametric model for SNe. To design experiments to evaluate the quality of the generated synthetic SN light-curves and to compare them with other synthetic generation baseline alternatives.

1.5. Main Contributions

In this work, we propose an attention-based model for the classification of different types of SN by using empirical multi-band light-curves from the ZTF survey. Our main contributions are the following: 1) We propose a Deep Attention model (TimeModAttn), based on temporal modulation (TimeFiLM) and attention mechanisms (MHSelfAttn), to process and classify SN multi-band light-curves. The proposed model avoids the computation of any time-consuming photometric or hand-crafted features, as well as the use of missing-value assumptions and explicit light-curve imputation or interpolation methods. From our experiments, we found that the TimeModAttn model achieved higher performance than other classical baselines: a feature-based model (BRF) and RNN-based models (GRU, LSTM). 2) To support the optimization of the tested Deep Learning models, we propose a new method to generate synthetic SN multi-band light-curves as an effort to increase both, the total number of samples and the diversity of the irregular cadence population from the original dataset. 3) We conduct several interpretability experiments for SN multi-band light-curves in order to explore, evaluate, and validate the proposed model.

1.6. Organization of this Work

This thesis is structured as follows:

- **Introduction (chapter §1)**: the general problem and the main motivations of this work are introduced. Additionally, some of the most important previous works are described (literature review) along with a brief summary of the most important challenges

of this work. Finally, the hypotheses, objectives, and main contributions of this work are described.

- **Theoretical Background (chapter §2)**: the mathematical notation used in this work is introduced. In addition, a brief background about relevant theoretical concepts is presented. First, in section §2.2, methods associated with the area of Data Science and Machine Learning are described. Then, in section §2.3, general astronomy concepts are presented.
- **Methodology (chapter §3)**: in section §3.1, we introduce the light-curve dataset used in this work and describe the pre-processing procedures. In section §3.2, we describe our methodology for generating synthetic SN multi-band light-curves. In section §3.3, the classifier baseline used for comparison purposes is described, which is based on photometric features and the Balanced Random Forest (BRF) model. In section §3.4, we describe the complete methodology associated with our proposed model (TimeModAttn) and the optimization process for the classification of SN light-curves. We also describe other baselines based on RNN models.
- **Results and Analyses (chapter §4)**: the results from our experiments are reported using several metrics to compare the performance of the TimeModAttn model w.r.t. the tested baselines. In addition, we conduct several interpretability experiments based on the proposed model.
- **Conclusions (chapter §5)**: finally, we draw the conclusions and propose guidelines for future work in this research line.
- **Appendices**: this section describes extra details about the methodology and the work in general. In addition, additional experiments and results are shown and explored, e.g., ablation studies, different and alternative model configurations, etc.

Chapter 2

Theoretical Background

In this chapter, a set of key and relevant theoretical contents for this thesis are presented. This chapter is distributed into three main sections: 1) In section §2.1, the notations used in this work are defined. 2) In section §2.2, several concepts and methods associated with Data Science and Machine Learning are explored. 3) In section §2.3, general theoretical concepts about astronomy are presented.

2.1. Notation

2.1.1. Multi-Band Light-Curve

An arbitrary i-th multi-band light-curve Φ_i, from a light-curve dataset, is defined as follows:

$$\Phi_i \equiv \left\{ \left(\mu_{i,j}, \sigma_{i,j}, t_{i,j}, b_{i,j} \right) \right\}_{j=1}^{L_i} | t_{i,j'} > t_{i,j}, \forall j' > j, \tag{2.1}$$

where the light-curve Φ_i is defined as a sequence set[1] with an arbitrary (variable-length) number of L_i photometric multi-band observations. Each observation contains photometric information, such as the observation-flux $\mu_{i,j}$ (flux) and the observation-error $\sigma_{i,j}$ (flux error). Also, each observation is associated with an observation-time $t_{i,j}$ (days) and an observation-band indicator $b_{i,j} \in \{1, \ldots, B\}$, where B is the total number of photometric bands available on the survey dataset. Note that the subscript notation i, j denotes that the observation value is associated with the j-th observation in the i-th light curve Φ_i. The light-curve sequence object is defined to be causally sorted over time, i.e., the observation-time $t_{i,j}$ increases monotonically if the sequence step j also increases: $t_{i,j'} > t_{i,j}, \forall j' > j$.

In addition, given a target band b, a single-band operator $(\cdot)^{(b)}$, for an arbitrary i-th

[1] $\{\phi_{i,j}\}_{j=1}^{L_i} = \{\phi_{i,1}, \ldots, \phi_{i,L_i}\}$, where $\phi_{i,j}$ is an arbitrary object.

multi-band light-curve Φ_i, is defined as follows[2]:

$$\Phi_i^{(b)} \equiv (\Phi_i)^{(b)} \equiv \left(\left\{ \left(\mu_{i,j}, \sigma_{i,j}, t_{i,j}, b_{i,j} \right) \right\}_{j=1}^{L_i} \right)^{(b)}, \tag{2.2}$$

$$\equiv \left\{ \left(\mu_{i,j^{(b)}}, \sigma_{i,j^{(b)}}, t_{i,j^{(b)}}, b_{i,j^{(b)}} \right) \right\}_{j^{(b)}=1}^{L_i^{(b)}}, \tag{2.3}$$

where the resulting single-band light-curve $\Phi_i^{(b)}$ is defined as a sequence set collection of all the photometric observations from the multi-band light-curve Φ_i that are associated with the selected band b. In this case, the light-curve $\Phi_i^{(b)}$ has an arbitrary (variable-length) number of $L_i^{(b)} \leq L_i$ photometric observations[3]. For simplicity, if a sequence step has the form $j^{(b)}$, it is then related with the single-band light-curve $\Phi_i^{(b)}$. This single-band operator is used, in the following sections, to define operations and formulations exclusively over a target band b.

2.1.2. First and Last Sequence Steps

The sequence step $j = 1$ is associated with the very first observation from a multi-band light-curve Φ_i (at any band). The sequence step $j = L_i$ is associated with the very last observation from a multi-band light-curve Φ_i (at any band). As a simplified notation, we use 1 and -1 for the first and last sequence steps, respectively (e.g., $\mu_{i,1}$, $\mu_{i,-1}$).

The sequence step $j^{(b)} = 1$ is associated with the very first observation from a single-band light-curve $\Phi_i^{(b)}$ (first observation, occurring in band b, from the multi-band light-curve Φ_i). The simplified sequence step $j^{(b)} = L_i^{(b)}$ is used to denote the sequence step that is associated with the very last observation from a single-band light-curve $\Phi_i^{(b)}$ (last observation, occurring in band b, from the multi-band light-curve Φ_i). As a simplified notation, we use $1^{(b)}$ and $-1^{(b)}$ for the first and last sequence steps, respectively (e.g., $\mu_{i,1^{(b)}}$, $\mu_{i,-1^{(b)}}$).

2.1.3. Time Difference

Given a multi-band light-curve Φ_i, an arbitrary time difference is defined as follows:

$$\Delta t_{i,j} \equiv \begin{cases} 0, & \text{if } j = 1, \\ t_{i,j} - t_{i,j-1}, & \text{otherwise }, \end{cases} \tag{2.4}$$

where the time difference associated with the first observation is $\Delta t_{i,1} = 0$. Moreover, the time difference between the current observation (at the sequence step j) and the previous observation (at the sequence step $j-1$) is denoted as $t_{i,j} - t_{i,j-1}$.

In addition, given a single-band light-curve $\Phi_i^{(b)}$, an arbitrary time difference is defined as

[2]Note that the band indicator $b_{i,j^{(b)}}$ is redundant after applying the band operator.

[3]The total variable-length of the multi-band light-curve is the sum from all band observations as $L_i = \sum_{b=1}^{B} L_i^{(b)}$.

follows:

$$\Delta t^{(b)}_{i,j^{(b)}} \equiv \begin{cases} t_{i,j^{(b)}} - t_{i,1}, & \text{if } j^{(b)} = 1, \\ t_{i,j^{(b)}} - t_{i,j^{(b)}-1}, & \text{otherwise}, \end{cases} \tag{2.5}$$

where the time difference associated with the first observation is $t_{i,j^{(b)}} - t_{i,1}$. Therefore, the first time difference is $\Delta t^{(b)}_{i,1^{(b)}} = 0$ only if the first observation of the multi-band light-curve Φ_i occurs in the target band b. Moreover, the time difference between the current observation (at the sequence step $j^{(b)}$) and the previous observation (at the sequence step $j^{(b)} - 1$) is denoted as $t_{i,j^{(b)}} - t_{i,j^{(b)}-1}$.

2.2. Data Science

In this section we describe relevant models, algorithms, methods, and concepts about Data Science, Machine Learning, and Deep Learning. In addition, some basic concepts about Linear Algebra, Probability Theory, and Information Theory are also briefly described for the sake of completeness.

2.2.1. Linear Algebra

2.2.1.1. Vector Concatenation

Given a set of vectors $\left\{\boldsymbol{x}_i \in \mathbb{R}^{M_i}\right\}_{i=1}^{N}$ [4], the concatenation operation of these vectors is defined as follows:

$$\boldsymbol{x}' = \text{cat}[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N] = \left[\boldsymbol{x}_1^T, \ldots, \boldsymbol{x}_N^T\right]^T, \tag{2.6}$$

$$= [x_{1,1}, \ldots, x_{1,M_1}, \ldots, x_{N,1}, \ldots, x_{N,M_N}]^T, \tag{2.7}$$

where $\boldsymbol{x}' \in \mathbb{R}^{M_1+\cdots+M_N}$ is the new vector produced by the concatenation operation (cat[...]).

2.2.1.2. Linear Projection

A linear projection is defined as a linear transformation of a vector by using a matrix. Then, a linear projection $f_{\boldsymbol{A}}(\boldsymbol{x}) : \mathbb{R}^M \mapsto \mathbb{R}^N$ is defined as follows:

$$\boldsymbol{z} = f_{\boldsymbol{A}}(\boldsymbol{x}) = \boldsymbol{A}^T\boldsymbol{x}, \tag{2.8}$$

where $\boldsymbol{A} \in \mathbb{R}^{M\times N}$ is a matrix, $\boldsymbol{x} \in \mathbb{R}^M$ is the input vector, and $\boldsymbol{z} \in \mathbb{R}^N$ is the output vector that is projected into a new vectorial space.

[4] For simplicity, we denote the domain $\mathbb{R}^{M_i\times 1\times\cdots\times 1}$ just as the domain $\mathbb{R}^{M_i}$.

2.2.1.3. Eigenvectors and Eigenvalues

Given a square matrix $\boldsymbol{A} \in \mathbb{R}^{M \times M}$, an associated arbitrary eigenvector $\boldsymbol{v}_i \in \mathbb{R}^M$ and eigenvalue $\lambda_i \in \mathbb{R}$ must satisfy the following condition:

$$\boldsymbol{A}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i, \tag{2.9}$$

where this expression can be generalized and extended as follows:

$$(\boldsymbol{A} - \lambda \boldsymbol{I})\boldsymbol{v}_i = \vec{0}, \tag{2.10}$$

$$\det(\boldsymbol{A} - \lambda \boldsymbol{I}) = \vec{0}, \tag{2.11}$$

where, by solving eq. (2.11), the different eigenvectors and eigenvalues associated with the matrix $\boldsymbol{A}$ can be obtained. Note that the solution over the eigenvector $\boldsymbol{v}$ must be distinct from the zero vector $\vec{0}$.

Intuitively, in eq. (2.9), the vector $\boldsymbol{v}_i$ can be interpreted as a vector that is marking the direction where, after applying a linear transformation $\boldsymbol{A}$, all vectors aligned in that direction (the direction of the vector $\boldsymbol{v}_i$) do not change their direction in the projected hyperspace. While the direction is not affected, the scale is still affected. This produces that all vectors in the direction of $\boldsymbol{v}_i$ are scaled, after the linear transformation using $\boldsymbol{A}$, by a weighted factor λ_i.

2.2.2. Probability Theory

2.2.2.1. Random Variable and Probability Distribution

A discrete random variable is defined as a function and it is denoted as follows: $X : \Omega \mapsto \mathbb{R}^+$. This function allows us to map the outcome of a random experiment, from the domain set Ω, to a certain positive real numerical value. The set Ω is the sample space, i.e., the set of all possible experiment outcomes associated with the random experiment. It should be noted that a formal and extended definition can be described by using the Measure Theory and the σ-algebra.

In this probability framework, the expression $p_X(X = x)$ is defined as the probability that the random variable X (random experiment) takes the particular result of the experiment event x. For simplicity, the following notation can also be used: $p_X(X = x) = p_X(x) = p(x)$. Then, we call the function $p(\cdot)$ as the Probability Mass Function (PMF) which, in the discrete case, must satisfy the following conditions:

$$p(x) \leq 1, \forall x \in X, \tag{2.12}$$

$$\sum_{x \in X} p(x) = 1. \tag{2.13}$$

The above concept of random variable can also be extended to the continuous case. Here, we call the function $p(\cdot)$ as the Probability Density Function (PDF)[5], satisfying the following

[5]It is very common to refer interchangeably to both, the PMF and the PDF, as the «probability distribution».

condition:

$$\int p(x)dx = \int_{x \in X} p(x) = 1, \tag{2.14}$$

where, it should be noted, the discrete condition $p(x) \leq 1$ is not necessary in the continuous case as long as the density integrates a value equals to 1 over the entire domain of the random variable X.

2.2.2.2. Cumulative Distribution Function (CDF)

Given a random variable X, the cumulative probability function (CDF) is defined as follows:

$$F_X(x) \equiv p_X(X \leq x) = \sum_{x_i \leq x} p(x_i), \tag{2.15}$$

where $F_X : \mathbb{R} \mapsto [0, 1]$ is the CDF and p_X is the probability distribution of X. It should be noted that the CDF is a monotonically increasing function that holds the following condition: $\lim_{x \to -\infty} F_X(x) = 0 \land \lim_{x \to +\infty} F_X(x) = 1$.

Similarly, in the continuous case, the definition of the CDF is as follows:

$$F_X(x) \equiv p_X(X \leq x) = \int_{-\infty}^{x} p(x)dx. \tag{2.16}$$

2.2.2.3. Expectation Operator

Given a discrete random variable X, then the definition of the expectation operator is as follows:

$$\mathop{\mathbb{E}}_{x \sim p(x)} [f(x)] \equiv \mathop{\mathbb{E}}_{x} [f(x)] = \sum_{x \in X} f(x)p(x). \tag{2.17}$$

Moreover, this definition is also directly extensible to the continuous case:

$$\mathop{\mathbb{E}}_{x \sim p(x)} [f(x)] \equiv \mathop{\mathbb{E}}_{x} [f(x)] = \int_{x \in X} f(x)p(x). \tag{2.18}$$

2.2.2.4. Covariance and Variance

Given two random variables X and Y, the covariance and variance operations are defined as follows:

$$\text{cov}(X, Y) = \text{cov}(Y, X) = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \hat{x})(y_i - \hat{y}), \tag{2.19}$$

$$\text{var}(X) = \text{cov}(X, X) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \hat{x})(x_i - \hat{x}), \tag{2.20}$$

$$\text{var}(Y) = \text{cov}(Y, Y) = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \hat{y})(y_i - \hat{y}), \tag{2.21}$$

where $\text{cov}(X, Y) \in \mathbb{R}$ is the covariance between X and Y, and $\text{var}(X) \in \mathbb{R}$ and $\text{var}(Y) \in \mathbb{R}$ are the variances of X and Y, respectively. The covariance offers a basic idea of the linear relationship between the two random variables X and Y. If the covariance is positive, then it means that an increase in the values of x is related with an increase in the values of y. If negative, an increase in x is related with a decrease in y. Finally, if the covariance is 0, it indicates that there is no trend between the random variables.

It should be noted that the covariance can only offer a broad notion of the relationship between X and Y. In fact, the magnitude of the covariance does not relate to the strength of the trend. In general, covariance is very difficult to interpret on its own, but it is used to calculate a more interpretable statistic: the correlation.

2.2.2.5. Covariance Matrix

The covariance operation can be extended, for more than two random variables, into the covariance matrix. Given a random vector $\boldsymbol{X} = [X_1, \ldots, X_K]^T$, the covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{K \times K}$ (square matrix) is defined as follows:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \text{var}(X_1) & \ldots & \text{cov}(X_K, X_1) \\ \vdots & \ddots & \vdots \\ \text{cov}(X_1, X_K) & \ldots & \text{var}(X_K) \end{bmatrix}, \tag{2.22}$$

where $\{X_1, \ldots, X_K\}$ are the different random variables used to build the random vector $\boldsymbol{X}$.

Intuitively, the covariance matrix $\boldsymbol{\Sigma}$ can be interpreted as a linear transformation that turns any vector toward the direction of the largest dispersion/variance in the data (assuming mean equals to $\vec{0}$). Given this matrix, it is observed that the eigenvector $\boldsymbol{v}_i$, with largest eigenvalue λ_i, is aligned in the direction of largest general dispersion. Note that, given that $\text{cov}(X_k, X_{k'}) = \text{cov}(X_{k'}, X_k)$, the covariance matrix $\boldsymbol{\Sigma}$ is symmetric; therefore, its eigenvectors are orthogonal, defining a new orthogonal basis space.

2.2.3. Information Theory

The Information Theory (Hockett et al., 1953) consists of the mathematical background of concepts, parameters, and fundamental rules that govern the transmission of messages through communication systems, usually governed by high noise and uncertainty. Information Theory is a concept founded by Claude Shannon (1916–2001), who laid the foundations and fundamental limits that would be exploited and implemented throughout the preceding years in various areas, including the Data Science and Machine Learning research fields.

2.2.3.1. Entropy

One of the fundamental definitions in Information Theory consists of the Shannon entropy $H(p)$, which is a function defined as a measure of the uncertainty of a random variable X. The entropy definition is as follows:

$$H(p) \equiv \mathop{\mathbb{E}}_{x} \left[-\log \left(p(x) \right) \right], \tag{2.23}$$

$$\equiv -\sum_{i=1}^{N} p(x_i) \log \left(p(x_i) \right), \tag{2.24}$$

where X corresponds to a discrete random variable and $p(x)$ is its probabilistic distribution. One way to analyze this definition is to consider that the entropy $H(p)$ consists of the expectation of the information, defined as $-\log \left(p(x_i) \right)$, which is offered by the different outcome events x from a random variable X. Intuitively, rare events, with low probability of occurrence $p(x_i)$, offer more information when they occur. In contrast, frequent or trivial events, with high probability of occurrence, usually do not offer any valuable information.

It should be noted that the definition of entropy can also be directly extended over a framework of continuous random variables, bearing the name of differential entropy. The differential entropy definition is as follows:

$$H(p) \equiv -\int_{x} p(x) \log \left(p(x) \right), \tag{2.25}$$

where X is a continuous random variable.

2.2.3.2. Cross-Entropy

Given two probability distributions associated with the random variable X, $p(x)$ and $q(x)$, the cross-entropy operator is defined as follows:

$$H(p, q) \equiv \mathop{\mathbb{E}}_{x \sim p(x)} \left[-\log \left(q(x) \right) \right], \tag{2.26}$$

$$\equiv -\sum_{i=1}^{N} p(x_i) \log \left(q(x_i) \right), \tag{2.27}$$

where $q(x)$ consists of an approximated distribution of the real distribution $p(x)$ of the random variable X. If the approximation $q(x)$ is adequate and it properly describes the real distribution $p(x)$; then, the cross-entropy is minimized when $q(x) \approx p(x)$, reaching the theoretical lower bound $H(p)$[6].

2.2.4. Machine Learning

Machine Learning (ML) can be described as a broad family of methods and algorithms that are capable of detecting, partially or fully automatically, patterns, structures, and re-

[6] $H(p, q) = H(p) + D_{KL} \left(p \,||\, q \right) \geq H(p)$, where $D_{KL} \left(p \,||\, q \right)$ is the Kullback Leibler divergence between the real distribution $p(x)$ and the approximated distribution $q(x)$.

lationships hidden in a set of data (dataset). Usually, these datasets are often governed by high noise, uncertainty, and other types of ambiguities and challenges.

Currently, the world is rushing headlong into a new paradigm called Big Data. In this paradigm, it is common to find, throughout all areas of knowledge, different scenarios that generate an increasingly abundant amount of data. For instance, in the case of the Internet itself, it already store a massive amount of data, where almost 1,700 Terabytes of data are uploaded each minute[7] and almost 500 hours of video are uploaded each minute on websites like Youtube[8]. Another example is astronomy, where a huge stream of data is expected to be extracted from the cosmos by the new generation of telescopes, such as the future Vera C. Rubin Observatory Legacy Survey of Space and Time (LSST; Ivezić et al., 2019) survey. For instance, for the LSST project, approximately 15 terabytes of raw data are expected per night by observing a large number of astronomical objects up to 37 billion objects in 10 years (Ivezić et al., 2019).

In general, this raw data is not as useful or interesting to study, by itself, as the information that can be extracted from it. As an example, although one can access an image, composed of hundreds or thousands of pixels; ultimately, the value of the data lies in the information contained within the image, e.g., knowing if there are faces in the image, describing textures or geometric shapes, finding and describing letters. This requisite for information extraction, as well as the vertiginous increase in the data quantity, means that the study and processing of the data gets out of hand when it is delegated solely to human experts. For these reasons, this large volume of data needs to be processed automatically in order to extract useful and condensed information, as well as to discover new knowledge about the data and the processes that generate it.

In the Big Data paradigm context, Machine Learning results in an extremely attractive strategy because it allows scientists and engineers to automatically process the increasingly colossal data collections, with little or even no human intervention. Another possible advantage is that the use of Machine Learning methods can help to reduce strong assumptions coming from expert knowledge (human bias) about the proposed problems to solve, letting the algorithms, by themselves, try to find the best solutions by extracting information directly from the raw data.

A frequently accepted way of tackling the problems proposed by Machine Learning is to consider a probabilistic framework or scheme, also called a Bayesian scheme. This approach considers that, in a modeling or approximation problem, there always will exist associated uncertainties; therefore, the best alternative is to consider and deal with them by working according to a probabilistic approach. Some intrinsic encountered uncertainties may be: ¿Is this approximation really the most correct one? ¿Is the model used the best alternative for this problem? ¿Are these data reliable or are they the product of noise and/or interference? among many others.

In general, in the literature, it is very common to divide the Machine Learning research area into three main sub-categories: Supervised Learning, Unsupervised Learning, and Reinforcement Learning. For the sake of simplicity, we will only briefly introduce the first two of

[7] `https://everysecond.io/the-internet`.

[8] `https://everysecond.io/youtube`.

them.

2.2.4.1. Supervised Learning

In Supervised Learning, the main goal is to generate (a.k.a. to learn) a mapping relationship from a data instance $\boldsymbol{x}$ (input vector) to a response $\boldsymbol{y}$ (output vector). This map can be represented as finding a function approximation $\hat{f}(\cdot)$, of a hidden function $f(\cdot)$, that allows us to correctly describe the following relationship: $\hat{\boldsymbol{y}} = \hat{f}(\boldsymbol{x})$[9].

We define the dataset $\mathcal{D}$ as a collection of N individual instances, each one denoted vectorially as follows: $\boldsymbol{x}_i = [x_{i,1}, \ldots, x_{i,D}]^T \in \mathbb{R}^D$, where an input vector $\boldsymbol{x}_i$ aims to describe the i-th data instance by considering D numerical variables, usually called characteristics or features. The vector $\boldsymbol{x}_i$ can be the representation of an image, a video, a time series, a point cloud, a paragraph, etc. For simplicity, the theoretical explanation will be carried out by considering $\boldsymbol{x}_i$ as a vector, but it is directly extensible to another type of more complex mathematical object, such as matrices or tensors (high-dimensional matrix).

In addition, for each input vector $\boldsymbol{x}_i$, we also have the output response $\boldsymbol{y}_i$, also denoted as a vector. This response vector is usually generated from the supervision of a human expert, but it can even be generated by another external algorithm. The vector $\boldsymbol{y}_i$ aims to describe some known output response associated with the input vector $\boldsymbol{x}_i$. Finally, the dataset in a supervised learning scenario can be represented according to the following dataset notation: $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$.

Classification Problem Scenario

A frequent scenario in supervised learning is the classification task. When working on a classification problem, the output variable $\boldsymbol{y}_i = y_i$ is often used as a categorical variable, being an integer of the form $y_i \in \{1, \ldots, C\}$, where C is the number of classes existing in the dataset $\mathcal{D}$. Also, the value of y_i directly indicates the index of the class to which the input vector $\boldsymbol{x}_i$ belongs to. As an example, different types of supernova photometric light-curves are present in a dataset $\mathcal{D}$; then, the supervised categorical variable y_i can be independently generated from a external study performed with complex spectrometry techniques and evaluated by human experts.

Recalling the objective of the input-output mapping relationship $\boldsymbol{x} \mapsto \boldsymbol{y}$, it is possible to express the classification problem according to a probabilistic approach as follows:

$$\hat{y}_i = \hat{f}(\boldsymbol{x}_i) = \underset{c \in \{1,\ldots,C\}}{\arg\max}\, p(y_i = c | \boldsymbol{x}_i; \mathcal{D}, \boldsymbol{\theta}), \tag{2.28}$$

where the function $\hat{f}(\cdot)$, to be approximated, corresponds to a density or conditional probability distribution p that describes the behavior of the categorical variable y_i. Thus, the class distribution of the i-th data instance is conditioned by the characteristics of the input vector

[9]The notation $\hat{\cdot}$ indicates that the mathematical object is an approximation, either of a variable or of a function.

$\boldsymbol{x}_i$, the rest of the dataset $\mathcal{D}$, and the model $\boldsymbol{\theta}$ that is used for modeling or approximating the function.

Given a model $\boldsymbol{\theta}$, the vector $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_M]^T$ corresponds to a set of M parameters that conforms the model, where these parameters can be changed and adjusted by some optimization process, obtaining the optimal values denoted as $\boldsymbol{\theta}^* = [\theta_1^*, \ldots, \theta_M^*]$[10]. In the context of Machine Learning, the process of finding $\boldsymbol{\theta}^*$ is called «training», where it is said that these values are «learned», coining the term «Machine Learning».

For simplicity, it is very common to omit some variables in the formulation of the problem, as well as to handle subscripts indicating the model linked to the approximation of some function, as follows:

$$\hat{y}_i = \hat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \underset{c \in \{1,\ldots,C\}}{\arg\max}\, p(y_i = c|\boldsymbol{x}_i), \tag{2.29}$$

where it is said that the function $f_{\boldsymbol{\theta}}$ is modeled by the model $\boldsymbol{\theta}$. By considering this probabilistic approximation, the most probable class is obtained by finding the maximum of the class distribution over the classes $c \in \{1, \ldots, C\}$. Finding this value corresponds to finding the mode of the distribution of y_i; therefore, finding the best bet or approximation of $\hat{y}_i$ associated with the input vector $\boldsymbol{x}_i$. This methodology is also known as the Maximum a Posteriori estimation (MAP)[11].

Regression Problem Scenario

Another very common supervised learning scenario is the regression task. In this scenario, if we want to regress a variable, we have that $\boldsymbol{y}_i = y_i$ is a number value of the form $y_i \in \mathbb{R}$, which represents a response associated with the input vector $\boldsymbol{x}_i$. In such a case, the probabilistic model is similar to the one described above, being of the form $p(y_i|\boldsymbol{x}_i; \mathcal{D}, \boldsymbol{\theta})$. As an example, a regression problem could be to perform a regression of the brightness flux from a supernova light-curve.

2.2.4.2. Unsupervised Learning

In the unsupervised learning scenario, we have a dataset composed of the input vectors only: $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} = \{\boldsymbol{x}_i\}_{i=1}^N$. Because there is no externally characterized supervised output vector $\boldsymbol{y}_i$ (as in the supervised scenario), the goal of unsupervised learning is to perform knowledge discovery, i.e., finding hidden and interesting patterns, structures, and/or relationships in the data.

In the absence of a supervised output variable linked to an input variable, the unsupervised problem is formalized as a probabilistic density estimation problem, where the idea is to express models as $p(\boldsymbol{x}_i|\mathcal{D}, \boldsymbol{\theta})$. In this sense, there are two main differences w.r.t. the supervised case: 1) In the supervised case a conditional density must be estimated, while in the

[10] The notation $\cdot^*$ is used to denote the optimal values reached after some optimization process.

[11] In this case, the MAP estimation is performed over the variable y_i and not over the model $\boldsymbol{\theta}$. The latter case will be discussed later in section §2.2.5.2.

unsupervised case this distribution is non-conditional. 2) The fact that $\boldsymbol{x}_i$ is a vector makes the density estimation to consider multi-modal distributions, which drastically increases the complexities and challenges for solving the problem.

In general, the unsupervised scenario is a highly challenging scenario because the problem to be solved is usually not very intuitive, especially considering that usually there is no a clear idea of the hidden patterns that we are actually trying to find. The great advantage is that in the unsupervised scenario it is not necessary (or explicitly required) to have supervised variables, which are usually very expensive to obtain (or even impossible) in some cases since the effort of human experts is required.

2.2.4.3. Semi-Supervised Learning

It is often possible to find cases where, although there is an supervised output vector $\boldsymbol{y}_i$, it is only partially characterized for a subset of input vectors $\boldsymbol{x}_i$. Then, in the semi-supervised learning scenario, techniques and strategies from both, the supervised and the unsupervised learning, are used to solve the problem.

2.2.4.4. Models and Algorithms

Currently, there are many algorithms, also called models, belonging to the Machine Learning family. A major distinction of these algorithms is usually made by noting whether these models are considered as parameterized or non-parameterized models.

The parameterized models consist of models $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_M]^T$ whose number of parameters M is fixed. An advantage of having a fixed defined quantity is that these models are usually fast and computationally less expensive. A disadvantage is that it is necessary to make strong assumptions about the distribution of the data, which may not always be correct or may only consider particular cases and circumstances. Some examples of parameterized models are the following: Naive Bayes Classifier, Linear Regression, Logistic Regression, Artificial Neural Networks (ANNs)[12], among others.

On the other hand, in the case of the non-parameterized models, the number of parameters M that conforms the model $\boldsymbol{\theta}$ usually grows w.r.t. the amount of data worked. This normally translates into higher computational times and costs as the dataset $\mathcal{D}$ grows into a larger volume. Some examples of non-parameterized models are the following: Classification and Regression Trees (CART), Random Forest (RF), Support Vector Machine (SVM), k-Nearest Neighbors (kNN), Gaussian Process (GP), Kernel Density Estimation (KDE), Genetic Algorithms (GAs), Particle Swarm Optimization (PSO), among others.

[12]Along with the whole wide family of ANN-based models, some of which will be studied in the following sections.

2.2.5. Bayesian Learning

The Bayesian Learning corresponds to a probabilistic framework based on the Bayes theorem. This concept is radically important to understand the basis of many types of Machine Learning algorithms, such as the ANN-based model optimization.

2.2.5.1. Bayes Theorem

Rev. Thomas Bayes (1702–1761) was an English Nonconformist Prebyterian minister and author of one of the most influential ideas in the probability theory: the Bayes theorem. The Bayes theorem is defined as follows:

$$\textbf{posterior} = p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} = \frac{\textbf{likelihood} \cdot \textbf{prior}}{\textbf{marginalization}}, \tag{2.30}$$

where the detailed explanation is as follows[13]:

1. **Model**: the model $\boldsymbol{\theta}$ corresponds to an hypothesis of a possible model that aims to explain the empirical observations of the data $\mathcal{D}$. It should be clarified that the model $\boldsymbol{\theta}$ comes from the space of all possible models that can be constructed, i.e., all possible combinations of parameters that generate the vector $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_M]^T$[14].

2. **Prior**: the prior term $p(\boldsymbol{\theta})$ corresponds to the probability of occurrence of the model $\boldsymbol{\theta}$. The prior is used to induce any kind of knowledge about the context of the given problem. For example, by using the prior, one can assign a higher probability on certain models that are known to be more plausible than others. Likewise, a non-informative prior can be used when all models are equally probable in the prior distribution. In this case, the lack of preliminary knowledge can be expressed by using a prior distribution defined as an uniform distribution over the models.

3. **Likelihood**: the likelihood term $p(\mathcal{D}|\boldsymbol{\theta})$ corresponds to the conditional likelihood of the observations from the data $\mathcal{D}$ given a fixed model $\boldsymbol{\theta}$. The likelihood attempts to express the likelihood of having actually observed the data $\mathcal{D}$ given the model choice (not necessarily optimal) of a model that aims to describe the data observations. Note that, if we considers a number of N data observations $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$, then the likelihood must consider the joint probability of all observation as follows: $p(\mathcal{D}|\boldsymbol{\theta}) = p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N|\boldsymbol{\theta})$.

4. **Marginalization**: the marginalization term $p(\mathcal{D})$ consists of the marginal likelihood of the observed data $\mathcal{D}$. Theoretically, this probability can be expressed as $p(\mathcal{D}) = \sum_{\boldsymbol{\theta}'} p(\mathcal{D}, \boldsymbol{\theta}') = \sum_{\boldsymbol{\theta}'} p(\mathcal{D}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')$ in the discrete case[15], which is a marginalization w.r.t. the full domain of models. In addition, $p(\mathcal{D})$ could be directly considered as a constant

[13] It is common to find this explanation based on the concept of hypothesis H and evidence E. For this work, the hypothesis H can be assumed directly as the hypothesis of considering a possible configuration of the model $\boldsymbol{\theta}$ which attempt to explain the empirical observations (i.e., the evidence E) from the data $\mathcal{D}$.

[14] In a real scenario, we only work with a bounded set of possible solutions for the model $\boldsymbol{\theta}$ due to computational numerical constraints.

[15] $p(\mathcal{D}) = \int_{\boldsymbol{\theta}'} p(\mathcal{D}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')$ in the continuous case.

term since it is a scaling factor, ensuring that the final result (posterior term) effectively corresponds to probabilities coming from a well-defined distribution that satisfies the associated conditions, e.g., $\sum_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D}) = 1$[16].

5. **Posterior**: the posterior term $p(\boldsymbol{\theta}|\mathcal{D})$ corresponds to the conditional probability of the model $\boldsymbol{\theta}$ given the empirical observation of the data $\mathcal{D}$. The posterior is simply the multiplication of the likelihood with the prior probability, finally scaled by the marginalization term.

Therefore, the posterior is a distribution that exposes which models $\boldsymbol{\theta}$ are the most or least likely given that the data observations $\mathcal{D}$ have already been empirically observed. In general, with sufficient data, the interest lies in studying the modes of the posterior distribution. Usually, in those distribution regions, the models that are statistically the most adequate to explain the observations from the data $\mathcal{D}$ are presented.

An interpretation of Bayesian Learning is that it consists of a scheme where the idea is to be able to update the current hypotheses as new evidence is gathered and empirically observed from the world, i.e., to update and correct the initial beliefs about a model $\boldsymbol{\theta}$ after having observed the data $\mathcal{D}$. In that sense, it is even possible to dynamically update and replace the prior distribution, with the posterior distribution, as follows: $p(\boldsymbol{\theta}) \leftarrow p(\boldsymbol{\theta}|\mathcal{D})$, a process that is also known as Bayesian Update.

2.2.5.2. Maximum A Posteriori (MAP) Estimation

As previously mentioned, the key idea of the Bayesian Learning is to be able to find an optimal model $\boldsymbol{\theta}^*$ that can adequately explain the empirical observations in the data $\mathcal{D}$. One formulation of this problem is the Maximum a Posteriori (MAP) estimation over the model distribution, which is defined as follows:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D}) = \arg\max_{\boldsymbol{\theta}} \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}, \tag{2.31}$$

$$= \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}), \tag{2.32}$$

where this expression states that the optimal model $\boldsymbol{\theta}^*$ is presented in the mode of the distribution of models $\boldsymbol{\theta}$ given the empirical observations of the data $\mathcal{D}$.

Intuitively, the MAP estimation attempts to solve the following questions:

1. Given the set of empirical data observations $\mathcal{D}$ ¿What is the distribution of possible models $\boldsymbol{\theta}$ that attempt to explain them?

2. According to this model distribution, where the most probable model is the one that can best explain the behavior of the data observations $\mathcal{D}$ ¿Which is the most probable model, i.e., the optimal model $\boldsymbol{\theta}^*$?

[16] $\int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D}) = 1$ in the continuous case.

Eq. (2.32) is valid since the data $p(\mathcal{D})$ is an expression that does not depend on the model $\boldsymbol{\theta}$ for the maximization problem. Also, the following proportion holds: $\frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$. This results in a great advantage as the term $p(\mathcal{D})$ is intractable in the vast majority of practical cases. The intractability of the term $p(\mathcal{D})$ easily occurs in scenarios where the models possess high-dimensional vectors, where the integral $\int_{\boldsymbol{\theta}'} p(\mathcal{D}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')$ hardly have a known or even computable analytical solution.

2.2.5.3. Maximum Likelihood Estimation (MLE)

It is often common to simplify the MAP estimation into the Maximum Likelihood Estimation (MLE), which is defined as follows:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}|\mathcal{D}), \tag{2.33}$$

$$\ell(\boldsymbol{\theta}|\mathcal{D}) = p(\mathcal{D}|\boldsymbol{\theta}), \tag{2.34}$$

where eq. (2.34) is obtained by directly considering the following proportion: $\frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \propto p(\mathcal{D}|\boldsymbol{\theta})$.

It can be observed that, in MLE, it is not necessary to work with a prior distribution for the model $\boldsymbol{\theta}$: the maximization optimization depends only on the likelihood distribution. Also, it is important to note that the term $\ell(\boldsymbol{\theta}|\mathcal{D})$ is called the likelihood function, which does not necessarily holds the condition $\sum_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}|\mathcal{D}) = 1$; therefore, it does not correspond to a probability distribution of the models.

In summary, in the MLE context, we seek to explain how plausible an arbitrary model $\boldsymbol{\theta}$ is given the data observation $\mathcal{D}$. Then, finding the maximum of the likelihood function is equivalent to finding a potentially optimal model $\boldsymbol{\theta}^*$.

Linear Regression Example

In this section, we present a brief example of the regression optimization problem by using the aforementioned Bayesian Learning scheme. Given that the regression task can be considered as a supervised learning problem, we define the observations as follows: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$. In this case, the problem is simplified by considering observations of the input values x_i and output values y_i as scalar values. By assuming that all output values y_i are independent of each other, then the likelihood function is defined as follows:

$$\ell(\boldsymbol{\theta}|\mathcal{D}) = p(y_1, \ldots, y_N|x_1, \ldots, x_N; \boldsymbol{\theta}), \tag{2.35}$$

$$= p(y_1|x_1, \ldots, x_N; \boldsymbol{\theta}) \ldots p(y_N|x_1, \ldots, x_N; \boldsymbol{\theta}), \tag{2.36}$$

$$= \prod_{i=1}^{N} p(y_i|x_1, \ldots, x_N; \boldsymbol{\theta}), \tag{2.37}$$

then, by assuming that each output value y_i depends only on the associated input value x_i and the model $\boldsymbol{\theta}$, the likelihood function can be expressed as follows:

$$\ell(\boldsymbol{\theta}|\mathcal{D}) = \prod_{i=1}^{N} p(y_i|x_i;\boldsymbol{\theta}), \tag{2.38}$$

where, to further analytically work this expression, the following assumptions are required:

1. All output values y_i follow an i.i.d.[17] Gaussian distribution scheme of the form $y_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, where the fixed standard deviation σ_i is defined as follows: $\sigma_i = \sigma, \forall i \in \{1, \ldots, N\}$.
2. The mean of the i-th Gaussian distribution is defined as follows: $\mu_i = \boldsymbol{W}^T \boldsymbol{x}_i$. In this case, the model is defined as $\boldsymbol{\theta} = \boldsymbol{W} = [b, a]^T \in \mathbb{R}^2$, where a is a scale and b is an offset. In addition, a notation trick is used to represent the original input values in a vectorial form as follows: $\boldsymbol{x}_i = [1, x_i]^T \in \mathbb{R}^2$. In this way, it is straightforward to obtain a linear expression for an output estimation: $\hat{y}_i = \boldsymbol{W}^T \boldsymbol{x}_i = a \cdot x_i + b$.

Eq. (2.38) is analytically worked as follows:

$$\ell(\boldsymbol{\theta}|\mathcal{D}) = \prod_{i=1}^{N} p(y_i|x_i;\boldsymbol{\theta}), \tag{2.39}$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_1 - \mu_1)^2}{2\sigma^2}\right) \ldots \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_N - \mu_N)^2}{2\sigma^2}\right), \tag{2.40}$$

and by taking the logarithm of the likelihood:

$$\log\left(\ell(\boldsymbol{\theta}|\mathcal{D})\right) = c - \frac{(y_1 - \mu_1)^2}{2\sigma^2} \ldots - \frac{(y_N - \mu_N)^2}{2\sigma^2}, \tag{2.41}$$

$$= c - \frac{\sum_{i=1}^{N}(y_i - \mu_i)^2}{2\sigma^2}, \tag{2.42}$$

$$= c - \frac{\sum_{i=1}^{N}(y_i - \mathbf{w}^T \boldsymbol{x}_i)^2}{2\sigma^2}, \tag{2.43}$$

where c is an arbitrary constant term. Finally and returning to the maximization optimization problem, the optimal model can be expressed as follows:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}|\mathcal{D}) = \arg\min_{\boldsymbol{\theta}} -\log\left(\ell(\boldsymbol{\theta}|\mathcal{D})\right), \tag{2.44}$$

$$= \arg\min_{\boldsymbol{\theta}} \frac{\sum_{i=1}^{N}(y_i - \mathbf{w}^T \boldsymbol{x}_i)^2}{2\sigma^2}, \tag{2.45}$$

$$= \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N}(y_i - \mathbf{w}^T \boldsymbol{x}_i)^2, \tag{2.46}$$

$$= \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N}(y_i - (a \cdot x_i + b))^2, \tag{2.47}$$

[17] Independent and identically distributed.

where, in eq. (2.47), the Mean Squared Error (MSE), between the empirical output value y_i and the approximation value $\hat{y}_i = a \cdot x_i + b$, is computed. The MSE expression is highly important as it is widely used in the literature. This is because it has a well-known analytical solution and it is directly extensible to the high-dimensional case when the input values are vectors.

2.2.6. Iterative Optimization Algorithms

Although the previously example of the linear regression does have a well-known analytical solution, this is not the case for the majority of real-world problems, where it is almost impossible to find a closed analytical solution for the MAP or the MLE schemes. Given this challenge, it is then necessary to incorporate a mechanism capable of dealing with the complex optimization problems proposed in Machine Learning in general, without relying on the existence of closed analytical solutions. Therefore, in this section, some relevant algorithmic solutions for this challenge are described, which are iterative optimization algorithms based on Gradient Descent.

2.2.6.1. Gradient Descent (GD)

The Gradient Descent algorithm (GD) describes the iterative model update rule according to the error evaluation over an optimization surface. According to the GD algorithm, the model $\boldsymbol{\theta}$ update rule is described as follows:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \tag{2.48}$$

where the function J, also known as the target or loss function, is the function that is required to be minimized when evaluated over an optimal model $\boldsymbol{\theta}^*$. The term $\nabla_{\boldsymbol{\theta}} J$ is the gradient of the loss function J w.r.t. the model $\boldsymbol{\theta}$. The factor η corresponds to the model update factor, which is also called the «learning rate».

Intuitively, the minimization over the loss function or surface J is required to solve the proposed optimization problem. The assumption is that several local minimum points in J may exist associated with different sets of model parameters $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_M]^T$. In GD, given a model $\boldsymbol{\theta}$, we assume the existence of a nearest local minimum where the loss function is minimized as follows: $J^* = J(\boldsymbol{\theta}^*)$, where $\boldsymbol{\theta}^*$ is the evaluated optimal model that minimizes the loss function J. To find this optimal model, small update changes are applied over the model parameters by following the direction pointing to the nearest local minimum in J^*. Then, given a model $\boldsymbol{\theta}$, the update direction over the optimization surface can be computed as the opposite direction defined by the gradient of the loss function: $-\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$.

Given the previously defined model update direction, the model parameters are changed in order to perform small update steps towards the nearest local minimum loss function value J^* evaluated over the optimal model $\boldsymbol{\theta}^*$. The idea is to iteratively perform small update steps of the model, updating the parameters and the update direction $-\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ at each optimization iteration. This iterative process allows the model parameters to be slowly updated towards the local minimum solution at $\boldsymbol{\theta}^*$, eventually reaching the local minimum with an adequate

Algorithm 1: Stochastic Gradient Descent pseudo-code.

for epoch $\in$ epochs **do**
  Randomly shuffle the instances within the dataset $\mathcal{D}$
  Generate a number of $\frac{\#(\mathcal{D})}{N}$ mini-batches from the dataset $\mathcal{D}$// Where $N = \#(\mathcal{D}_{\text{mb}})$
  for mini-batch $\in$ mini-batches **do**
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\eta}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta})$// Compute the gradient estimation using the mini-batch $\mathcal{D}_{\text{mb}}$

number of optimization iterations. Finally, the strength of the update changes towards the local minimum is controlled by the learning rate factor η, performing small update changes if η is also an small value.

2.2.6.2. Stochastic Gradient Descent (SGD)

The Stochastic Gradient Descent algorithm (SGD) corresponds to the practical extension of GD, where the required gradient computation is estimated from a subset of instances $\mathcal{D}_{\text{mb}}$ within a dataset $\mathcal{D}$. The subset $\mathcal{D}_{\text{mb}}$ is known as the «mini-batch» set. Then, the model update rule in the SGD algorithm is described as:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \left(\frac{1}{N} \sum_{i=1}^{N} J_i(\boldsymbol{\theta}) \right) = \boldsymbol{\theta} - \frac{\eta}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}), \tag{2.49}$$

where N is the mini-batch subset size: $N = \#(\mathcal{D}_{\text{mb}})$. The main idea of SGD it to compute a gradient approximation of the original GD formulation as follows: $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta})$, where $J_i(\boldsymbol{\theta})$ is the loss function value evaluated on the i-th instance in the mini-batch subset $\mathcal{D}_{\text{mb}}$. This mini-batch is usually a random and smaller selection of instances from the complete dataset $\mathcal{D}$.

Although the SGD algorithm has an stochastic behavior when building the different mini-batches $\mathcal{D}_{\text{mb}}$, the optimization is still performed in such a way that all instances within the dataset $\mathcal{D}$ are finally evaluated on the loss function. As an extra explanation, algorithm 1 describes the SGD pseudo-code. In SGD, each evaluation and update step, using a mini-batch, is called an «iteration». In addition, an «epoch» is passed once all the mini-batches are used to evaluate the loss function, completing the evaluation of all the instances within the original dataset $\mathcal{D}$.

Noted that, in the context of real-world optimization problems, the implementation of the GD algorithm is almost impossible due to the high computational cost and memory that is required, specially with high-volume datasets expected in the Big Data paradigm. For this reason, the implementations are usually framed with the key concepts used in the SGD algorithm, where the use of mini-batches, iterations, and epochs is specially important to ensure tractable computational costs during the optimization.

In addition, the stochastic behavior of the SGD algorithm allows the optimization process to avoid and even escape from possible sub-optimal local minimum surfaces along the opti-

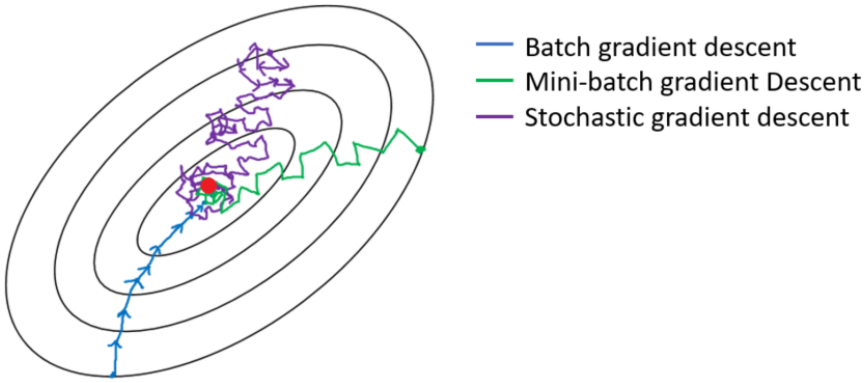

Figure 2.1: Illustrative comparison between the GD and the SGD optimization algorithms. A smooth and straightforward path towards the local surface minimum (red point) can be observed for the GD when using the complete dataset to update (blue path). A moderate stochasticity can be also observed when using a mini-batch routine for the GD algorithm (green path). For the SGD, a highly stochastic behavior can be observed, producing random changes along the path and reaching the minimum of the surface in a longer number of iterations (purple path). Original figure extracted from `https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4`.

mization path. Fig. 2.1 illustrates a comparison between the GD and the SGD optimization algorithms over a 2D optimization surface.

2.2.6.3. Other Optimization Algorithms

Based on the GD and SGD algorithm, a vast family of optimization algorithms has been proposed in Machine Learning, being the optimization algorithm design a very active research field. Some notable optimizer examples are the following: Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012), Momentum Optimizer (MO), Root Mean Square Propagation (RMSProp), Adaptive Moment Estimation (Adam; Kingma & Ba, 2015), AdamW (Loshchilov & Hutter, 2017), among many others.

2.2.7. Monte Carlo Method

The Monte Carlo is an experimental methodology that incorporates, to a greater or lesser extent, some mechanism of empirical sampling from a stochastic experiment to be studied. Although the MC usage cases cover a large number of algorithms and strategies, they all share the characteristic of considering empirical sampling to estimate some function or statistic of interest.

To illustrate the Monte Carlo methodology, a random variable with Gaussian distribution $x \sim \mathcal{N}(\mu, \sigma^2)$ and an arbitrary function $f(\cdot)$ are defined as an example. According to the

Monte Carlo method, an estimate of the expected value on such function, based on a collection of empirical samples, is expressed as follows:

$$\mathop{\mathbb{E}}_{x}[f(x)] = \int_x p(x)f(x) \approx \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} f(x_i), x_i \sim p(x), \tag{2.50}$$

where x_i corresponds to the i-th empirical sample obtained from the random variable X. In this case, the quality of the empirical estimation is expected to improve as the number of samples N also increases.

For instance, in this example, if $f(x) = x$ is used, then the Monte Carlo method would give an estimate of $\mathbb{E}_x[f(x)] = \mathbb{E}_x[x] = \mu \approx \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} x_i, x_i \sim p(x)$, where μ is the mean of the Gaussian distribution. Noted that the estimates may be only adequate if there is a large number of samples N.

Finally, given the practical difficulties associated with the generation of samples from a Gaussian distribution (e.g., infinite domain), the above estimation can be also described in terms of the inverse transform method as follows:

$$\mathop{\mathbb{E}}_{x}[f(x)] \approx \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} f\big(F_X^{-1}(z_i)\big), \tag{2.51}$$

where z is sampled from an uniform distribution $z \sim Z = \mathrm{U}(0,1)$ and F_X^{-1} is the inverse CDF of the Gaussian distribution p_X.

2.2.8. Markov Chain Model

The Markov Chain is a statistical model that aims to describe the evolution of a dynamic system between different possible discrete states for a random variable X. This evolution can operate over sequential iterations, sequential steps, or discrete changes in time.

In this probabilistic framework it is assumed that the state changes are stochastic; therefore, changing from a previous state $X_{j-1} = a_{j-1}$ to a current state $X_j = a_j$ is subject to a probability by considering that there exist a set of possible states: $\{a_1, \ldots, a_N\}$. This transition probability is expressed as follows:

$$p_{a_{j-1}\mapsto a_j} = p_{a_{j-1},a_j} = p(X_j = a_j | X_{j-1} = a_{j-1}, X_{j-2} = a_{j-2}, \ldots, X_1 = a_{j=1}), \tag{2.52}$$

where it is observed that the current state of X_j, at the sequence step j, depends explicitly on all the previous states (the complete evolution of the system) starting from the initial state $X_1 = a_{j=1}$.

A Markov Chain must satisfies the Markov property that is defined as follows:

$$\begin{aligned} p_{a_{j-1}\mapsto a_j} &= p_{a_{j-1},a_j} = p(X_j = a_j | X_{j-1} = a_{j-1}, X_{j-2} = a_{j-2}, \ldots, X_1 = a_{j=1}), && (2.53)\\ &= p(X_j = a_j | X_{j-1} = a_{j-1}), && (2.54) \end{aligned}$$

where the distribution over the current state X_j, given all the history of the system, depends only on the most immediate past X_{j-1}. Intuitively, this expresses that knowing the

states $X_{j-2}, \ldots, X_1$ is irrelevant when describing the current state X_j since such evolution is implicitly «stored» in the previous state X_{j-1}. This property is of great help as it removes the need to follow the complete history of a dynamical system, which can be prohibitively expensive in some cases.

The probabilistic change of states in a Markov Chain can be expressed into the following square matrix P, also called state transition matrix:

$$P = \text{from states} \left\{ \overbrace{\begin{bmatrix} p_{a_1,a_1} & \cdots & p_{a_1,a_N} \\ \vdots & \ddots & \vdots \\ p_{a_N,a_1} & \cdots & p_{a_N,a_N} \end{bmatrix}}^{\text{to states}} \right. \tag{2.55}$$

where the entries of the matrix P follow the form $p_{a_k \mapsto a_m} = p_{a_k,a_m} = p(X_j = a_m | X_{j-1} = a_k)$ which is the conditional probability of changing from the previous state a_k to the current state a_m. Note that the sum across the rows of this matrix must sum up to one: $\sum_{m=1}^{N} p_{a_k,a_m} = 1, \forall k$.

Additionally, an auxiliary vector $\boldsymbol{\pi}_{j-1}$, conformed of the probabilities of being in any of the N states at a certain sequence step j, is defined as follows:

$$\boldsymbol{\pi}_{j-1} = [\pi_{1,j}, \ldots, \pi_{N,j}]^T = [p(X_{j-1} = a_1), \ldots, p(X_{j-1} = a_N)]^T, \tag{2.56}$$

where this vector is constructed by using the marginal probabilities of each possible state, satisfying the following condition: $\sum_{k=1}^{N} \pi_{k,j} = 1, \forall j$. Therefore, the evolution of a dynamical system, from one sequence step $j-1$ to another sequence step j, can be expressed as follows:

$$\boldsymbol{\pi}_j = P^T \boldsymbol{\pi}_{j-1}. \tag{2.57}$$

2.2.8.1. Stationary Distribution

Depending on the values of the state transition matrix P, the dynamical system can converge to a particular and stable distribution. This distribution is called the stationary distribution of the system. To find this distribution, it is necessary to solve the following system:

$$\boldsymbol{\pi} = P^T \boldsymbol{\pi}, \tag{2.58}$$

$$\sum_{k=1}^{N} \pi_k = 1 | \pi_k \geq 0, \forall k, \tag{2.59}$$

which arises by imposing the condition $\boldsymbol{\pi}_{j-1} = \boldsymbol{\pi}_j = \boldsymbol{\pi}$, where $\boldsymbol{\pi}$ is the stationary distribution. Intuitively, this expression indicates that, given a number of evolution steps of the dynamical system, the vector with the marginal distributions $\boldsymbol{\pi}$ will not change by the linear projection P, reaching the steady state of the system defined under the stationary distribution in $\boldsymbol{\pi}$.

Eq. (2.58) describes the same restriction imposed in the extraction of the eigenvectors and eigenvalues of a matrix. Therefore, to find the stationary distribution of the system $\boldsymbol{\pi}$, it is required to find the eigenvectors and eigenvalues of the matrix P. In addition, in eq. (2.59), an extra condition is presented to ensure that the elements of the eigenvectors correspond to a valid distribution.

2.2.9. Monte Carlo Inference

The Monte Carlo inference is a strategy that uses the generation of empirical samples for the estimation of probability distributions by using the Monte Carlo method. By its nature, this strategy is particularly useful and necessary in scenarios involving the estimation of intractable distributions, i.e., those without the existence of trivial or computable analytical solution.

An example of a complex distribution estimation is the posterior distribution in the context of Bayesian inference:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}, \tag{2.60}$$

where the term $p(\mathcal{D})$ (marginalization constant) is generally intractable in practical problems, especially when high-dimensional models $\boldsymbol{\theta}$ are used.

2.2.9.1. Markov Chain Monte Carlo (MCMC)

The Markov Chain Monte Carlo (MCMC) is a Monte Carlo inference algorithm based on the Markov Chain statistical model. An example of an MCMC algorithm is the popular Metropolis Hastings algorithm (Chib & Greenberg, 1995).

2.2.9.2. Metropolis Hastings (MH)

The Metropolis Hastings (MH) is a well-known algorithm belonging to the MCMC family. This algorithm aims to estimate the posterior distribution $p(\boldsymbol{\theta}|x, y)$, of a model $\boldsymbol{\theta}$, by starting with a certain initial condition $\boldsymbol{\theta}_1$. Algorithm 2 describes the iterative estimation according of the MH algorithm.

A common design in the literature is to choose an isotropic Gaussian proposal distribution: $q(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1}) = \mathcal{N}(\boldsymbol{\theta}_k; \boldsymbol{\theta}_{k-1}, I\sigma_\tau^2)$, which considers a Gaussian hypersphere centered on the previous model $\boldsymbol{\theta}_{k-1}$. The standard deviation σ_τ gives us an idea of the size of the random jump that is realized in the hyperspace when sampling the candidate model $\boldsymbol{\theta}_k^{(c)}$. Intuitively, this algorithm will accept and retain the candidate models that obtain a higher posterior probability w.r.t. the previous iterations: $p(\boldsymbol{\theta}_k^{(c)}|x, y) > p(\boldsymbol{\theta}_{k-1}|x, y)$.

Since the candidate model update is based on a probability, this algorithm works over the Markov Chain assumption: after a large number of iterations, it is expected that the chain will reach the equilibrium, converging to the stationary distribution of the dynamic system. At this theoretical point, the generated samples of the algorithm will correspond to samples obtained from an adequate estimation of the posterior distribution $p(\boldsymbol{\theta}|x, y)$[18].

[18]While having access to a mechanism for generating samples from an estimated distribution is extremely useful, it is also possible to estimate the posterior distribution using other techniques based on histogram computation or KDE.

Algorithm 2: Metropolis Hastings MCMC algorithm.

A proposal distribution $q(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1})$ is selected// this distribution describes the probability of a new model $\boldsymbol{\theta}_k$ given a previous model $\boldsymbol{\theta}_{k-1}$.
A model initial condition is generated $\boldsymbol{\theta}_1$
for $k \in \{2, \ldots, K\}$ **do**
    A candidate model is sampled from $\boldsymbol{\theta}_k^{(c)} \sim q(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1})$
    The posterior probability is computed by using the new candidate model $\boldsymbol{\theta}_k^{(c)}$:
    $p(\boldsymbol{\theta}_k^{(c)}|x, y) \propto p(y|\boldsymbol{\theta}_k^{(c)}, x)p(\boldsymbol{\theta}_k^{(c)}|x) = p(y|\boldsymbol{\theta}_k^{(c)}, x)p(\boldsymbol{\theta}_k^{(c)})$
    The posterior probability is computed by using the previous model $\boldsymbol{\theta}_{k-1}$:
    $p(\boldsymbol{\theta}_{k-1}|x, y) \propto p(y|\boldsymbol{\theta}_{k-1}, x)p(\boldsymbol{\theta}_{k-1}|x) = p(y|\boldsymbol{\theta}_{k-1}, x)p(\boldsymbol{\theta}_{k-1})$
    The acceptance probability of the candidate model $\boldsymbol{\theta}_k^{(c)}$ is computed:
    $\alpha = \min\left\{1, \frac{p(y|\boldsymbol{\theta}_k^{(c)}, x)p(\boldsymbol{\theta}_k^{(c)})}{p(y|\boldsymbol{\theta}_{k-1}, x)p(\boldsymbol{\theta}_{k-1})}\right\}$
    $r \sim \mathrm{U}(0, 1)$
    if $r < \alpha$ **then**
        $\boldsymbol{\theta}_k = \boldsymbol{\theta}_k^{(c)}$// the candidate model $\boldsymbol{\theta}_k^{(c)}$ is accepted: $\boldsymbol{\theta}_k^{(c)}$ is used to update the model.
    else
        $\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1}$// the candidate model $\boldsymbol{\theta}_k^{(c)}$ is rejected: $\boldsymbol{\theta}_{k-1}$ is used to update the model.

A notable advantage of the MH is that it allows us to generate samples from a posterior distribution even if the analytical solution of that distribution is unknown. This is remarkably important and useful in many practical cases, especially when using high-dimensional models $\boldsymbol{\theta}$. In that sense, to achieve the estimation of the posterior distribution, it is only required to compute the product between the probabilities from the likelihood and the prior distributions: $p(y|\boldsymbol{\theta}_k^{(c)}, x)p(\boldsymbol{\theta}_k^{(c)})$.

In contrast, this algorithm present some disadvantages which are described as follows:

1. **Initial condition sensitivity**: given the Markov chain assumption, the MH algorithm has a high sensitivity to the initial condition $\boldsymbol{\theta}_1$. A commonly accepted way to deal with this problem consists of the «burning period». This strategy consists of defining a certain number of initial iterations only as the evolution of the system, discarding them as invalid samples from the distribution to be estimated. For example, by considering a burning period of k', the valid samples are defined in the following iterations: $k \geq k'$.
2. **Markov Chain correlation**: given the Markov Chain assumption in the MH algorithm, the model samples $\boldsymbol{\theta}_k$ and $\boldsymbol{\theta}_{k-1}$ can be strongly correlated. This behavior is incorrect as it is assumed that these samples come from independent samples from the estimated distribution. A solution for this problem could be to consider as valid samples only intermediate samples within the chain, a process called «thinning». For example, when considering a certain thinning value of k', then the valid samples iterations must satisfy $k \,\%k' = 0$, where $\%$ is the modulus operator. However, it is not always possible to ensure uncorrelated samples as this strongly depends on the distribution to be estimated and the data. Therefore, it is very important to study this behavior empirically.

3. **Long convergence time**: a major difficulty associated with the MH algorithm is that the Markov Chain assumption precludes the implementation of a parallelizable computation of the iterative evolution of the Markov Chain, i.e., it is required to perform the computation in the previous iteration $k-1$ to perform the current iteration k. Because of this, the convergence time of the algorithm is usually quite high. One way to mitigate this problem is to implement, in parallel, the evolution of multiple Markov Chains, each one independent of each other (and with different initial conditions). This has two main advantages: 1) Samples from all chains can be considered valid simultaneously once they have reached the equilibrium distribution. At this point, it is even common to concatenate the samples from the chains. 2) Since each parallel chain has a different initial condition, this can also help to deal with the sensitivity of the algorithm to initial conditions.

2.2.10. Artificial Neural Networks (ANNs) and Deep Learning

Throughout history, nature has inspired humans in the development of science and technology. For instance, the first prototypes of flight machines were inspired on birds. In the search of building intelligent machines, it results natural then to take inspiration from nature and from the core of the human intelligence: our own brains. The Artificial Neural Networks (ANNs) are versatile models inspired on the biological neuron networks that conform the animal brain.

The ANNs are known to have the flexibility to build highly complex architectures with a large number of free parameters and possible architecture designs. This flexibility has been the very core of the «Deep Learning» paradigm. Since the formulation of the perceptron model, more complex, powerful, and scalable automatic learning system architectures have arisen in the literature of computer science. In general, this modern phenomenon, coined as the Deep Learning paradigm, considers the use of big and complex systems to estimate functions and distributions, allowing scientists to solve challenging real-world problems that were considered impossible decades ago (e.g., computational vision, Natural Language Processing (NLP), image generation). For example, millions or even billions of parameters can conform a model, such as the GTP model (Radford et al., 2019): a famous and big model used to solve several NLP tasks.

2.2.10.1. Perceptron Model

The perceptron (Rosenblatt, 1958) is a mathematical model inspired and designed on how the biological neuron cells operate and interact between each other. In this model, an output value y can be computed given a set of different input values (cell units). The perceptron model is defined as follows:

$$y = \phi(x_1 w_1 + \cdots + x_N w_N), \tag{2.61}$$

$$= \phi\big(\boldsymbol{w}^T \boldsymbol{x}\big), \tag{2.62}$$

where $\boldsymbol{x} = [x_1, \ldots, x_N]^T$ is the input vector, containing N input values. The vector $\boldsymbol{w} = [w_1, \ldots, w_N]^T$ contains the N weights of the model, each one associated with an input value. Finally, the function $\phi : \mathbb{R} \mapsto \mathbb{R}$ is the activation function, which is usually defined as a nonlinear function. Some commonly used nonlinear functions are the hyperbolic tangent function, sigmoid, Rectified Linear Unit (ReLU), softmax, among many others. In addition, usually an optional constant term is added to the above expression as follows:

$$y = \phi\big(\boldsymbol{w}^T\boldsymbol{x} + b\big), \tag{2.63}$$

where $b \in \mathbb{R}$ is the bias term.

In general, the weight vector $\boldsymbol{w}$ and the bias term b are considered as the free parameters that conforms the perceptron model. Therefore, these are the parameters that are iteratively adjusted during the optimization processes such as the SGD.

2.2.10.2. Fully-Connected Layer

Also called a dense layer, a fully-connected layer is a model designed as a extended collection of perceptrons. Given a number of M perceptron models, a fully-connected layer is connected in such a way that all the M output units are connected to all the N input units from a previous layer through the associated weights. Thus, an arbitrary output value y_i, in a fully-connected layer, is defined as follows:

$$y_i = \phi\big(\boldsymbol{w}_i^T\boldsymbol{x}\big), \forall i \in \{1, \ldots, M\}, \tag{2.64}$$

or, in matrix notation:

$$\boldsymbol{y} = \phi(\boldsymbol{W}\boldsymbol{x}), \tag{2.65}$$

where $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M]^T \in \mathbb{R}^{M \times N}$ is a matrix of weights. The vector $\boldsymbol{x} \in \mathbb{R}^M$ is the input vector, $\boldsymbol{y} \in \mathbb{R}^M$ is the output vector, and ϕ is an element-wise arbitrary nonlinear function. Furthermore, it is possible to observe a direct relationship with the concept of linear projection using the matrix $\boldsymbol{W}$[19].

In addition, as in the perceptron model, usually we add extra bias terms in the expression as follows:

$$\boldsymbol{y} = \phi(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}), \tag{2.66}$$

where $\boldsymbol{b} \in \mathbb{R}^M$ is the vector of biases.

Finally, a common notation is to express a fully-connected layer in terms of a parameterized function: $\boldsymbol{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x})$, where $\boldsymbol{\theta} = [w_{1,1}, \ldots, w_{M,N}, b_1, \ldots, b_M]^T$ is the vector containing all the free parameters of the model.

[19] A fully-connected layer is often referred to as a linear projection followed by a nonlinear operation.

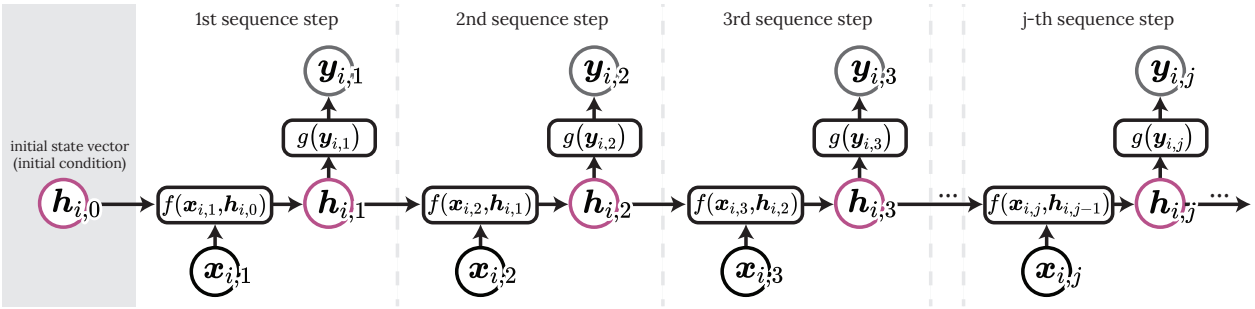

Figure 2.2: Graph diagram representing a causal sequence for an arbitrary i-th sequence instance. The vector $\boldsymbol{x}_{i,j}$ is the input vector, $\boldsymbol{h}_{i,j}$ is the state vector, and $\boldsymbol{y}_{i,j}$ is the output vector at an arbitrary sequence step j.

2.2.10.3. Multi-Layer Perceptron (MLP)

A Multi-Layer Perceptron (MLP; Rumelhart et al., 1986) model is an architecture based on a collection of fully-connected layers that are causally and sequentially connected in a stack scheme, i.e., the output of one layer is the input of the next layer. Thus, given an input vector $\boldsymbol{x} \in \mathbb{R}^N$, the output vector $\boldsymbol{y} \in \mathbb{R}^M$ is expressed as follows:

$$\boldsymbol{y} = f_{\boldsymbol{\theta}_N} \circ \cdots \circ f_{\boldsymbol{\theta}_1}(\boldsymbol{x}), \tag{2.67}$$

where $f_{\boldsymbol{\theta}_1}, \ldots, f_{\boldsymbol{\theta}_{N_L}}$ are the N different fully-connected layers used to define the MLP model.

Finally, a common notation is to express the MLP model as a parameterized function: $\boldsymbol{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x})$, where $\boldsymbol{\theta} = \text{cat}[\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{N_L}]$ is the vector containing all the free parameters of the model, which are the parameters of the N_L fully-connected layers used.

2.2.11. Recurrent Neural Networks (RNNs)

The Recurrent Neural Network (RNN; Rumelhart et al., 1986; Zimmermann et al., 2012) is a type of ANN-based architecture designed to process time series or causal sequences. In RNNs, the Markov property on a vector state $\boldsymbol{h}_{i,j}$ is used as a base assumption, i.e., the current state of the system $\boldsymbol{h}_{i,j}$ can be inferred based on its most immediate past $\boldsymbol{h}_{i,j-1}$. Then, the RNN model is described with the following relationships (Fig. 2.2 shows a graph-like illustration to represent a causal sequence):

$$\boldsymbol{h}_{i,j} = f(\boldsymbol{x}_{i,j}, \boldsymbol{h}_{i,j-1}), \tag{2.68}$$
$$\boldsymbol{y}_{i,j} = g(\boldsymbol{h}_{i,j}), \tag{2.69}$$

where $\boldsymbol{h}_{i,j} \in \mathbb{R}^{D_h}$ is the vector representing a D_h-dimensional state vector of the dynamical system. This state is modeled by a transition function $f(\cdot)$ that uses as argument both, the current D_x-dimensional input vector $\boldsymbol{x}_{i,j} \in \mathbb{R}^{D_x}$ and the previous state vector $\boldsymbol{h}_{i,j-1} \in \mathbb{R}^{D_h}$. In addition, the observed output of the system $\boldsymbol{y}$ is modeled by a function $g(\cdot)$ that uses as input the current state $\boldsymbol{h}_{i,j}$.

2.2.11.1. Vanilla RNNs

The first approach to model the functions $f(\cdot)$ and $g(\cdot)$ is to use a simple parameterization based on ANNs. In that case, the expressions (2.68)-(2.69) are rewritten as:

$$\boldsymbol{h}_{i,j} = f(\boldsymbol{x}_{i,j}, \boldsymbol{h}_{i,j-1}) = \phi\big(\boldsymbol{W}_x^T \boldsymbol{x}_{i,j} + \boldsymbol{b}_x + \boldsymbol{W}_h^T \boldsymbol{h}_{i,j-1} + \boldsymbol{b}_h\big), \tag{2.70}$$

$$\boldsymbol{y}_{i,j} = g(\boldsymbol{h}_{i,j}) = \boldsymbol{h}_{i,j}, \tag{2.71}$$

where eq. (2.70) computes the current state vector $\boldsymbol{h}_{i,j}$ as a function of the current input vector $\boldsymbol{x}_{i,j}$ and the previous state vector $\boldsymbol{h}_{i,j-1}$. For this, the weight matrices $\boldsymbol{W}_x \in \mathbb{R}^{D_x \times D_h}$ and $\boldsymbol{W}_h \in \mathbb{R}^{D_h \times D_h}$, and the bias vectors $\boldsymbol{b}_x \in \mathbb{R}^{D_h}$ and $\boldsymbol{b}_h \in \mathbb{R}^{D_h}$, are used. Additionally, an arbitrary nonlinear activation function ϕ is used to induce nonlinear interactions. Note that eq. (2.70) can also be implemented by using a fully-connected layer, where the input is the concatenation of the input vector $\boldsymbol{x}_{i,j}$ and the previous state vector $\boldsymbol{h}_{i,j-1}$: $\text{cat}[\boldsymbol{x}_{i,j}, \boldsymbol{h}_{i,j-1}]$.

Although the above formulation is theoretically correct, in practice, this model usually suffers from two main problems: the vanishing gradient and exploding gradient problems. The vanishing gradient problem consists in the fact that, after successive applications of the chain rule during the back-propagation algorithm, the gradient values may ends up converging to 0. Such a small gradient implies that the updating of the model parameters is also performed with small changes, which drastically slows down the convergence to a local minimum in the optimization of the cost function. On the other hand, the exploding gradient refers to the scenario when the values of the model parameters grow abruptly and uncontrollably during the training process, finally producing a computational overflow problem.

2.2.11.2. Long Short-Term Memory Model (LSTM)

The Long Short-Term Memory (LSTM; Hochreiter & Schmidhuber, 1997) model is a type of RNN that tries to solve the problems usually encountered in the Vanilla RNN models: the Vanishing and exploding gradients problems. Fig. 2.3 shows a diagram that describes a single cell of the LSTM model.

The main assumption of the LSTM model is the existence of a memory cell that can learn how to handle which information should be read and stored, at each sequence step, in a current internal state $\boldsymbol{h}_{i,j}$. It also could learn which information should be forgotten from $\boldsymbol{h}_{i,j}$ and which information should be displayed in the output vector $\boldsymbol{y}_{i,j}$. The equations that define the operation of an LSTM cell are the following:

$$\boldsymbol{i}_{i,j} = \sigma\big(\boldsymbol{W}_{xi}^T \boldsymbol{x}_{i,j} + \boldsymbol{b}_{xi} + \boldsymbol{W}_{hi}^T \boldsymbol{h}_{i,j-1} + \boldsymbol{b}_{hi}\big), \tag{2.72}$$

$$\boldsymbol{f}_{i,j} = \sigma\big(\boldsymbol{W}_{xf}^T \boldsymbol{x}_{i,j} + \boldsymbol{b}_{xf} + \boldsymbol{W}_{hf}^T \boldsymbol{h}_{i,j-1} + \boldsymbol{b}_{hf}\big), \tag{2.73}$$

$$\boldsymbol{o}_{i,j} = \sigma\big(\boldsymbol{W}_{xo}^T \boldsymbol{x}_{i,j} + +\boldsymbol{b}_{xo} + \boldsymbol{W}_{ho}^T \boldsymbol{h}_{i,j-1} + \boldsymbol{b}_{ho}\big), \tag{2.74}$$

$$\hat{\boldsymbol{c}}_{i,j} = \phi_{\tanh}\big(\boldsymbol{W}_{x\hat{c}}^T \boldsymbol{x}_{i,j} + \boldsymbol{b}_{x\hat{c}} + \boldsymbol{W}_{h\hat{c}}^T \boldsymbol{h}_{i,j-1} + \boldsymbol{b}_{h\hat{c}}\big), \tag{2.75}$$

$$\boldsymbol{c}_{i,j} = \boldsymbol{f}_{i,j} \odot \boldsymbol{c}_{i,j-1} + \boldsymbol{i}_{i,j} \odot \hat{\boldsymbol{c}}_{i,j}, \tag{2.76}$$

$$\boldsymbol{y}_{i,j} = \boldsymbol{h}_{i,j} = \boldsymbol{o}_{i,j} \odot \phi_{\tanh}(\boldsymbol{c}_{i,j}), \tag{2.77}$$

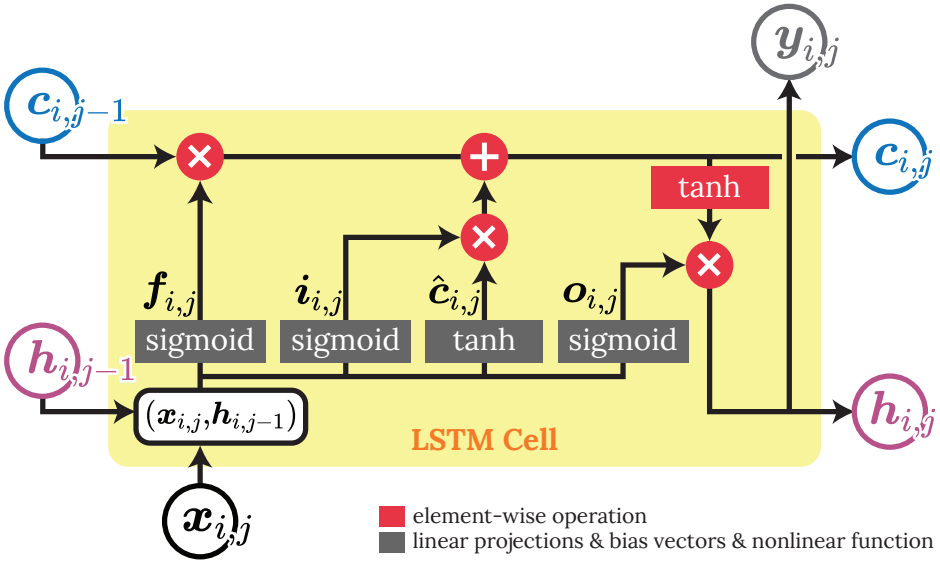

Figure 2.3: Explanatory diagram of a single LSTM cell. The operations are shown at an arbitrary sequence step j, where $\boldsymbol{x}_{i,j}$ is the input vector, $\boldsymbol{y}_{i,j}$ is the output vector, $\boldsymbol{h}_{i,j}$ is the current state vector, $\boldsymbol{h}_{i,j-1}$ is the previous state vector, $\boldsymbol{c}_{i,j}$ is the current memory vector, and $\boldsymbol{c}_{i,j-1}$ is the previous memory vector.

where ϕ_{tanh} and σ are the nonlinear hyperbolic tangent and sigmoid functions, respectively. The operator $\odot$ is the element-wise product. The detailed explanation of the LSTM model is as follows:

1. **Input gate vector**: in eq. (2.72), the input gate vector $\boldsymbol{i}_{i,j}$ is computed by using the input vector $\boldsymbol{x}_{i,j}$ and the previous state vector $\boldsymbol{h}_{i,j-1}$. Note that the values of $\boldsymbol{i}_{i,j}$ are in the range $[0, 1]$ given the sigmoid function σ. This operation is modeled by using a fully-connected layer, with the weight matrices $\boldsymbol{W}_{xi}$ and $\boldsymbol{W}_{hi}$, and the bias vectors $\boldsymbol{b}_{xi}$ and $\boldsymbol{b}_{hi}$.

2. **Output forget vector**: in eq. (2.73), the output forget vector $\boldsymbol{f}_{i,j}$ is computed by using the input vector $\boldsymbol{x}_{i,j}$ and the previous state vector $\boldsymbol{h}_{i,j-1}$. Note that the values of $\boldsymbol{i}_{i,j}$ are in the range $[0, 1]$ given the sigmoid function σ. This operation is modeled by using a fully-connected layer, with the weight matrices $\boldsymbol{W}_{xf}$ and $\boldsymbol{W}_{hf}$, and the bias vectors $\boldsymbol{b}_{xf}$ and $\boldsymbol{b}_{hf}$.

3. **Output gate vector**: in eq. (2.74), the output forget vector $\boldsymbol{o}_{i,j}$ is computed by using the input vector $\boldsymbol{x}_{i,j}$ and the previous state vector $\boldsymbol{h}_{i,j-1}$. Note that the values of $\boldsymbol{o}_{i,j}$ are in the range $[0, 1]$ given the sigmoid function σ. This operation is modeled by using a fully-connected layer, with the weight matrices $\boldsymbol{W}_{xo}$ and $\boldsymbol{W}_{ho}$, and the bias vectors $\boldsymbol{b}_{xo}$ and $\boldsymbol{b}_{ho}$.

4. **Candidate memory vector**: in eq. (2.75), the candidate memory vector $\hat{\boldsymbol{c}}_{i,j}$ is computed by using the input vector $\boldsymbol{x}_{i,j}$ and the previous state vector $\boldsymbol{h}_{i,j-1}$. Note that the values of $\hat{\boldsymbol{c}}_{i,j}$ are in the range $(-1, 1)$ given the hyperbolic tangent function ϕ_{tanh}. This operation is modeled by using a fully-connected layer, with the weight matrices $\boldsymbol{W}_{x\hat{c}}$ and $\boldsymbol{W}_{h\hat{c}}$, and the bias vectors $\boldsymbol{b}_{x\hat{c}}$ and $\boldsymbol{b}_{h\hat{c}}$.

5. **Memory vector, input, and forget gate operation**: in eq. (2.76), the memory vector $\boldsymbol{c}_{i,j}$ is computed, controlling the flow of information that is transferred from the candidate memory vector $\hat{\boldsymbol{c}}_{i,j}$ to a new memory vector $\boldsymbol{c}_{i,j}$. Given the operation $\boldsymbol{i}_{i,j} \odot \hat{\boldsymbol{c}}_{i,j}$, unwanted elements from $\hat{\boldsymbol{c}}_{i,j}$ can be attenuated or «ignored» when the values associated with the input gate $\boldsymbol{i}_{i,j}$ are close to 0. On the other hand, the operation $\boldsymbol{f}_{i,j} \odot \boldsymbol{c}_{i,j-1}$ allows the model to attenuate or «forget» unnecessary elements from the previous memory vector $\boldsymbol{c}_{i,j-1}$, when the values associated with the input gate vector $\boldsymbol{f}_{i,j}$ are close to 0.
6. **State, output vector, and output gate operation**: in eq. (2.77), the «selection» of the elements in the memory vector $\boldsymbol{c}_{i,j}$ that should be stored in the current state $\boldsymbol{h}_{i,j}$ is performed. Like the previous cases, this is performed by attenuating unwanted elements by using values close to 0 in the output gate vector $\boldsymbol{o}_{i,j}$. Note that, in the LSTM model, the external output vector $\boldsymbol{y}_{i,j}$ directly corresponds to the state vector $\boldsymbol{h}_{i,j}$.

Additionally, an important consideration is taken w.r.t. the initialization of the bias vectors in the LSTM model. The bias vector $\boldsymbol{b}_f$, associated with the forget gate, is initialized as a one vector $\vec{1}$ (instead of the usual alternative of zero vector $\vec{0}$ initialization). This consideration is taken in order to prevent that all the information is «forgotten» just in the first few iterations of the training optimization. Finally, if it is not explicitly mentioned, the memory vector $\boldsymbol{c}_{i,j}$ and the state vector $\boldsymbol{h}_{i,j}$ are initialized as zero vectors: $\boldsymbol{c}_0 = \boldsymbol{h}_0 = \vec{0}$.

2.2.11.3. Gated Recurrent Unit (GRU) Model

The Gated Recurrent Unit (GRU; Chung et al., 2014) model is a simplified alternative of the LSTM model. A direct advantage of the GRU model is that it has fewer parameters than the LSTM model. This makes the GRU model less complex, occupying less computational memory and operations. Fig. 2.4 shows a diagram that describes a single cell of the GRU model.

By following similar ideas from the LSTM model, the equations for the GRU model are the following:

$$\boldsymbol{z}_{i,j} = \sigma\big(\boldsymbol{W}_{xz}^T \boldsymbol{x}_{i,j} + \boldsymbol{b}_{xz} + \boldsymbol{W}_{hz}^T \boldsymbol{h}_{i,j-1} + \boldsymbol{b}_{hz}\big), \tag{2.78}$$

$$\boldsymbol{r}_{i,j} = \sigma\big(\boldsymbol{W}_{xr}^T \boldsymbol{x}_{i,j} + \boldsymbol{b}_{xr} + \boldsymbol{W}_{hr}^T \boldsymbol{h}_{i,j-1} + \boldsymbol{b}_{hr}\big), \tag{2.79}$$

$$\boldsymbol{g}_{i,j} = \phi_{\text{tanh}}\big(\boldsymbol{W}_{xg}^T \boldsymbol{x}_{i,j} + \boldsymbol{b}_{xg} + \boldsymbol{r}_{i,j} \odot \big(\boldsymbol{W}_{hg}^T \boldsymbol{h}_{i,j-1} + \boldsymbol{b}_{hg}\big)\big), \tag{2.80}$$

$$\boldsymbol{y}_{i,j} = \boldsymbol{h}_{i,j} = \boldsymbol{z}_{i,j} \odot \boldsymbol{h}_{i,j-1} + (1 - \boldsymbol{z}_{i,j}) \odot \boldsymbol{g}_{i,j}, \tag{2.81}$$

where ϕ_{tanh} and σ are the nonlinear hyperbolic tangent and sigmoid functions, respectively. The operator $\odot$ is the element-wise product. A detailed explanation of the GRU model is described as follows:

1. **Update gate vector**: in eq. (2.78), the update gate vector $\boldsymbol{z}_{i,j}$ is computed by using the input vector $\boldsymbol{x}_{i,j}$ and the previous state vector $\boldsymbol{h}_{i,j-1}$. Note that the values of $\boldsymbol{z}_{i,j}$ are in the range $[0, 1]$ given the sigmoid function σ. This operation is modeled by using

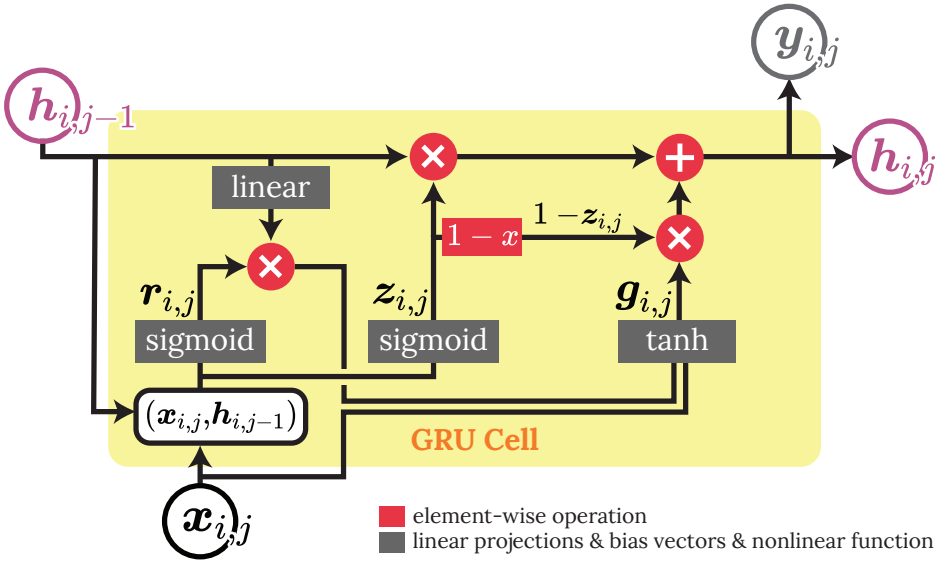

Figure 2.4: Explanatory diagram of a single GRU cell. The operations are shown at an arbitrary sequence step j, where $\boldsymbol{x}_{i,j}$ is the input vector, $\boldsymbol{y}_{i,j}$ is the output vector, $\boldsymbol{h}_{i,j}$ is the current state vector, and $\boldsymbol{h}_{i,j-1}$ is the previous state vector.

a fully-connected layer, with the weight matrices $\boldsymbol{W}_{xz}$ and $\boldsymbol{W}_{hz}$, and the bias vectors $\boldsymbol{b}_{xz}$ and $\boldsymbol{b}_{hz}$.

2. **Reset gate vector**: in eq. (2.79), the reset gate vector $\boldsymbol{r}_{i,j}$ is computed by using the input vector $\boldsymbol{x}_{i,j}$ and the previous state vector $\boldsymbol{h}_{i,j-1}$. Note that the values of $\boldsymbol{r}_{i,j}$ are in the range $[0, 1]$ given the sigmoid function σ. This operation is modeled by using a fully-connected layer, with the weight matrices $\boldsymbol{W}_{xr}$ and $\boldsymbol{W}_{hr}$, and the bias vectors $\boldsymbol{b}_{xr}$ and $\boldsymbol{b}_{hr}$.

3. **Candidate memory vector and update gate operation**: in eq. (2.80), the candidate memory vector $\boldsymbol{g}_{i,j}$ is computed by using the input vector $\boldsymbol{x}_{i,j}$ and the previous state vector $\boldsymbol{h}_{i,j-1}$. Note that the values of $\boldsymbol{g}_{i,j}$ are in the range $(-1, 1)$ given the hyperbolic tangent function ϕ_{tanh}. This operation is modeled by using a fully-connected layer, with the weight matrices $\boldsymbol{W}_{xg}$ and $\boldsymbol{W}_{hg}$, and the bias vectors $\boldsymbol{b}_{xg}$ and $\boldsymbol{b}_{hg}$. Similar to the input gate in the LSTM model, the reset vector $\boldsymbol{r}_{i,j}$ controls which elements from the state vector $\boldsymbol{h}_{i,j-1}$ should be attenuated or «discarded» when constructing the candidate memory vector $\boldsymbol{g}_{i,j}$.

4. **State, output vector, and reset gate operation**: in eq. (2.81), the current state vector $\boldsymbol{h}_{i,j}$ is computed. Two main operations can be observed. First, the operation $\boldsymbol{z}_{i,j} \odot \boldsymbol{h}_{i,j-1}$ aims to attenuate or «forget» irrelevant values from the previous state vector $\boldsymbol{h}_{i,j-1}$ by using the update vector $\boldsymbol{z}_{i,j}$. Second, the operation $(1 - \boldsymbol{z}_{i,j}) \odot \boldsymbol{g}_{i,j}$ aims to attenuate or «ignore» unwanted values from the candidate memory vector $\boldsymbol{g}_{i,j}$ by using the vector $(1 - \boldsymbol{z}_{i,j})$. Given an arbitrary k-th component of the update gate vector $\boldsymbol{z}_{i,j}$, then the update dynamic for the state vector $\boldsymbol{h}_{i,j}$ is performed as follows:

   (a) When the value $z_{i,j,k}$ is close to 1, then the value $h_{i,j-1,k}$, from the previous state vector $\boldsymbol{h}_{i,j-1}$, are completely used in the updated of the new state vector $\boldsymbol{h}_{i,j}$.

(b) At the same time, this induces a value close to 0 for the value $(1 - z_{i,j,k})$, which produces that the value $g_{i,j,k}$, from the candidate memory vector $\boldsymbol{g}_{i,j}$, is completely ignored in the update of the new state vector $\boldsymbol{h}_{i,j}$.

Note that the inverse effect occurs if a value in the update gate vector $\boldsymbol{z}_{i,j}$ is close to 0. Also, notice that, in contrast to the LSTM model, in the GRU model there is no output gate vector $\boldsymbol{o}_{i,j}$ that exclusively controls the elements stored in the state vector $\boldsymbol{h}_{i,j}$. Finally, the output $\boldsymbol{y}_{i,j}$ is directly the state vector of the model: $\boldsymbol{h}_{i,j}$.

From the above explanation, it can be observed how the GRU model simultaneously simplifies and unifies the operations, previously seen in the LSTM model, of «forget» and «input» into a single new operation «reset». It should be highlighted that although the GRU model is a simplified version of the LSTM model, in practice, it is usual to achieve competitive performance between in both models. Therefore, it is a good idea to experiment with both models and to compare the results empirically. Finally, if it is not explicitly mentioned, the state vector $\boldsymbol{h}_{i,j}$ is initialized as a zero vector: $\boldsymbol{h}_0 = \vec{0}$.

2.2.12. Multi-Head Self-Attention Mechanisms

The attention mechanisms were initially conceived as strategies to support the processing of word sequences (tokens) in the Natural Language Processing (NLP) research field. One of the first architectures to include an attention mechanism was proposed as an alternative to improve the performance of an LSTM-RNN model (Hochreiter & Schmidhuber, 1997) for the language translation task (Bahdanau et al., 2014). In general, this composite architecture design, based on RNNs and supported by attention mechanisms, has been vastly used in the NLP research field for several years.

Later on, the idea of implementing Deep Learning models based solely on attention mechanisms was introduced by the Transformer model (Vaswani et al., 2017). Along with this model, an increasing interest in the use of attention mechanisms has been developed over the last few years along with notable examples, such as the BERT model (Devlin et al., 2019) or the GTP model (Radford et al., 2019) for different NLP tasks. Recently, this interest has also spread among other research fields outside NLP, with attention-based models used in general multi-variate time-series classification (Lin et al., 2020), healthcare and clinical time-series processing (Horn et al., 2019; Lee et al., 2021; Shukla & Marlin, 2021), financial time-series (Kim & Kang, 2019), or simulated photometric transient light-curves classification (Ibsen & Mann, 2020; Allam & McEwen, 2021).

2.2.12.1. Multi-Head Dot-Attention

In this section, the multi-head dot-attention mechanism is described, which is the principal mechanism of the Transformer model (Vaswani et al., 2017). The attention mechanisms allows us to process and extract, hopefully useful, information from a collection of an arbitrary number of vectors. Moreover, this capability can be extended to process a causal sequence of

vectors, e.g., the photometric information from an astronomical event expressed as a variable-length light-curve.

The definition of the multi-head dot-attention mechanism is based on concepts such as the query, key, value, and context vectors. An intuitive understanding of the attention mechanism can be grasped as an user search operation over a collection of contents. First, a search operation (query) is performed over a set of content descriptions (keys), each one related with a different content (values), matching the current query against each one of the keys. Then, the search system can «pay high attention to» and retrieve all the best-matched values according to the best-matched associated keys given the current query. Additionally, a summarized description (context) can be constructed by summarizing all the common and relevant information among all the best-matched retrieved values. Note that the above operation (single-head attention mechanism) could be performed multiple times independently (multi-head attention mechanism), where, depending on the user search query, different contexts can be constructed.

Let's assume a sequence set of arbitrary input vectors $\{\boldsymbol{x}_{i,1}, \ldots, \boldsymbol{x}_{i,L}\} = \{\boldsymbol{x}_{i,j'}\}_{j'=1}^{L}$ (associated with the key and value vectors) and an arbitrary input vector $\boldsymbol{x}_{i,j}$ (associated with the query vector), where the sequence of vectors is analogous to an arbitrary light-curve Φ_i. As it is fully detailed in section §3.4, each input vector $\boldsymbol{x}_{i,j}$ is a high-dimensional representation that is expected to automatically contain information about the current observation, such as the observation-flux $\mu_{i,j}$, the band indicator $b_{i,j}$, and the observation-time $t_{i,j}$ (days).

The equations that describe the multi-head dot-attention mechanism are the following[20]:

$$a_{i,j'}^{(h)} = \frac{1}{\sqrt{D_k}} \underbrace{\left(\boldsymbol{W}_k^{(h)^T} \boldsymbol{x}_{i,j'} + \boldsymbol{b}_k^{(h)}\right)^T}_{\text{key: } \boldsymbol{k}_{i,j'}^{(h)}} \underbrace{\left(\boldsymbol{W}_q^{(h)^T} \boldsymbol{x}_{i,j} + \boldsymbol{b}_q^{(h)}\right)}_{\text{query: } \boldsymbol{q}_{i,j}^{(h)}}, \quad \text{Aligment values} \tag{2.82}$$

$$s_{i,j'}^{(h)} = \frac{\exp\left(a_{i,j'}^{(h)}\right)}{\sum_{j'=1}^{L} \exp\left(a_{i,j'}^{(h)}\right)}, \quad \text{Attention score} \tag{2.83}$$

$$\boldsymbol{c}_{i,j}^{(h)} = \sum_{j'=1}^{L} s_{i,j'}^{(h)} \cdot \underbrace{\left(\boldsymbol{W}_v^{(h)^T} \boldsymbol{x}_{i,j'} + \boldsymbol{b}_v^{(h)}\right)}_{\text{value: } \boldsymbol{v}_{i,j'}^{(h)}}, \quad \text{Context vector} \tag{2.84}$$

$$\boldsymbol{c}_{i,j} = \boldsymbol{W}_c^T \left(\text{cat} \left[\underbrace{\boldsymbol{c}_{i,j}^{(1)}, \ldots, \boldsymbol{c}_{i,j}^{(H)}}_{\text{head contexts}} \right] \right) + \boldsymbol{b}_c, \quad \text{Multi-head context vector} \tag{2.85}$$

where eqs. (2.82)-(2.84) follow the example scheme shown in Fig. 2.5: given a query vector $\boldsymbol{q}_{i,j}^{(h)}$, the goal of the attention mechanism is to compute a context vector $\boldsymbol{c}_{i,j}^{(h)}$ based on the set of key vectors $\left\{\boldsymbol{k}_{i,j'}^{(h)}\right\}_{j'=1}^{L}$ and the set of value vectors $\left\{\boldsymbol{v}_{i,j'}^{(h)}\right\}_{j'=1}^{L}$. Note that the computation of the context vector can be performed independently and in parallel for each attention head $h \in \{1, \ldots, H\}$, where H is an arbitrary number of context vector computations: the number

[20] According to the implementation in `https://pytorch.org/docs/stable/generated/torch.nn.MultiheadAttention.html`. For extra details, see the original Transformer model in Vaswani et al. (2017).

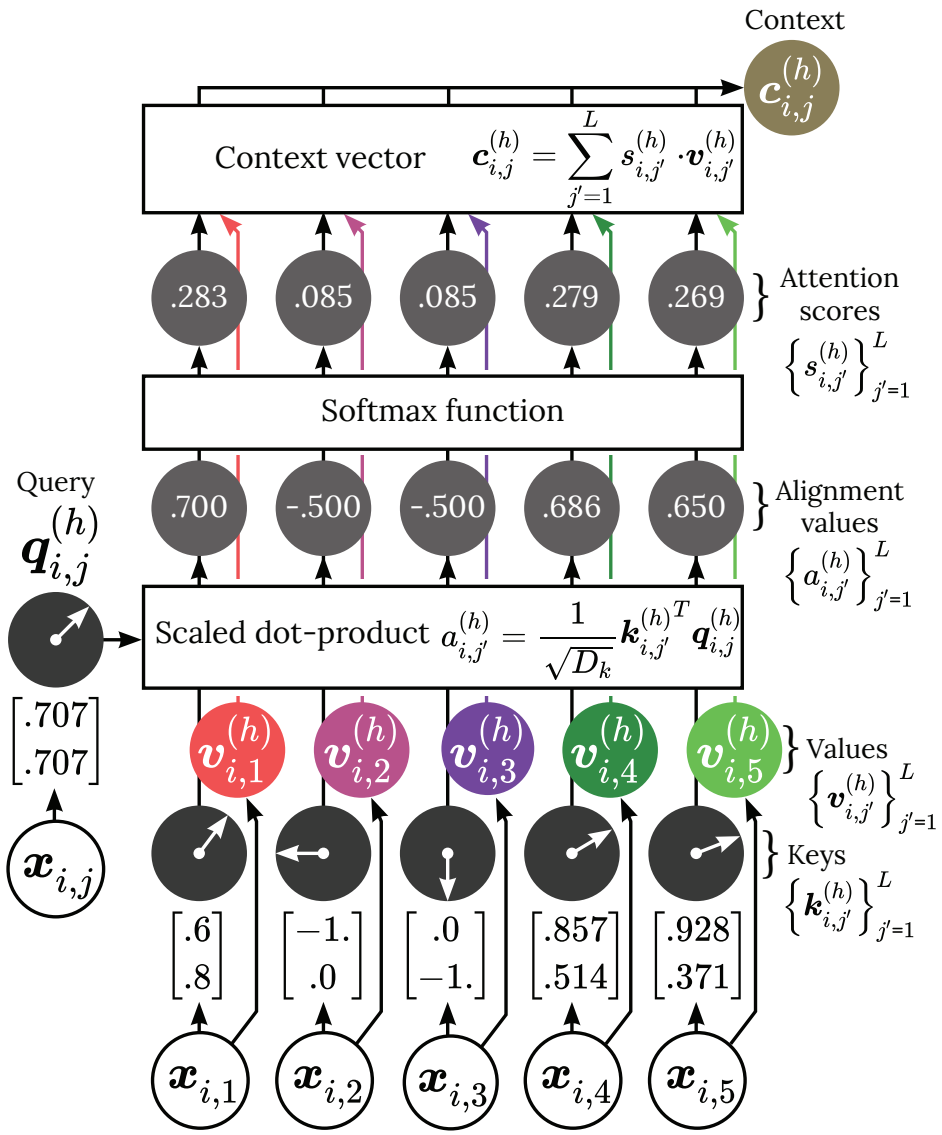

Figure 2.5: A simplified example of a dot-attention mechanism given an arbitrary attention head h. In order to better illustrate the alignments, the query and key vectors are represented as pointing arrow 2D vectors (e.g., $\boldsymbol{q}_{i,j}^{(h)} = [.707, .707]^T$). Those 2D vectors (and the value vectors) are the projected vectors, from the input vectors, obtained by using the associated learnable linear projections and bias vectors for the attention head h. The attention scores are based on the scaled dot-product between the query and key vectors. The value vectors are represented as color codes, explaining the final color used for the context vector $\boldsymbol{c}_{i,j}^{(h)}$. This operation can be extended and parallelized for an arbitrary number of H multiple attention heads.

of attention heads (denoting h as an arbitrary attention head). Therefore, in eq. (2.85), the final context vector $\boldsymbol{c}_{i,j}$ is computed by using the information from all the H parallel attention heads (the operator cat[...] stands for the concatenation operator for vectors). A detailed explanation of the multi-head dot-attention mechanism is as follows:

1. **Alignment values**: given an arbitrary attention head h, in eq. (2.82), the alignment

values $\left\{a_{i,j'}^{(h)}\right\}_{j'=1}^{L}$ are computed as the dot product between a query vector $\boldsymbol{q}_{i,j}^{(h)}$ and the key vectors in the set $\left\{\boldsymbol{k}_{i,j'}^{(h)}\right\}_{j'=1}^{L}$. Both, the query and key vectors, are projected from an input vector $\boldsymbol{x}_{i,j}$ and a set of input vectors $\{\boldsymbol{x}_{i,j'}\}_{j'=1}^{L}$, respectively. A learnable linear projection $\boldsymbol{W}_q^{(h)} \in \mathbb{R}^{D\times D_q}$, and a bias vector $\boldsymbol{b}_q$, are used for the query vector; and a learnable linear projection $\boldsymbol{W}_k^{(h)} \in \mathbb{R}^{D\times D_k}$, and a bias vector $\boldsymbol{b}_k$, are used for the key vectors. The dimensions D, D_q, and D_k are associated with the input, query, and key vectors, respectively[21]. The alignment values are computed using a scaled dot-product operation between the query and key vectors as $\frac{1}{\sqrt{D_k}}\boldsymbol{k}_{i,j'}^T\boldsymbol{q}_{i,j}$ (scaled dot-attention mechanism). This operation represents an estimation of the linear correlation between the query and key vectors, where a high correlation implies a high alignment value.

2. **Attention scores**: given an arbitrary attention head h, in eq. (2.83), the attention scores $\left\{s_{i,j'}^{(h)}\right\}_{j'=1}^{L}$ are computed using the alignment values $\left\{a_{i,j'}^{(h)}\right\}_{j'=1}^{L}$. This operation is performed using the softmax function over the set of alignment values, constructing a discrete distribution, where the following condition holds: $\sum_{j'=1}^{L} s_{i,j'}^{(h)} = 1, s_{i,j'}^{(h)} \in (0,1), \forall h \in \{1,\ldots,H\}$. Note that the best-matched key vectors, w.r.t. the query vector, achieve higher attention scores.

3. **Context vector**: given an arbitrary attention head h, in eq. (2.84) the context vector $\boldsymbol{c}_{i,j}^{(h)}$ is computed as the vector aggregation from the value vectors $\left\{\boldsymbol{v}_{i,j'}^{(h)}\right\}_{j'=1}^{L}$ weighted by the attention scores $\left\{s_{i,j'}^{(h)}\right\}_{j'=1}^{L}$. These value vectors are projected from the same input vectors used for the key values, through the learnable linear projection $\boldsymbol{W}_v^{(h)} \in \mathbb{R}^{D\times D_v}$ and a bias vector $\boldsymbol{b}_v$, where $D_v = D/H$ is the value vector dimension. Recalling that the attention scores represent a discrete distribution, this operation is like the estimation of the vector expectation over the set of value vectors. Finally, if a given query $\boldsymbol{q}_{i,j}^{(h)}$ gets a high alignment with the key $\boldsymbol{k}_{i,j'}^{(h)}$; then, the associated vector $\boldsymbol{v}_{i,j'}^{(h)}$ will be weighted higher in the resulting context vector $\boldsymbol{c}_{i,j}^{(h)}$.

4. **Multi-head context vector**: one of the novel architecture ideas introduced by the Transformer model is the multi-head attention capability. This configuration allows the model to distribute the attention computation, as described above, among several heads running independently and in parallel. This usually helps to increase the model performance, as each head can specialize in different tasks, paying attention to different patterns along the input sequence. Therefore, in eq. (2.85), a final context vector $\boldsymbol{c}_{i,j}$ is projected from the concatenation of all the H parallel attention heads context vectors $\left\{\boldsymbol{c}_{i,j}^{(1)},\ldots,\boldsymbol{c}_{i,j}^{(H)}\right\}$. This final context is computed through the learnable linear projection $\boldsymbol{W}_c \in \mathbb{R}^{(H\cdot D_v)\times D}$ and a bias vector $\boldsymbol{b}_c$. Finally, this operation allows the model to capture relevant information from all the H parallel context vectors $\boldsymbol{c}_{i,j}^{(h)}$.

[21]The dimensions D_q and D_k are selected as $D_q = D/H$ and $D_k = D/H$, respectively (Vaswani et al., 2017). Note that the dimensions for the query and key values must be the same in order to properly compute the dot-product between both vectors.

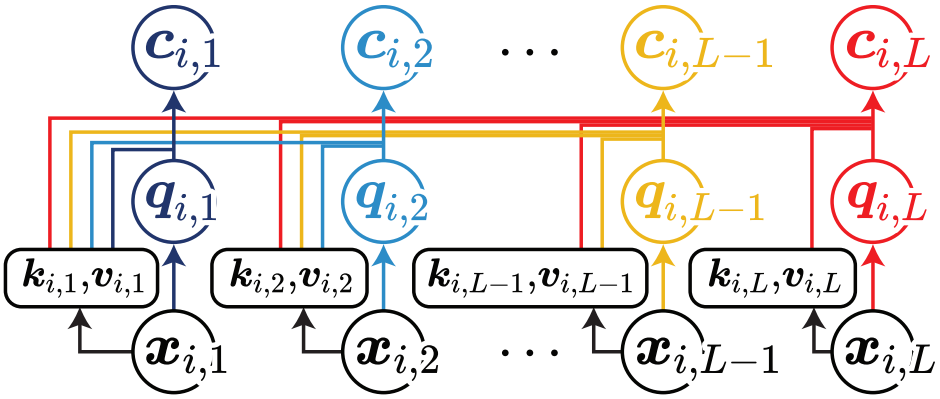

Figure 2.6: Self-attention example diagram. The key $\boldsymbol{k}_{i,j}$, value $\boldsymbol{v}_{i,j}$, and query $\boldsymbol{q}_{i,j}$ vector relationships are shown for the 1st, 2nd, $(L-1)$-th, and L-th sequence steps. For simplicity, we assume a single attention head mechanism and omit the attention head superscript $(\cdot)^{(h)}$. This operation is causal as each context vector $\boldsymbol{c}_{i,j}$ computation only depends on the current and previous sequence steps. All query, key, and value vectors come from the same sequence.

2.2.12.2. Multi-Head Self-Attention

A self-attention scenario arises when the query vectors are projected from the same sequence domain as the key and value vectors. Given a sequence of input vectors $\{\boldsymbol{x}_{i,j'}\}_{j'=1}^{L}$, the context vector $\boldsymbol{c}_{i,j}$ (at an arbitrary sequence step j) is computed using query vectors also projected from the current sequence step j, with key and value vectors projected from the current and previous sequence steps: $\{1,\ldots,j\}$. An example diagram for this operation is shown in Fig. 2.6, where the causality of the self-attention mechanisms can be observed.

This configuration directly produces a causal context vector computation for each intermediate sequence step j, obtaining a new context vector sequence from the original input vector sequence, where both share the same variable-length L. This process reminds the causal structure in the output of the RNN sequential processing: an output sequence where each memory-vector computation strictly depends on the current and previous memory-vectors.

2.2.12.3. Additional Operations

Additional Residual Connections (where an arbitrary function $f(\boldsymbol{x}_{i,j})$ is added to the current function input $\boldsymbol{x}_{i,j}$), plus a nonlinear operation, are included in the Transformer model formulation as follows[22]:

$$\boldsymbol{c}'_{i,j} = \underbrace{\boldsymbol{c}_{i,j}}_{\text{MHSelfAttn: } f(\boldsymbol{x}_{i,j})} + \boldsymbol{x}_{i,j}, \tag{2.86}$$

$$\boldsymbol{c}''_{i,j} = \underbrace{\boldsymbol{W}_2^T\big(\phi_{\text{ReLU}}\big(\boldsymbol{W}_1^T\boldsymbol{c}'_{i,j} + \boldsymbol{b}_1\big)\big) + \boldsymbol{b}_2}_{\text{MLP: } f(\boldsymbol{c}'_{i,j})} + \boldsymbol{c}'_{i,j}, \tag{2.87}$$

[22]For simplicity of the explanation, we omit the Layer Normalization (Ba et al., 2016) used in the original Transformer model.

where a 1-hidden-layer Multi-Layer Perceptron model (MLP; Rumelhart et al., 1986) is used to induce nonlinear interactions among the context vectors from all heads. The terms $\boldsymbol{W}_1 \in \mathbb{R}^{D\times(k_{\text{mlp}}\cdot D)}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{(k_{\text{mlp}}\cdot D)\times D}$ are the MLP linear projections and bias vectors $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$, where $k_{\text{mlp}} = 2$ controls the number of the MLP hidden units used: $k_{\text{mlp}} \cdot D$. The expression ϕ_{ReLU} stands for the Rectified Linear Unit function (ReLU).

Recalling that self-attention can compute a new causal sequence of context vectors, this operation can be stacked into a multi-layer sequence processing architecture using an arbitrary number of N_L layers. For simplicity, in the following sections of this thesis, we use the term Multi-Head Self-Attention (MHSelfAttn) for the aforementioned and extended formulation, calling the processed vector $\boldsymbol{c}''_{i,j}$ for the context vector $\boldsymbol{c}_{i,j}$.

2.2.12.4. Extra Properties and Limitations

Some extra properties of the attention mechanisms are the following:

1. **Long sequences**: empirically, attention-based models have shown higher performance than the RNN models for long sequence processing in NLP tasks (Vaswani et al., 2017). For RNN models, the maximum length between long-term dependencies (maximum path length) results in $\mathcal{O}(L)$ as the maximum path connection requires to pass over the complete sequence length. In contrast, for attention mechanisms, this maximum path length is $\mathcal{O}(1)$ given the existence of a direct shortcut path between each context vector and each value vector, connecting the context with the entire sequence at a constant cost. These direct paths facilitate the learning of long-term sequence dependencies (Hochreiter et al., 2001).

2. **Parallelizable computation**: attention mechanisms have a complexity, per layer, of $\mathcal{O}(L^2 \cdot D)$ operations, while RNN models have a complexity of $\mathcal{O}(L \cdot D^2)$ operations. This implies that attention mechanisms have lower computational complexity than the RNN models when L is lower than the embedding dimension D (Vaswani et al., 2017). However, the RNN models require a number of $\mathcal{O}(L)$ strictly non-parallelizable sequential operations as each memory-vector computation requires the previously computed memory-vector. In contrast, for attention mechanisms, all context vectors for both, the entire sequence and H parallel heads, can be computed simultaneously in a highly parallelizable operation which is optimal for the GPU usage (constant number of $\mathcal{O}(1)$ non-sequential operations).

3. **Attention masks**: the variable-length and explicit causality of the self-attention formulation are directly implemented with attention score masks over the alignment values. A negative infinite value $-\infty$ can be used to replace the alignments values over invalid sequence steps, where null attention is required (zero attention score). Optionally, null attention scores can be randomly imputed in the attention mask, during the model training process, as a sequence dropout regularization technique (Vaswani et al., 2017).

4. **Positional encoding**: one of the main limitations of the attention mechanisms is the loss of the sequential information, i.e., the explicit information of which sequence step

Table 2.1: Comparison between different approaches for the processing of sequences (e.g., time series, light-curves): Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), Temporal CNNs (tCNNs), and the multi-head dot-attention mechanism (Vaswani et al., 2017). We denote D as the dimensionality of the model embedding, L as the variable-length of a processed sequence, and k as the kernel size of convolutions. For all models, a causal configuration is assumed.

Feature	RNNs (e.g., LSTM, GRU)	CNNs	TCNNs	Multi-head self-attention
Maximum path length	Complete: a cost of $\mathcal{O}(L)$ is required as the model requires to pass over the complete sequence	Medium: an stack of $\mathcal{O}(L/k)$ convolutions are required to reach the furthest sequence step	Medium: an stack of $\mathcal{O}(\log_k(L))$ convolutions are required to reach the furthest sequence step (less than those required for CNNs)	Direct: a cost of $\mathcal{O}(1)$ is required as a direct path exists to each sequence step
Complexity per layer	$\mathcal{O}(L \cdot D^2)$	$\mathcal{O}(k \cdot L \cdot D^2)$	$\mathcal{O}(k \cdot L \cdot D^2)$	$\mathcal{O}(L^2 \cdot D)$
Number of sequential non-parallelizable operations	$\mathcal{O}(L)$: RNNs have a strictly sequential optimization process. This may result in a computational cost bottleneck	$\mathcal{O}(1)$: it is highly parallelizable and optimal for GPU usage	same as CNNs	$\mathcal{O}(1)$: it is highly parallelizable and optimal for GPU usage
The variable length handling	Variable length can be directly handled by performing a recurrent graph unrolling for RNNs	The standard implementation is not designed to directly handle variable length input: other strategies, after the processing of CNNs, must be used, e.g., pooling operations	Same as CNNs	The variable length can be directly handled by inducing null attention in the attention score masks
Interpretability	The flow of the information in the memory vector of RNN could be really hard to interpret, especially in experiments that uses real data	Even if interpretability experiments can be performed based on the learned convolution kernels, this is not direct for convolution over time series data	Same as CNNs	Different and explicit interpretability experiments can be designed by exploring the attention scores that can help us to understand the model

comes before or after another sequence step. This is a direct consequence of the value vector aggregation using the attention scores: an operation that is invariant to the order of the vectors. As a solution, the original Transformer model proposed the use of a positional encoding vector, which is a collection of fixed sinusoidal waves that aims to preserve the sequential information for models based solely on attention mechanisms.

Additionally, table 2.1 shows a comparison between the self-attention mechanism w.r.t. RNNs and other sequential processing models.

2.2.13. Sequence Encoding

2.2.13.1. Positional Encoding

The Positional Encoding (PE; Vaswani et al., 2017) is a simple technique that aims to encode and preserve the positional information within an ordered sequence. This encoding was used as an ingenious and simple strategy to deal with the major difficulty in architectures

based solely on attention mechanisms: the loss of the sequential information in a sequence of objects (such as sequence of tokens in NLP).

Given an arbitrary integer sequence step $j \in \mathbb{N}$, then the associated positional encoding vector $\boldsymbol{z}_j \in \mathbb{R}^K$ is computed as follows:

$$\boldsymbol{z}_j = f_{\text{PE}}(j) = \begin{bmatrix} \mathcal{F}_1(\omega_1 j) \\ \vdots \\ \mathcal{F}_K(\omega_K j) \end{bmatrix}, \tag{2.88}$$

$$\omega_k = \frac{2\pi}{T_k}, \tag{2.89}$$

where $\mathcal{F}_k$ is a periodic function and T_k is the period of that function. The underlying idea is to construct an encoding vector where the k-th component of the vector is calculated by evaluating a periodic function. For this, it is required to define two main design settings: the periodic functions $\{\mathcal{F}_k\}_{k=1}^K$ and the different periods $\{T_k\}_{k=1}^K$.

2.2.13.2. Juxtaposed Positional Encoding

The juxtaposed Positional Encoding, which is the encoding method that is finally used in the Transformer model (PE; Vaswani et al., 2017), is built by using the following setting:

$$\mathcal{F}_k(\cdot) = \begin{cases} \sin(\cdot), & \text{if } k \text{ is an odd number} \\ \cos(\cdot), & \text{if } k \text{ is an even number} \end{cases}, \tag{2.90}$$

$$\omega_k = \begin{cases} \omega_k, & \text{if } k \text{ is an odd number} \\ \omega_{k-1}, & \text{if } k \text{ is an even number} \end{cases}, \tag{2.91}$$

therefore, the final expression of the juxtaposed Positional Encoding vector is as follows:

$$\boldsymbol{z}_j = \begin{bmatrix} \mathcal{F}_1(\cdot) \\ \mathcal{F}_2(\cdot) \\ \vdots \\ \mathcal{F}_{K-1}(\cdot) \\ \mathcal{F}_K(\cdot) \end{bmatrix} = \begin{bmatrix} \sin(\omega_1 j) \\ \cos(\omega_1 j) \\ \vdots \\ \sin(\omega_{K/2} j) \\ \cos(\omega_{K/2} j) \end{bmatrix}, \tag{2.92}$$

Note that the resulting vector $\boldsymbol{z}_j$ maintains the number dimensions K, but, given the chosen setting of interleaving sine and cosine functions, as well as the repetition of some periods, it is then only necessary to choose $K/2$ different periods to describe this type of encoding vector.

2.2.13.3. Temporal Encoding

The Temporal Encoding (TE; Horn et al., 2019; Takase & Okazaki, 2019; Sousa et al., 2020) is a simple technique for generating time descriptors in the form of dense vectors, often

presented as an extension of the Positional Encoding. These encoding vectors are expected to be a more suitable representation of time for the ANN-based models.

Given an arbitrary time value $t_j \in \mathbb{R}$, associated with a sequence step j, the temporal encoding vector $\boldsymbol{z}_j \in \mathbb{R}^K$ is computed as follows:

$$\boldsymbol{z}_j = f_{\text{TE}}(t_j) = \begin{bmatrix} \mathcal{F}_1(\omega_1 t_j + \phi_1) \\ \vdots \\ \mathcal{F}_K(\omega_K t_j + \phi_K) \end{bmatrix}, \tag{2.93}$$

$$\omega_k = \frac{2\pi}{T_k}, \tag{2.94}$$

where $\mathcal{F}_k$ is a periodic function, T_k is the period of such function, and ϕ_k is the associated phase. As in the Positional Encoding, it is required to define the periodic functions $\{\mathcal{F}_k\}_{k=1}^K$, the different periods $\{T_k\}_{k=1}^K$, and phases $\{\phi_k\}_{k=1}^K$, to be used.

2.2.13.4. Juxtaposed Temporal Encoding

The juxtaposed Temporal Encoding is build by using the following setting:

$$\mathcal{F}_k(\cdot) = \begin{cases} \sin(\cdot), & \text{if } k \text{ is an odd number} \\ \cos(\cdot), & \text{if } k \text{ is an even number} \end{cases}, \tag{2.95}$$

$$\{\omega_k, \phi_k\} = \begin{cases} \{\omega_k, \phi_k\}, & \text{if } k \text{ is an odd number} \\ \{\omega_{k-1}, \phi_{k-1}\}, & \text{if } k \text{ is an even number} \end{cases}, \tag{2.96}$$

therefore, the final expression for the juxtaposed Temporal Encoding is defined as follows:

$$\boldsymbol{z}_j = \begin{bmatrix} \mathcal{F}_1(\cdot) \\ \mathcal{F}_2(\cdot) \\ \vdots \\ \mathcal{F}_{K-1}(\cdot) \\ \mathcal{F}_K(\cdot) \end{bmatrix} = \begin{bmatrix} \sin\left(\omega_1 t_j + \phi_1\right) \\ \cos\left(\omega_1 t_j + \phi_1\right) \\ \vdots \\ \sin\left(\omega_{K/2} t_j + \phi_{K/2}\right) \\ \cos\left(\omega_{K/2} t_j + \phi_{K/2}\right) \end{bmatrix}, \tag{2.97}$$

where, as in the juxtaposed Positional Encoding, the resulting encoding vector $\boldsymbol{z}_j$ still maintains a number of K dimensions, but it is only required to choose a number of $K/2$ different periods and phases.

2.2.13.5. Time2Vec Encoding

For the sake of completeness, the Time2Vec encoding is also described in this theoretical background. The Time2Vec (Kazemi et al., 2019) encoding is a simple variant of the temporal encoding that introduces an extra flexibility by allowing the values of periods and phases to

be learned. The Time2Vec encoding $\boldsymbol{z}_j \in \mathbb{R}^K$ is defined as follows:

$$\boldsymbol{z}_j = \begin{bmatrix} \mathcal{F}_1(\omega_1 t_j + \phi_1) \\ \mathcal{F}_2(\omega_2 t_j + \phi_2) \\ \vdots \\ \mathcal{F}_K(\omega_K t_j + \phi_K) \end{bmatrix} = \begin{bmatrix} \omega_1 t_j + \phi_1 \\ \mathcal{F}_2(\omega_2 t_j + \phi_2) \\ \vdots \\ \mathcal{F}_K(\omega_K t_j + \phi_K) \end{bmatrix}, \tag{2.98}$$

where the periods $\{T_k\}_{k=1}^K$ and phases $\{\phi_k\}_{k=1}^K$ are free learnable parameters. Therefore, the main idea of this formulation is to allow the model to learn an specific encoding via back-propagation, learning the optimal values of the periods and phases. As the periodic function, the sine function is directly used: $\mathcal{F}_k(\cdot) = \sin(\cdot), \forall k \in \{2, \ldots, K\}$. Additionally, note that the encoding associated with the 1st component of the encoding vector ($\mathcal{F}_1$) is defined as a non-periodic linear expression: $z_{j,1} = \omega_1 t_j + \phi_1$.

2.2.14. Fourier Decomposition

The Fourier Decomposition is a mathematical tool for decomposing a given function or signal into its basic components. With this tool, a function can be broken down into different levels by using a collection of sine and cosine functions with different frequencies. In consequence, an arbitrary function can be build, by assuming a Fourier Decomposition framework, as an aggregation of basic periodic components. Given an arbitrary time defined function $f(t)$, the Fourier decomposition is described as follows:

$$f(t) = \sum_{m=0}^{M} a'_m \sin\left(\frac{2\pi m}{T_{\max}} t\right) + b'_m \cos\left(\frac{2\pi m}{T_{\max}} t\right), \tag{2.99}$$

$$= b'_0 + \sum_{m=1}^{M} a'_m \sin\left(\frac{2\pi m}{T_{\max}} t\right) + b'_m \cos\left(\frac{2\pi m}{T_{\max}} t\right), \tag{2.100}$$

where the time-function $f(t)$ is decomposed into M periodic components plus a constant component b'_0 (induced by $m = 0$). We denote this type of decomposition as a Fourier decomposition with a number of M harmonics components. The term $T_{\max}$ is the maximum period value selected for the Fourier decomposition. Fig. 2.7 illustrates a Fourier decomposition for an arbitrary signal, where several harmonics, for sine and cosine functions, are shown.

Note that a limitation of this method[23] is that the resulting time-function $f(t)$ is only non-redundant in the time range $[0, T_{\max}]$. This means that the same function output could be found in the time ranges $[T_{\max}, 2T_{\max}]$, $[2T_{\max}, 3T_{\max}]$, etc. To alleviate this effect, and depending on the signal, a large enough value of $T_{\max}$ could be selected.

[23]Note that this limitation also applies for the sequence encoding methods, such as the Positional and Temporal Encodings.

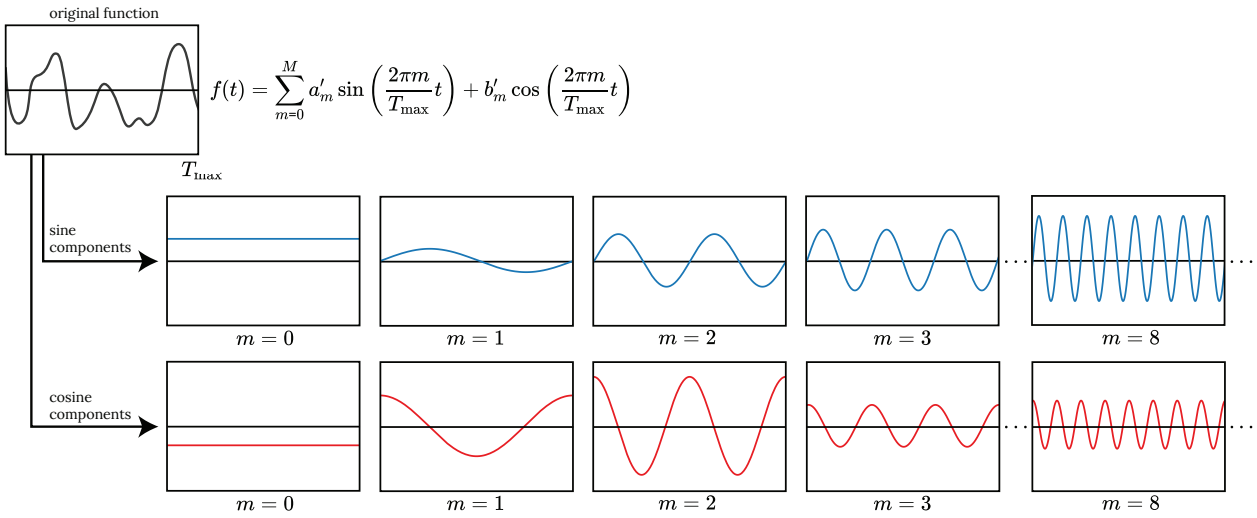

Figure 2.7: Example of an arbitrary Fourier decomposition operation. Several harmonics for the sine and cosine functions are shown.

2.2.15. Feature-wise Linear Modulation (FiLM)

The Feature-wise Linear Modulation (FiLM; Perez et al., 2018) consists of an operation that aims to perform a modulation over an arbitrary vector $\boldsymbol{x}_{i,j}$ by using another, related or non-related, arbitrary external vector $\boldsymbol{z}_{i,j}$. Usually, this operation is used along with ANN-based models. The FiLM operation is defined as follows:

$$\tilde{\boldsymbol{x}}_{i,j} = f_{\boldsymbol{\alpha}\boldsymbol{\beta}}(\boldsymbol{x}_{i,j}, \boldsymbol{z}_{i,j}) = \phi_{\tanh}(\boldsymbol{\alpha}(\boldsymbol{z}_{i,j})) \odot \boldsymbol{x}_{i,j} + \boldsymbol{\beta}(\boldsymbol{z}_{i,j}), \tag{2.101}$$

where the vector $\tilde{\boldsymbol{x}}_{i,j}$ is the output vector that is «modulated» by the vectors $\boldsymbol{\alpha}(\boldsymbol{z}_{i,j})$ and $\boldsymbol{\beta}(\boldsymbol{z}_{i,j})$, which are called the scale and bias vectors, respectively. This modulation operation is defined as the element-wise product ($\odot$) followed by the element-wise addition ($+$) (FiLM operation). Note that the scale and bias vectors are explicitly related with the vector $\boldsymbol{z}_{i,j}$. For instance, a fully-connected layer can be used to project the vector $\boldsymbol{z}_{i,j}$ into $\boldsymbol{\alpha}(\boldsymbol{z}_{i,j})$ and $\boldsymbol{\beta}(\boldsymbol{z}_{i,j})$. Additionally, an optional hyperbolic tangent function $\phi_{\tanh}$ can be used to prevent explosive product values.

2.2.16. Deep Learning Regularization and Training Techniques

As previously explored, a high complexity and flexibility can be achieved with Deep Learning models, which is often accompanied with a notable increment in the number of free adjustable/learnable parameters. Usually, this can also increase the risk of undesired effects and problems, such as the overfitting. Overfitting occurs when a complex model is able to adapt almost perfectly to a training dataset, but, at the same time, it is unable to generalize its predictions to new examples never seen before. Therefore, in the literature, multiple techniques and methods have been proposed to mitigate the risk of overfitting when optimizing Deep Learning models.

2.2.16.1. Dropout

The Dropout technique (Srivastava et al., 2014; Hinton et al., 2012) is one of the most popular regularization techniques used in Machine Learning algorithms based on ANNs and SGD training strategies. The Dropout is defined as a function $\boldsymbol{z} = f(\boldsymbol{x}; p)$ with two main behaviors:

1. **Model training**: the Dropout function randomly deactivates some component of the vector $\boldsymbol{x}$ according to a probability p. For example, the vector $\boldsymbol{x}$ could correspond to the hidden-layer in a fully-connected layer model. Deactivating a component implies the replacement of that value by a null value equals to 0 in the new output vector $\boldsymbol{z}$. These stochastic deactivations are renewed at each new iteration of the optimization process.
2. **Model evaluation**: after the optimization process in finished, when evaluating the model, the Dropout function is replaced by $\boldsymbol{z} = f(\boldsymbol{x}; p) = k \cdot \mathbf{I}(\boldsymbol{x})$, being now an identity function and weighting the vector with the factor $k = \frac{1}{1-p}$.

The success of the Dropout as a regularization technique is associated with the fact that it aims to prevent the co-adaptation of neighboring units or «neurons» in an ANN-based model.

2.2.16.2. Early Stopping

The early stopping is a very simple regularization technique that is applied along with iterative GD-based optimizations. During training, a Deep Learning model gradually learns to solve a desired task for instances from a training-set $\mathcal{D}_{\text{training}}$. Eventually, this model may learn to make perfect predictions for these instances, overfitting for the training-set $\mathcal{D}_{\text{training}}$. Therefore, the idea of the early stopping is to stop the training process before this overfitting is produced, assuming that an optimal model, given an unknown training iteration, exists for both: the training instances and the new unobserved instances.

To achieve this, a validation-set $\mathcal{D}_{\text{val}}$ is used, which is conformed of instances that are not used for the optimization of the model's parameters during the training process. Given the GD training framework, the idea of the early stopping is to constantly evaluate, after a number of epochs, the performance of the model over the instances of the validation-set $\mathcal{D}_{\text{val}}$, preserving the model version that obtain the best performance for this dataset. The training process stops when no longer and consistent improvement in performance is achieved by training for more epochs, returning the best preserved model (according to the performance using the validation-set $\mathcal{D}_{\text{val}}$) as the final optimized version.

2.2.16.3. Batch Normalization

The Batch Normalization (BN; Ioffe & Szegedy, 2015) is a normalization technique used for Deep Learning architectures. Given a input vector $\boldsymbol{x}_i \in \mathbb{R}^K$ the BN is performed as

follows:

$$\boldsymbol{z}_i = \frac{\boldsymbol{x}_i - \mathbb{E}\left[\boldsymbol{x}\right]}{\sqrt{\text{var}(\boldsymbol{x}) + \varepsilon}} \odot \boldsymbol{\gamma} + \boldsymbol{\beta}, \tag{2.102}$$

where $\boldsymbol{z}_i \in \mathbb{R}^K$ is the normalized vector output, and $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are learnable vectors. The terms $\mathbb{E}\left[\boldsymbol{x}\right]$ and $\text{var}(\boldsymbol{x})$ are the mean and variance calculated over the mini-batches during training, respectively.

For the evaluation of the model, instead of computing the mean and variance of the mini-batches, running estimates are used. The running estimates are defined as follows:

$$\mathbb{E}\left[\boldsymbol{x}\right]^* \leftarrow \alpha\, \mathbb{E}\left[\boldsymbol{x}\right]^* + (1-\alpha)\, \mathbb{E}\left[\boldsymbol{x}\right], \tag{2.103}$$

$$\text{var}(\boldsymbol{x})^* \leftarrow \alpha \text{var}(\boldsymbol{x})^* + (1-\alpha)\text{var}(\boldsymbol{x}), \tag{2.104}$$

where $\mathbb{E}\left[\boldsymbol{x}\right]^*$ and $\text{var}(\boldsymbol{x})^*$ are the running estimates for the mean and variance, respectively. These running estimates are dynamically updated during training and are saved for the posterior evaluation of the models. Note that this process is controlled by the momentum term α.

Following similar ideas of the BN method, several other normalization alternatives could be used, such as the Layer Normalization, Group Normalization, Instance Normalization, etc. In general, the selection of these alternatives depends on the proposed Deep Learning architecture used to process the data.

2.2.16.4. Linear Warm-Up Schedule

In the literature, different strategies have been explored in order to stabilize the optimization process on models based on ANNs and SGD optimization strategies. These strategies are usually used to control the evolution over one of the most important and critical training SGD hyperparameters: the learning rate.

The linear warm-up corresponds to a strategy that gradually increases the learning rate throughout the optimization process. The simplest scenario contemplates a linear increment defined as follows:

$$p = \frac{\min\left(\text{epoch}, \Delta_{\text{epoch}}\right)}{\Delta_{\text{epoch}}}, \tag{2.105}$$

$$\text{lr} = (1-p) \cdot \text{lr}_{\min} + p \cdot \text{lr}_{\max}, \tag{2.106}$$

where the learning rate lr is incremented, from a minimum value $\text{lr}_{\min}$ to a maximum value $\text{lr}_{\max}$, in a defined number of training epochs Δ_{epoch}. Note that $0 \leq p \leq 1$ is a control variable and epoch ≥ 0 corresponds to an integer counter of the training epochs.

Additionally, extra learning rate schedules can be found in the literature, such as the exponential decay and the cosine annealing schedules. These methods offer the flexibility to not only increase the learning rate (e.g., linear warm-up), but also to decrease it (e.g., exponential decay) or to evolve it periodically (e.g., cosine annealing) according to the needs of the problem to be solved and the model to be optimized. Note that the combination of different methods is also possible to obtain more complex learning rate schedule.

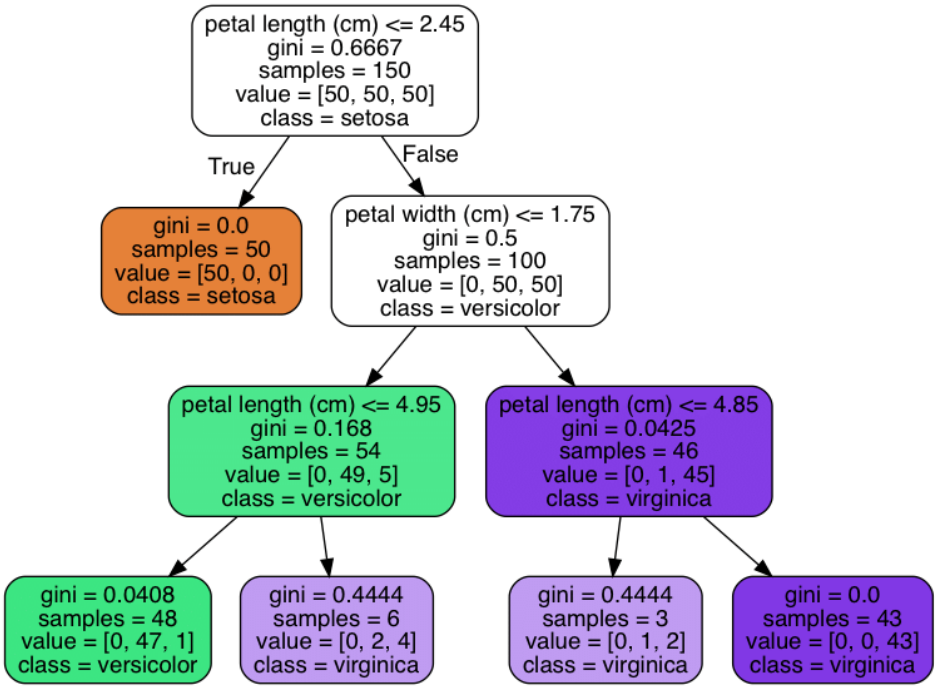

Figure 2.8: Example diagram of an optimized 3-depth decision tree structure for the Iris flower dataset (with 3 classes: setosa, versicolor, and virginica). For each node, the following information is shown: the gini impurity score, the number of samples in the node, the number of samples for each class, and the node predicted class. Original figure extracted from `https://www.researchgate.net/figure/Visualisation-for-a-decision-tree-trained-on-the-Iris-dataset-Source-http_fig5_320384121`.

2.2.17. Decision Trees

The decision trees are Machine Learning tree-like algorithms used to model predictions or event outcomes. Known by their simplicity, these algorithms can be used to solve supervised learning problems, such as classification, regression or even multi-output tasks. Even if the decision trees are simple and highly intuitive algorithms, they can also be very powerful and flexible, capable of fitting complex datasets. These algorithms are being used in real-world problems even up to date, being the core component of other powerful models such as Random Forest.

A decision tree consists of a graph-like collection of multiple conditions branches that control and guide the flow of decisions that an user can select (branch node), ultimately leading to a desired and final prediction outcome given an specific to-solve task. The multiple decisions of a tree are controlled by learnable thresholds which are applied over the conditions of different attributes or features from the input instances. For example, Fig. 2.8 illustrates a simple decision tree used to solve a classification task for the Iris flower dataset[24], where each node can be separated into two different sub-nodes (left and right) according to features such as petal length and petal width, and selected thresholds (e.g., petal width$\leq k_{\text{th}} = 1.75$).

To measure the quality of a decision node, an impurity score is introduced: the gini

[24] `https://www.kaggle.com/arshid/iris-flower-dataset`.

impurity. A node is called pure if the impurity score is 0. Then, given an arbitrary node w, the gini impurity score is defined as follows:

$$G_w = 1 - \sum_{c=1}^{C} p_{w,c}^2, \tag{2.107}$$

where C is the total number of classes in the dataset. The term $p_{w,c}$ is the ratio of the number of samples from the class c w.r.t. the total number of samples in the node w. For example, the gini impurity score of the depth-3 green node, of the previous diagram (Fig. 2.8), is calculated as follows: $G = 1 - \left(\frac{0}{48}\right)^2 - \left(\frac{47}{48}\right)^2 - \left(\frac{1}{48}\right)^2 \approx 0.0408$.

Alternatively, an impurity score, based on the entropy operator, can be defined as follows:

$$G_w = -\sum_{c=1}^{C} p_{w,c} \log\left(p_{w,c}\right). \tag{2.108}$$

2.2.17.1. Classification and Regression Trees (CART)

The Classification and Regression Trees (CART) algorithm is the iterative strategy used to, given a task to solve, automatically build the optimized structure of a decision tree and to learn the correct decision rules in each node (threshold values).

As usual in Machine Learning algorithms, a loss function must be defined in order to solve a proposed optimization problem (minimization). Given an arbitrary feature pair and threshold pair $\{k, k_{\text{th}}\}$, the loss function to minimize in the CART algorithm is defined as follows:

$$\mathcal{L}(k, k_{\text{th}}) = \frac{m_{\text{left}}}{m_{\text{left}} + m_{\text{right}}} G_{\text{left}} + \frac{m_{\text{right}}}{m_{\text{left}} + m_{\text{right}}} G_{\text{right}}, \tag{2.109}$$

where G_{left} and G_{right} are the impurity score of the left and right sub-nodes, respectively; and m_{left} and m_{right} are the number of instances in the left and right sub-nodes, respectively. Note that this definition corresponds to a weighted sum of the two impurity scores of both sub-nodes: the left and right nodes.

Intuitively, the idea of the CART algorithm is to first split the complete dataset into two sub-nodes (left and right) by using a threshold value k_{th} over the feature k, e.g., petal length$\leq k_{\text{th}} = 2.45$ (Fig. 2.8). In order to use the optimal pair $\{k, k_{\text{th}}\}$, the loss function defined above is computed: given the feature k, the threshold value k_{th} that minimizes the loss function $\mathcal{L}(k, k_{\text{th}})$ is found, i.e., the threshold value that produces the purest sub-nodes (weighted sum).

After this first initial split, a new split can be performed, by minimizing a new loss function, over the collection of instances of each of the created sub-nodes by selecting again the optimal pair $\{k, k_{\text{th}}\}$, e.g., petal width$\leq k_{\text{th}} = 1.75$ (Fig. 2.8). This process can be then performed iteratively for the sub-nodes produced by each new split, gradually building a complete decision tree. The stop criterion of this process is when the impurity score of a node can not be longer minimized, when a defined maximum depth tree value is reached or when other type of terminal conditions are satisfied (controlled with other decision tree hyperparameters).

2.2.18. Random Forest (RF)

The Random Forest (RF; Breiman, 2001) is an ensemble model that consist of a group of different decision tree predictors (ensemble learning system). The idea is to aggregate all the predictions obtained from these basic predictors. Usually, by doing this, a better, more accurate, and robust final prediction can be obtained than the one obtained from a single predictor («wisdom of the crowd»). For example, in the classification problem scenario, the final class prediction is computed by counting the class votes (predictions) from all the decision trees from the ensemble. Then, the final class prediction corresponds to the most voted class.

In order to induce a degree of variability in this type of ensemble model, two main strategies are usually introduced in the RF model: a random selection of samples and a random selection of features for each decision tree. First, the instance objects that are used to optimize (or train) each one of the different decision trees in the RF are different. In general, the instances for each tree are sampled by using the boostrap aggregating method (a.k.a. bagging), i.e., by generating a new subset by performing a stratified sampling, with replacement, of instances drag from an original complete dataset $\mathcal{D}$. Second, the sets of input features that are used by each tree are also randomly selected. Each decision tree, in the RF model, is optimized by using a random selection of the original set of features. For example, random sets of $\log_2(n)$ or $\sqrt{n}$ number of features are selected, where n is the number of original features in the dataset $\mathcal{D}$[25].

The optimization process of the RF, along with other model settings, can be controlled by setting several RF hyperparameters, such as the number of decision trees (estimators), the quality of a split criterion in the decision trees (e.g., gini, entropy), the maximum reachable depth in the decision trees, the number of features used to optimize each decision tree, among others.

As a very popular and versatile algorithm, the RF model has some notable advantages:

1. **Little data preparation**: in general, very little feature pre-preprocessing is required to use the RF model. Usually, no scaling, normalization or centering are required as the decision trees can properly handle different range of values for each of the processed features in the dataset. This makes the RF model very easy and straightforward to use.
2. **Interpretability**: a ranking of features can be obtained from an optimized RF. Ideally, this ranking describes, from the highest to the lowest, the importance or relevance that each one of the features has over the final prediction of the model. While having the ranking may offer us a certain degree of interpretability, it should be noted that this tool is not always reliable for all situations. For example, it may produce results that can be misleading when random forest variable importance measures are used with data of varying types (Strobl et al., 2007).

Depending on the dataset, one main disadvantage of the RF could be the requirement of extra meaningful features extracted from the original instances to be processed by the model.

[25]Note that, instead of a random selection, all the n features can be used.

For example, the astronomical light-curves are highly complex objects. In order to use them along with the RF model, a set of astrophysical or hand-crafted features are required to be pre-computed from these light-curves. In general, this task, also called feature engineering, is non-trivial and can be highly challenging and computationally costly, requiring time for a streaming application.

2.2.18.1. Balanced Random Forest (BRF)

The Balanced Random Forest (BRF; Chen et al., 2004) model is a variation of the RF model. The BRF algorithm aims to deal with the possible class imbalance presented in the dataset $\mathcal{D}$ that is used to optimized the model, i.e., when one or several classes (majority classes) present a notable higher number of instances than the others (minority classes). Usually, this type of class imbalance can be highly detrimental for the performance of some algorithms such as the RF, where a bias towards the majority class can be produced: the model may well-characterize instances from the majority class while ignoring the minority class.

To deal with the class imbalance, the BRF train each decision tree of a RF model by using a balanced bootstrapped sampling method, i.e., the samples are equally distributed in terms of the number of instances corresponding to each one of the classes in the dataset. This type of bootstrapping aims to allow the model to well-represent the instances from the minority class as well as those of the majority class.

2.2.19. Dimensionality Reduction

When working with real-world data, it is usual to find data that is described in term of high-dimensional structures, i.e., by using vector instances with a high number of dimensions. In general, the high number of dimensions can be a very challenging problem for Machine Learning and statistical methods and algorithms: an scenario known as the «curse of dimensionality». To deal with this type of data, a popular alternative is the dimensionality reduction. These techniques aim to automatically reduce the dimensional space of the original data, from a large number of dimensions K, to a lower and more tractable number of dimensions $K' < K$. For instance, even if a handwritten number image can be a high-dimensional object (with thousands of pixels), some basic structures can be described in order to adequately represent the content in a lower dimensional space, e.g., strokes, curves, circles, pixel correlations. On the other hand, some dimensions of the data can offer little or zero information and could be ignored, e.g., the border pixels are usually empty in a handwritten number image.

Another popular use of dimensionality reduction techniques is for data visualization. The comprehension of high-dimensional data can be a difficult task for the human brain; therefore, a good alternative is to find a proper way to represent this type of data into a lower number of dimensions. For example, the dimensionality reduction technique can be used to represent high-dimensional data into 2D or 3D maps that can be easily visualized and explored, allowing us to look for trends, relationships, class separations, structures like

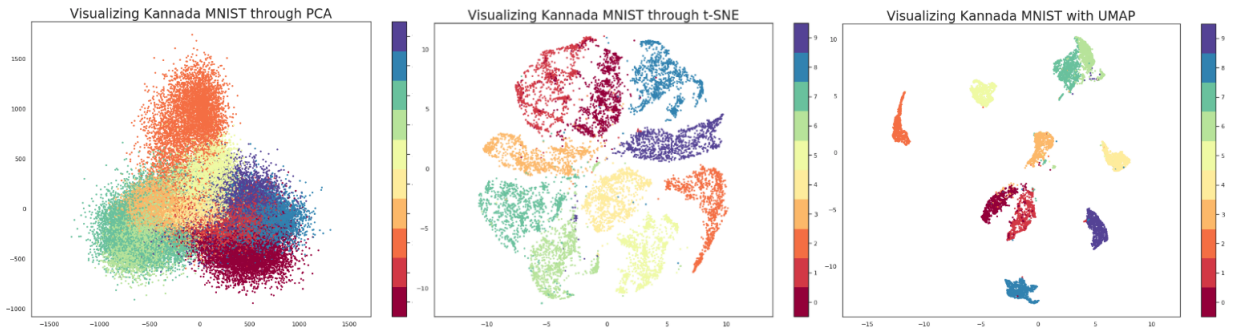

Figure 2.9: Examples of three dimensionality reduction techniques for the Kannada MNIST dataset: PCA, t-SNE, and UMAP. By representing each class with a different color, a class separation can be observed in the scatter maps for each method. This class separation is more appreciable depending of the method used, e.g., t-SNE, UMAP. Original figure extracted from `https://www.kaggle.com/parulpandey/part3-visualising-kannada-mnist-with-umap`.

clusters, outliers in the distributions, among others. Fig. 2.9 illustrates three dimensionality reduction techniques for the Kannada MNIST dataset[26], representing handwritten number images of 784 pixels (high-dimensional space) into 2D scatter maps (low-dimensional space).

2.2.19.1. Principal Component Analysis (PCA)

The Principal Component Analysis (PCA) algorithm is a popular, simple, and intuitive dimensionality reduction method. The goal of PCA is to build a new low-dimension hyperspace to project the original high-dimension data while preserving the maximum original variance. To solve this, the PCA algorithm is strongly based on the use of the covariance matrix. As a simple and straightforward to implement algorithm, the PCA is usually used as a pre-processing reduction for other and more complex dimensionality reduction techniques (e.g., t-SNE, UMAP).

Given a set of K-dimensional vector instances (i.e., a data instance can be described with K features values), a covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{K \times K}$ can be computed by assuming a random vector framework (see section §2.2.2.5 for details). Then, the associated eigenvectors $\boldsymbol{v}_i \in \mathbb{R}^K$ and eigenvalues $\lambda_i \in \mathbb{R}$ of $\mathbf{\Sigma}$ must satisfy the following condition:

$$\mathbf{\Sigma}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i, \tag{2.110}$$

where $\boldsymbol{v}_i$ is the i-th eigenvector associated with the eigenvalue λ_i.

In PCA, the eigenvectors of the covariance matrix $\mathbf{\Sigma}$ are calculated and a new orthogonal space, where the dataset is going to be projected, is defined. The eigenvector $\boldsymbol{v}_i$ associated with the eigenvalue λ_i is called the i-th principal component of the PCA. Then, the eigenvector with the highest eigenvalue is the 1st principal component and it is aligned in

[26] `https://www.kaggle.com/c/Kannada-MNIST`.

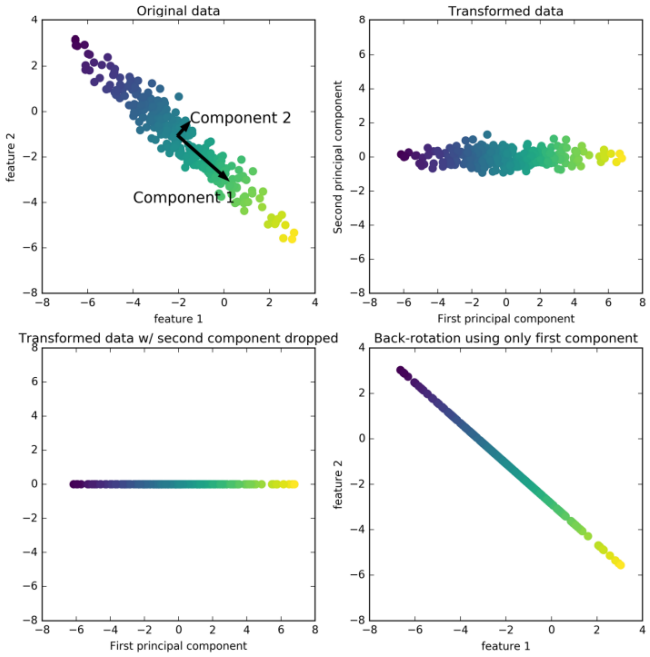

Figure 2.10: A simple example of the PCA algorithm. The first two principal component vectors are shown over the original data space. The projection over the first principal component is also shown. Original figure extracted from Müller & Guido (2017).

the direction of the highest dataset variance or dispersion. The 2nd principal component is orthogonal to the 1st principal component and it is associated with the second highest eigenvalue. Then, when working with K-dimensional data, the PCA algorithm construct K eigenvectors defining an orthogonal space of K principal components.

In order to reduce the dimensionality of the original dataset, the first $K' < K$ principal components should be selected. Then, all instances must be linearly projected into the new orthogonal space defined by the first K' principal components. The PCA algorithm aims to perform this reduction with a minimum loss of information by the assumption that it is desired to preserve the maximum variance information of the data, that is, by preserving the maximum statistical moment-2 criterion (minimum square error). For example, Fig. 2.10 illustrates the projection of the PCA algorithm over the 1st principal component.

2.2.19.2. Uniform Manifold Approximation and Projection (UMAP)

The Uniform Manifold Approximation and Projection (UMAP; McInnes et al., 2018) is a complex dimensionality reduction method that can be used for general nonlinear dimension reduction. The UMAP is founded on a solid mathematical background, where three assumptions about the data are required:

1. The data is uniformly distributed on a Riemannian manifold.
2. The Riemannian metric is locally constant (or can be approximated as such).
3. The manifold is locally connected.

Given these assumptions, it is possible to model the manifold by using a fuzzy topological structure. The embedding can be found by searching for a low dimensional projection of the data that has the closest possible equivalent fuzzy topological structure. The UMAP works under the influence of several important hyperparameters that controls the low dimensional projection of the data, e.g., number of neighbors, minimum distance, distance metric. In general, the selection of these hyperparameters is non-trivial and highly depends on the data[27].

Additionally, a parametric extension of UMAP was proposed in Sainburg et al. (2020): the Parametric UMAP. This method extends the second step of UMAP to a parametric optimization by using ANNs, learning a parametric relationship between the data and the embedding. Some advantages of the Parametric UMAP method are: 1) The optimization using ANNs usually reduces the computation costs and time. 2) The non-stochastic parametrization solves the intrinsic stochasticity of the UMAP algorithm, correctly producing the same output embeddings given the same data inputs[28].

2.3. Astronomy

Astronomy is the natural science that deals with the study of celestial bodies that populate the vast cosmos, identifying important phenomena that occur in the universe. This science field has a strong and complex theoretical background in different areas such as mathematics, physics, and chemistry. In this section, several concepts of astronomy are briefly described.

2.3.1. Optical Spectrometry

The optical spectrometry is a technique that allows scientists to study the interaction between the matter and the electromagnetic radiation. In astronomy, the optical spectrometry allows astronomers to inspect which are the different chemical components that affect the behaviour of the electromagnetic radiation emitted by a specific light-source (astronomical object or event). The characterization of the light-source can be achieved by measuring the absorption lines presented in a spectral profile, allowing scientists to build chemical profiles of the light-sources, which often are known as the «chemical fingerprints». By using spectrometry techniques, it is possible to identify concentrations of Hydrogen, Helium, traces of Silicon, among other elements, which, for instance, has helped to classify different types of SNe: SNIa, SNIb, SNIc, SNIIb, SNIIn, SNIIL, SNIIP, SLSN, etc (Petschek, 1990; Alsabti & Murdin, 2017).

Given the immense investment of time and human effort that optical spectrometry requires, only a marginal proportion of the data collected has been effectively studied and followed-up, especially given the new astronomical Big Data paradigm. This also heavily restrains the availability of other variables extracted from spectral studies such as the redshift.

[27]See `https://umap-learn.readthedocs.io/en/latest/parameters.html` for a deep explanation about the UMAP hyperparameters.

[28]see details about this issue in `https://github.com/lmcinnes/umap/issues/566`.

Roughly speaking, the redshift is the Doppler effect of light: if a light-source is moving away from the observer (planet Earth), the wavelengths of the perceived light are elongated. As a human interpretation of the color, the longer wavelengths are redder than the shorter ones, coining the name of the redshift effect. The redshift information, as metadata, may be useful for the characterization of the SNe (Boone, 2019; Möller & de Boissière, 2020), but consistently relying on this type of information is highly unexpected, especially in the Big Data paradigm.

2.3.2. Optical Photometry

The optical photometry is a technique that allows scientists to measure the perceived light level, by an observer, from a light-source. In astronomy, the photometry is used to extract the brightness level information from a astronomical object or event given a specific range of frequency (photometric band). In an optical telescope, a Couple Charge Device (CCD) is used to capture the photons from a light-source. Each pixel in the CCD can count the number of electrons charges, in Analog to Digital Units (ADUs) units, over a period of exposure time (controlled by a shutter). These counts are proportional to the brightness observed from the light-source. Finally, this photometry information can be transformed into other brightness measurement units such as the photometric flux or the apparent magnitude. In addition, consistently using the optical photometry over time allows astronomers to construct an useful type of time series: the light-curves.

In general, the photometry process can be performed by using multiple and different operating ranges of light frequencies, called the «photometric bands», which correspond to optical filters located in the telescopes used to capture the electromagnetic information from the astronomical objects. By using different types of bands, the information captured by a particular astronomical object can be described in terms of different modalities, providing a more complete profile to characterize a light-source. As an example, Fig. 2.11 shows the wavelength ranges of the bands of the commonly used SDSS system. Note that the use of multiple bands allows the construction of multi-variate time series: the multi-band light-curves (see section §2.3.2.4 for extra details).

2.3.2.1. Photometric Flux

The photometric flux is the measurement of the brightness captured, by an observer (planet Earth), from a light-source in the sky (astronomical object or event). The flux is directly related with the photons received by an observer and, hence, with the estimated brightness. In general, the brightness measurements in astronomy are strongly subject to a vastly family of sources of noise and uncertainty, e.g., unfavorable atmospheric conditions, light from the moon. Therefore, it is common to associate the flux measurement with an error estimation (observation-error). This observation-error is denoted by $\sigma_{F_s} \geq 0$ when working with the flux unit, obtaining brightness measurements of the form $F_s \pm \sigma_{F_s}$, where $F_s \geq 0$ is the photometric flux.

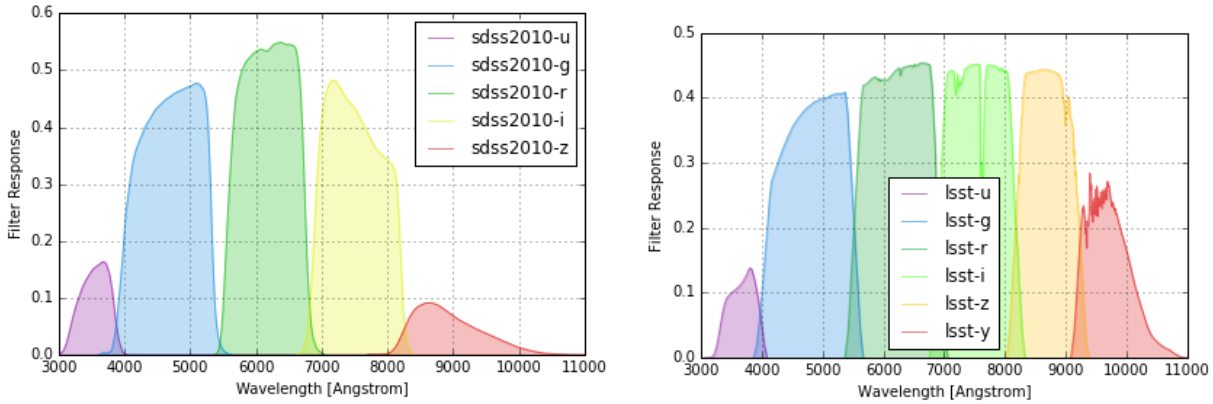

Figure 2.11: Wavelength ranges of the bands of the SDSS system. Left) The standard SDSS bands system is shown: {u, g, r, i, z}. Right) As a comparison, the LSST bands system is shown: {u, g, r, i, z, y}. Original figure extracted from `https://community.lsst.org/t/lsst-filters-versus-sdss/2429/6`.

2.3.2.2. Apparent Magnitude

Another widely used brightness measurement is the apparent magnitude, which is a measurement inversely proportional to the brightness of a star that is observed from the planet Earth. Then, a larger apparent magnitude implies that the observer perceives less brightness from the object. The apparent magnitude $m \geq 0$ is defined as follows:

$$m = -2.5 \log_{10} \left(\frac{F_s}{F_{\text{ref}}} \right), \tag{2.111}$$

$$m = -2.5 \log_{10} (F_s) + z_p, \tag{2.112}$$

where F_s is the photometric flux captured from the observed light-source and F_{ref} is a reference flux. Another way to represent m is to use an auxiliary variable z_p (zero point), which includes the information captured by the term F_{ref} and it is related to the absorption and diffraction rates of the color produced by the atmosphere. Similar to the flux measurements, the apparent magnitude has an associated observation-error. Therefore, the magnitude measurements are expressed in the form $m \pm \sigma_m$, where $\sigma_m \geq 0$ is the magnitude observation-error.

2.3.2.3. Magnitude-Flux Transformation

In order to convert an apparent magnitude measurement, to a flux measurement, the following transformation is required:

$$F_s(m) = 10^{-\frac{1}{2.5}(m - z_p)}, \tag{2.113}$$

$$\sigma_{F_s}(m, \sigma_m) = 10^{-\frac{1}{2.5}(m - z_p)} - 10^{-\frac{1}{2.5}((m + \sigma_m) - z_p)}, \tag{2.114}$$

where, the flux and flux error are computed in eqs. (2.113) and (2.114), respectively.

Figure 2.12: Example of a sequence of stamp-images captured over a period of time. The science, template, and difference are shown for the object ExZTF19aavounq from the ZTF survey. Original figures extracted from `https://alerce.online/object/ZTF19aavounq`.

2.3.2.4. Multi-Band Light-Curve

Astronomical events, such as SNe, can be characterized by using light-curves, which are time series structures that describe the evolution or change of the brightness of an astronomical object over time. Therefore, a light-curve can be defined as a collection of observations, over time, that contains information of the brightness in terms of photometric counts, flux or apparent magnitude. For instance, in the case of SNe, the typical light-curve shows a phase of increasing the brightness (SN-rise), a peak of the brightness (SN-peak), and then a decay (SN-fall) until the extinction of the brightness (SN-extinction).

First, given an astronomical object of interest, a sequence of stamp-images is captured by a telescope over a period of time. Usually, these sequences are composed of three types of stamp-images: science, template, and difference stamp-images (see Fig. 2.12 for an example). Then, complex techniques such as the Naylor's optimal photometry (Naylor, 1998) are used to extract the brightness measurements from the sequence of stamp-images (in photometric counts, flux or apparent magnitude units). As a sequential measurement of an astronomical object over time, this process can be used to construct a light-curve object.

As discussed in previous sections, an error estimation is associated with the brightness measurements. Therefore, the light-curves are constructed as a sequence or time series containing both, the brightness information (observation) and the brightness error information (observation-error) associated with that observation. The brightness observations of the light-curve can also be captured by using different frequency ranges along with the optical filters of the telescope (photometric bands). This produces multi-band light-curves or, in a broader context, multi-variate time series.

The change in brightness can be produced by either intrinsic physicochemical reactions of the astronomical object or by other external effects and phenomena. For instance, the light-curves are strongly subject to independent factors of the observed astronomical event, such as the atmospheric conditions, telescope state, among many others. Ultimately, this produces

challenges such as light-curves with an arbitrary number of observations (variable-length), a highly irregular sampling rate (irregular cadence), a non-aligned multi-variability among the photometric bands, etc.

2.3.3. Supernovae (SNe)

The supernovae (SNe) are transient astronomical events resulting from a powerful and bright stellar explosion. There is a wide range of types of SNe, leading to a vast nomenclature. In general, the nomenclature of SNe is guided by their electromagnetic spectrum, often known as the «chemical fingerprint», which allows the astronomers to inspect which are the different chemical elements that affect the behaviour of the light emitted by the light-source (absorption lines). Therefore, it is possible to identify concentrations of Hydrogen, Helium, traces of Silicon, among other elements, giving rise to different types of SNe such as SNIa, SNIb, SNIc, SNIIb, SNIIn, SNIIL, SNIIP, SLSN, etc (Petschek, 1990; Alsabti & Murdin, 2017).

There are several plausible theories and explanations that attempt to describe the SN phenomenon, in particular, the explosion mechanism that trigger this astronomical event. In general, the SNe can be separated into two main groups: thermonuclear and core-collapse SNe.

2.3.3.1. Thermonuclear SNe

A thermonuclear SN is produced by an imbalance in the nuclear fusion process in a stellar close binary system, which contain a white dwarf star. Given special circumstances, the white dwarf progressively increases its mass due to material absorbed from its companion in the binary system (where a short distance separates the two stars). When the white dwarf reaches the Chandrasekhar limit (about 1.44 solar masses) a thermonuclear runaway occurs due to a positive feedback in the nuclear fusion mechanism, which ultimately leads to the explosion of the white dwarf (Petschek, 1990; Alsabti & Murdin, 2017).

The above mechanism is thought to be the particular cause of the type Ia SNe (thermonuclear SNe), which show a high consistency of the reached maximum apparent brightness. This behaviour could be explained due to the particularly stable intrinsic conditions that trigger the explosion mechanism of the progenitor star. Given this high consistency, the SNIa type is also known as the cosmological standard candle (Wright & Li, 2018). Due to its peculiar properties, the study of the SNIa type has played a critical role in several scientific and astronomical research fields, allowing cosmologists to measure extra-galactic distances in the cosmos, which ultimately led to the discovery of the accelerated expansion of the universe (nobel prize in 2011; Schmidt et al., 1998; Riess et al., 1998) and its precise characterization with projects such as the Dark Energy Survey (DES; Sánchez, 2006; Dark Energy Survey Collaboration et al., 2016).

2.3.3.2. Core-Collapse SNe

Theoretically, the core-collapse explosion scenario explains the birth of the great majority of SNe, e.g., SNIb/c, SNII, SLSN. A core-collapse SN is produced by the collapse of the star nucleus due to the energetic exhaustion of the nuclear fusion process, breaking the existing equilibrium between the thermonuclear repulsion and the gravitational attraction of the stellar plasma. Due to this imbalance, the gravitational force ends up being predominant, producing a strong implosion of the nucleus and a rebound of the surrounding material, which shoots out in a large and abrupt explosion (Petschek, 1990; Alsabti & Murdin, 2017).

Additionally, the study of Type Ib/c and Type II supernovae (core-collapse SNe) has helped astronomers to understand the evolution and explosion mechanisms of stars, including insights about the formation of stellar mass black holes (Sukhbold & Adams, 2020).

2.3.4. Astronomical Surveys, Datasets, and Brokers

The astronomical surveys are the execution of a plan to perform a stellar mapping of the sky. This mapping is performed by using images from specialized telescopes, defining which regions of the sky will be captured, as well as in which bands and dates. Usually, surveys are designed with the objective of capturing a special type of astronomical object or event, such as transient events or SNe. For example, the Zwicky Transient Facility survey (ZTF; Bellm et al., 2019) was designed to detect objects with a rapid and abrupt change in brightness, such as transient events like SNe, Gamma Rays or collisions between stars. Table 2.2 shows the properties of several surveys along with extra astronomical datasets, such as simulated SN light-curves datasets.

Usually, the astronomical surveys produce a high-volume of data collected from the sky every night (stream of alerts). To collaborate with the process of these data, the alerts can be distributed to the called community «brokers», such as the Automatic Learning for the Rapid Classification of Events (ALeRCE; Förster et al., 2021; Sánchez-Sáez et al., 2021; Carrasco-Davis et al., 2021), Arizona-NOIRLab Temporal Analysis and Response to Events System (ANTARES; Matheson et al., 2021), Lasair (Smith et al., 2019), among others. Therefore, a broker is a public pipeline that will ingest and process the stream of astronomical alerts, helping the scientific community to handle the high-volume data with task such as cross-match association, identification and prioritization of astronomical objects to follow-up, light-curve analysis, etc. For instance, the Chilean ALeRCE broker performs a real-time processing of astronomical events captured by the ZTF survey. The broker can process discovery images and light-curves, automatically extracting astrophysical features. These features are then used, along with Machine Learning methods, to classify detected astronomical events into three main different categories: transient, variable, and stochastic events.

Table 2.2: Brief summary of some astronomical datasets. Most of the datasets correspond to empirical data from surveys projects, while the rest correspond to simulated data. For the multi-band system, we denote SDSS ({u, g, r, i, z}) and INT WFC ({U, B, V, R, I, Z}) for SDSS-like and INT WFC-like multi-band systems, respectively. *: datasets/surveys that use other multi-band system.

Acronym	Survey name	Simulated data	Number of bands	Multi-band system
MACHO (Alcock et al., 1996)	Massive Compact Halo Object		2	*
SDSS (York et al., 2000)	Sloan Digital Sky Survey		5	SDSS
LINEAR (Stokes et al., 2000)	LIncoln Near-Earth Asteroid Research		1	SDSS
OGLE-III (Udalski, 2003)	Optical Gravitational Lensing Experiment		2	INT WFC
ASAS (Pojmanski, G., Pilecki, B., Szczygiel, 2005)	All Sky Automated Survey		2	INT WFC
DES (Sánchez, 2006; Dark Energy Survey Collaboration et al., 2016)	Dark Energy Survey		5	SDSS
CRTS (Drake et al., 2009)	Catalina Real-Time Survey		1	INT WFC
SNANA (Kessler et al., 2009)	Supernova Analysis software	✓	≥ 1	SDSS
WISE (Wright et al., 2010)	Wide-field Infrared Survey Explorer		4	*
SPCC (Kessler et al., 2010)	Supernova Photometric Classification Challenge	✓	4	SDSS
HiTS (Förster et al., 2016)	The High Cadence Transient Survey		1	SDSS
Gaia (Gaia Collaboration, 2016)	Gaia		3	*
Pan-STARRS (Chambers et al., 2016)	Paranoramic Survey Telescope and Rapid Response System		5	SDSS
PLAsTiCC (The PLAsTiCC team et al., 2018)	Photometric LSST Astronomical Time-Series Classification Challenge	✓	6	SDSS
ASAS-SN (Jayasinghe et al., 2018)	All Sky Automated Survey for Supernovae		2	SDSS, INT WFC
ZTF (Bellm et al., 2019)	Zwicky Transient Facility		2	SDSS
LSST (Ivezić et al., 2019)	Vera C. Rubin Observatory Legacy Survey of Space and Time		6	SDSS

2.3.5. Supernova Parametric Model (SPM)

The Supernova Parametric Model (SPM; Villar et al., 2019; Sánchez-Sáez et al., 2021) is an analytical function that attempts to describe the typical behavior of a SN light-curve. The SPM function is defined as follows:

$$f_{\rm sne}(t;\boldsymbol{\theta}) = f_{\rm early}(t)\cdot(1-g(t)) + f_{\rm late}(t)\cdot g(t), \tag{2.115}$$

$$g(t) = \sigma(s\cdot(t-(\gamma+t_0))), \tag{2.116}$$

$$f_{\text{early}}(t) = \frac{A \cdot \left(1 - \beta' \frac{(t-t_0)}{\gamma}\right)}{1 + \exp\left(\frac{-(t-t_0)}{\tau_{\text{rise}}}\right)}, \tag{2.117}$$

$$f_{\text{late}}(t) = \frac{A \cdot (1-\beta') \cdot \exp\left(\frac{-(t-(\gamma+t_0))}{\tau_{\text{fall}}}\right)}{1 + \exp\left(\frac{-(t-t_0)}{\tau_{\text{rise}}}\right)}, \tag{2.118}$$

where the SPM function $f_{\text{sne}}(t) : \mathbb{R} \mapsto \mathbb{R}$ allows us to construct a light-curve (flux) for a SN by evaluating the SPM model, given a vector of SPM parameters $\boldsymbol{\theta} = [A, t_0, \gamma, \beta', \tau_{\text{rise}}, \tau_{\text{fall}}]^T$, along with a collection of arbitrary and continuous-time values $\{t_j\}_{j=1}^{L}$ (days). This analytical function is defined as a smooth transition between an early function $f_{\text{early}}(t) : \mathbb{R} \mapsto \mathbb{R}$, in eq. (2.117), and a late function $f_{\text{late}}(t) : \mathbb{R} \mapsto \mathbb{R}$, in eq. (2.118). The use of these functions aims to characterize a typical SN behavior: the brightness abruptly increases (SN-rise) up to a maximum (SN-peak), followed with a decrease (SN-fall) (where a plateau or radioactive tail (SN-plateau) could be observed), and ending with the final extinction of the transient event (SN-extinction). The transition between the early and late functions is controlled by the function $g(t) : \mathbb{R} \mapsto (0, 1)$, where σ is the sigmoid function and $s = .2$ is a transition smoothness control factor.

Intuitions behind the 6 SPM parameters are given by:

1. $A \in \mathbb{R}^+$: affects the brightness scale for the SN light-curve.
2. $t_0 \in \mathbb{R}$: acts as a temporal shift for the light-curve. Even though this value is close to the light-curve maximum brightness, it does not exactly correspond to the SN-peak time.
3. $\gamma \in \mathbb{R}^+$: controls the time duration of the SN-plateau region.
4. $\beta' \in [0, 1]$: controls the slope of the SN-plateau region.
5. $\tau_{\text{rise}} \in \mathbb{R}^+$: controls the required time to reach the maximum brightness along the light-curve.
6. $\tau_{\text{fall}} \in \mathbb{R}^+$: controls the brightness decay time along and after the SN-plateau region.

To better illustrate the behavior of the SPM, Fig. 2.13 shows several light-curves obtained by changing some of the SPM parameter in a particular range. It can be observed how the model aims to characterize the SN typical regions described above: SN-rise, SN-peak, SN-plateau, SN-fall, and SN-extinction.

2.3.6. Computation of Astrophysical Features for Light-Curves

The astronomical light-curves are challenging objects to handle due to their intrinsic irregular cadence, variable-length, multi-band observations, among others. Therefore, several descriptors (features) can be extracted from the light-curves by using irregular time-series related methods and astrophysical knowledge. These features aim to characterize the general

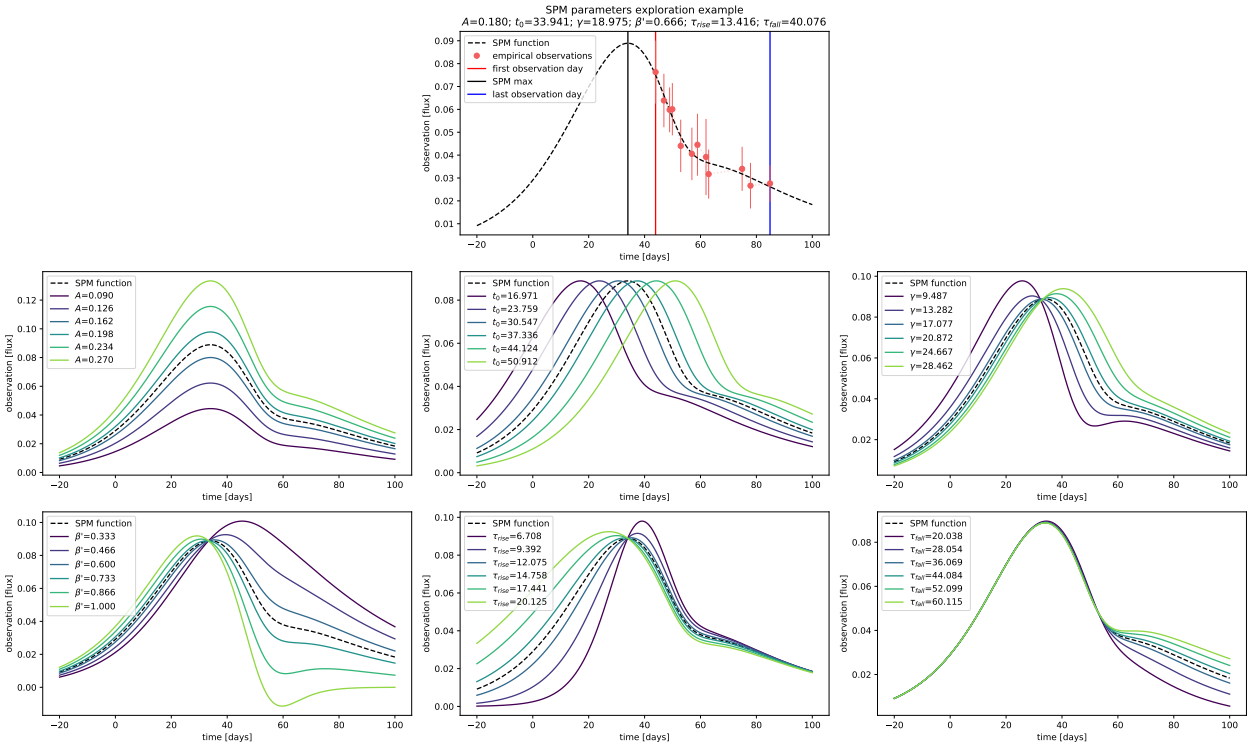

Figure 2.13: Examples of different arbitrary and simulated SN light-curves (represented by the SPM function) produced by changing the parameters of the SPM.

behavior of a variable-length multi-band light-curve Φ_i into a fixed-length feature vector $\boldsymbol{z}_i = f(\Phi_i) \in \mathbb{R}^K$, which is composed of K descriptors.

For example, some features can be statistics such as the mean or standard deviation of the observations in a light-curve. Other features can be obtained by complex methods, such as the Lomb–Scargle periodogram. This periodogram can be used to estimate periods in a light-curve by fitting a set of sinusoidal functions to the observations (Scargle, 1982). Other type of functions can be also fitted to the light-curves, such as analytical models of SNe (Villar et al., 2019). All these extracted features can be used to classify astronomical events along with classical Machine Learning models, such as the Balanced Random Forest (BRF), Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), Gradient Boosting, etc.

In the literature, a vast and diverse family of astrophysical feature extraction methods has been proposed over the years (Sánchez-Sáez et al., 2021; Stellingwerf & Donohoe, 1986; Richards et al., 2011; Villar et al., 2019; Kim et al., 2011; Arévalo et al., 2012; Nun et al., 2015; Kim et al., 2014; Schmidt et al., 2010; Graham et al., 2017; Tachibana & Miller, 2018; Huijse et al., 2018; Allevato et al., 2013; McLaughlin et al., 1996; Eyheramendy et al., 2018; Mondrik et al., 2015)[29]. In addition, Application Programming Interfaces (APIs), such as the Feature Analysis for Time Series (FATS; Nun et al., 2015), has helped the scientific community to easily incorporate feature extraction methods into light-curve processing pipelines. As a real-world example, the Automatic Learning for the Rapid Classification of Events (ALeRCE; Förster et al., 2021; Sánchez-Sáez et al., 2021; Carrasco-Davis et al., 2021) broker can process,

[29]See `http://alerce.science/features/` for extra details of these features.

in real-time, empirical light-curves from the ZTF survey. For this, the broker computes a vast set of features based on prior astrophysical expert knowledge (Sánchez-Sáez et al., 2021).

Chapter 3

Methodology

3.1. Dataset and Pre-Processing

3.1.1. Dataset

In this work, we use a dataset $\mathcal{D}$ that consists of a collection of flux SN multi-band light-curves from the Zwicky Transient Facility survey (ZTF; Bellm et al., 2019), composed of two bands: g and r. These SN events have been confirmed spectroscopically and reported in the Transient Name Server (TNS) catalog[1]. As we aim to classify different types of SNe, the following SN types are used from the dataset $\mathcal{D}$: SNIa, SNIbc, SNII, and SLSN, as researched in Sánchez-Sáez et al. (2021). We remove short-length SN multi-band light-curves: only multi-band light-curves having at least $L_i^{(b)} \geq 5$ observations, in any of the B bands, are preserved in the dataset $\mathcal{D}$. Fig. 3.1 shows the class distribution, where a high class imbalance can be observed with majority classes (SNIa, SNII) and minority classes (SLSN, SNIbc).

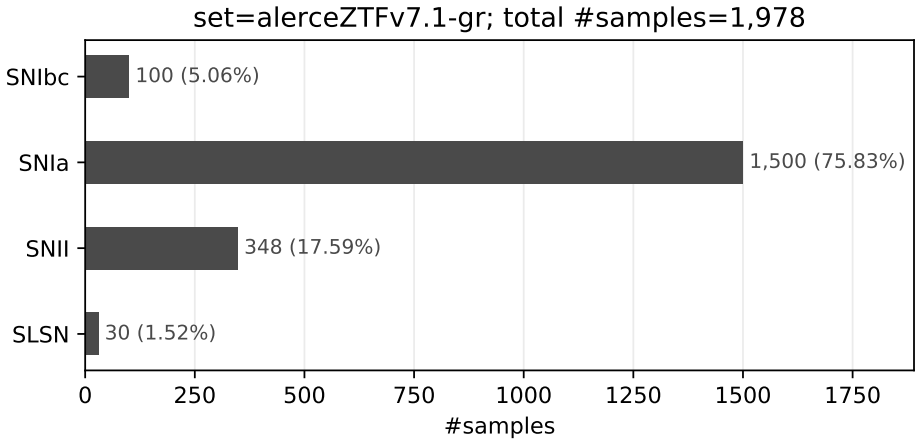

Figure 3.1: Class population distribution of SN types (from the original dataset $\mathcal{D}$).

[1] `https://wis-tns.weizmann.ac.il`.

3.1.2. Pre-Processing

In order to prepare the dataset and light-curves, the following pre-processing procedures are implemented.

3.1.2.1. Stratified 5-Fold Cross-Validation

A non-stochastic 5-fold cross-validation procedure is performed. The dataset $\mathcal{D}$ is split into 5 different variations of training/validation/test sets, following the proportion $\#(\mathcal{D}_{\text{training}})$ $/\#(\mathcal{D}_{\text{val}})/\#(\mathcal{D}_{\text{test}}) = 60/20/20$. The imbalance of classes is similar for all sets as this split is stratified. When performing the splits, we aim to ensure that each fold configuration is almost unique, ensuring that each SN light-curve appears at least once in some of the test-set variations. This methodology aims to correctly preserve the representativeness of each split, which is important for the minority classes.

3.1.2.2. Simultaneous Multiple Observations

Given a single-band light-curve $\Phi_i^{(b)}$, all photometric observations reported within a short-range time-window $\Delta t = 12$ [hours] are merged into a single observation. Close observations are merged because multiple observations in the same night can be redundant and can harm the performance of some algorithms. Given an arbitrary group of close observations, the merging process is performed by using weight factors $w_{i,j^{(b)}} \in [0, 1]$ defined as follows:

$$w_{i,j^{(b)}} = \frac{e^{-\log\left(\sigma_{i,j^{(b)}}+\varepsilon\right)}}{\sum_{j^{(b)}\in\Delta J_i^{(b)}} e^{-\log\left(\sigma_{i,j^{(b)}}+\varepsilon\right)}}, \forall j^{(b)} \in \Delta J_i^{(b)}, \tag{3.1}$$

where $\sigma_{i,j^{(b)}}$ is the observation-error and $\Delta J_i^{(b)}$ is a subset of the sequence steps associated with all the observations, in the single-band light-curve $\Phi_i^{(b)}$, that are sharing the same time-window Δt. A new observation-flux can be generated as a weighted sum of all close observations within Δt, as $\mu_{i,j^{(b)}} \leftarrow \sum_{j^{(b)}\in\Delta J_i^{(b)}} w_{i,j^{(b)}} \cdot \mu_{i,j^{(b)}}$. The motivation of this weighting method is to make observations with lower observation-errors (lower uncertainty) to be more represented in the final weighted sum. Similarly, this weighting is also used to generate the new observation-times and observation-errors. With this process we merge close observations, reducing the total number of observations in our ZTF dataset by approximately 13 %[2].

3.1.2.3. Sigma Clipping Error Filter

Sigma clipping is used to remove highly uncertain photometric observations from the datasets. As this is related to the observation-error, the sigma clipping is applied to remove observations, from a single-band light-curve $\Phi_i^{(b)}$, with observation-errors $\sigma_{i,j^{(b)}}$ above a

[2] 13.151 %.

threshold of $5\sigma^{(b)}$. The standard deviation $\sigma^{(b)}$ is computed using all the observation-errors from the band b in the training-set $\mathcal{D}_{\text{training}}$.

3.2. Generation of Supernova Synthetic Multi-Band Light-Curves

In this section, a method to generate synthetic multi-band light-curves for SNe, based on the SPM model, is described. Given the scarcity of spectroscopically confirmed SN light-curves in the ZTF survey, we perform this artificial generation to support the training of several Deep Learning models with a stable and well-behaved optimization scenario.

3.2.1. Estimation of the Posterior Distribution of SPM Parameters

To generate a new SN light-curve, a method to sample an optimal and well-behaved set of SPM parameters is required. We use a Bayesian framework to estimate the posterior distribution of the SPM parameters $\boldsymbol{\theta}_i^{(b)}$, given an empirical single-band light-curve $\Phi_i^{(b)}$, as $p\Big(\boldsymbol{\theta}_i^{(b)}|\Phi_i^{(b)}\Big) \propto p\Big(\Phi_i^{(b)}|\boldsymbol{\theta}_i^{(b)}\Big)p\Big(\boldsymbol{\theta}_i^{(b)}\Big)$.

In this framework, a correct estimation of the posterior distribution allows us to sample SPM parameters given a set of empirical observations from a single-band light-curve as $\boldsymbol{\theta}_i^{(b)*} \sim p\Big(\boldsymbol{\theta}_i^{(b)}|\Phi_i^{(b)}\Big)$. For estimating this distribution, the Markov Chain Monte Carlo (MCMC) Ensemble Samplers algorithm (Goodman & Weare, 2010) is used. To estimate the distribution with the MCMC algorithm, given an arbitrary single-band light-curve $\Phi_i^{(b)}$, the likelihood and prior distributions are defined as follows:

$$p\Big(\Phi_i^{(b)}|\boldsymbol{\theta}_i^{(b)}\Big) = \prod_{j^{(b)}=1}^{L_i^{(b)}} \frac{1}{\sigma'_{i,j^{(b)}}\sqrt{2\pi}} \exp\left(\frac{-1}{2\sigma'^2_{i,j^{(b)}}}\Big(\mu_{i,j^{(b)}} - f_{\text{sne}}\Big(t;\boldsymbol{\theta}_i^{(b)}\Big)\Big)^2\right), \tag{3.2}$$

$$p\Big(\boldsymbol{\theta}_i^{(b)}\Big) = \mathcal{N}\Big(\boldsymbol{\theta}_i^{(b)};\boldsymbol{\theta}_i^{(b'\neq b)},\boldsymbol{\Sigma}\Big). \tag{3.3}$$

The explanation for the choices made above are the following:

1. **Likelihood**: in eq. (3.2), the likelihood distribution is defined. This formulation is based on the assumption of a Gaussian distribution for the empirical observation-fluxes $\mu_{i,j^{(b)}}$, where the standard deviation is proportional to the empirical observation-errors $\sigma_{i,j^{(b)}}$. The standard deviation is defined as $\sigma'_{i,j^{(b)}} = \gamma \cdot \sigma^2_{i,j^{(b)}} + \beta$, where $\gamma = 10$ and $\beta = 1$ are empirically selected to adjust the influence of the observation-error in the standard deviation. This setting seeks to prevent that observations with extremely low observation-errors completely control the likelihood of the light-curve.

As mentioned above, γ and β were empirically selected, searching for a correct optimization of the MCMC algorithm. The most important parameter is β as it helps to avoid explosive values for the function cost, during the optimization, when the observation-errors are zero or close to zero, i.e., β allows us to induce a systematic error in the optimization. In such cases, and without β ($\beta = 0$), there may exist a risk of biasing the optimization toward the observations with observation-errors close to zero, possibly causing the undesired effect of the model ignoring observations with higher observation-errors. The term γ allows us to increase or decrease the influence of the observation-error on the optimization. A value of γ was selected to obtain an acceptable diversity of the generated synthetic light-curves after the MCMC optimization. Note that $\gamma = 0$ transforms the eq. (3.2) into a MSE-like function, completely ignoring the influence of the observation-errors.

2. **Prior**: in eq. (3.3), the prior distribution is defined. This formulation implies that the SPM prior selection is based on the SPM optimal parameters from the companion band $b' \neq b$ of the single-band light-curve $\Phi_i^{(b)}$ (a companion band within the multi-band light-curve Φ_i). The prior is defined as an isotropic multivariate Gaussian distribution centered in the companion band b' optimal SPM parameters, with $\mathbf{\Sigma} \in \mathbb{R}^{6\times6}$ a diagonal matrix for the standard deviation. This prior selection attempts to induce information from the companion band b' in the optimization of the target band b. This might correct the optimization in scenarios where no empirical observation is found from the SN-rise and SN-peak regions from the current band. The optimal SPM parameters from the companion band are found using Maximum Likelihood Estimation (MLE) over the empirical observations (see Appendix §B.2 for details). This prior formulation could be extended, for more than two bands, by using a Gaussian Mixture Model (GMM) as the prior distribution.

   As it is fully detailed in Appendix §B.2, we remark that all the Gaussian distributions used in the prior definition are truncated by the SPM bounds to avoid sampling invalid SPM parameter values during the MCMC optimization. These SPM bounds are designed and defined in order avoid the sampling of non-valid SPM parameters, e.g., some SPM parameters are required to be bounded and/or non-negative values. On the other hand, other prior distributions could be further explored for an extended definition of the prior. For example, distributions such as beta could be used for bounded SPM parameters, and distributions such as log-normal or gamma could be used for non-negative SPM parameters. In this work we opted for a simpler solution, leaving the exploration of better prior distributions out the scope of this thesis. By following this line of research, an improvement of correct convergence of the MCMC algorithm may be reach as a finer selection of prior distributions may facilitate the convergence of the sampler.

3.2.1.1. MCMC parameters and settings

As previously mentioned, the Markov Chain Monte Carlo (MCMC) Ensemble Samplers algorithm (Goodman & Weare, 2010) is used for the estimation of the posterior distribution. For this, the Emcee[3] implementation was used. The, the important parameters that control

[3] `https://emcee.readthedocs.io/en/stable/`.

the optimization process are the following[4]:

1. **Number of chains**: the number of walkers (chains) used in the ensemble. A number of `n_chains` = 24 parallel ensembles running in parallel are used.
2. **Burning period**: the number of initial steps that are discarded from the chains as burn-in. A burning period of `discard` = `n_tune`/`n_chains`, for each one of the parallel ensemble, is selected. The value `n_tune` = 2000 is used.
3. **Number of steps**: the number of steps (MCMC iterations) to run each one of the parallel walkers. A number of `nsteps` = (`n_trace_samples` + `n_tune`)/`n_chains` steps, for each one of the parallel ensemble, is used. Note that the burning period `n_tune` is included in the total number of iterations as these iterations will be subsequently discarded. The value `n_trace_samples` = 480 is used. Note that `n_tune` > `n_trace_samples` as there is a priority to have a high burning period, aiming to lower the impact of the initial condition. In general, `n_trace_samples` can be further increased but with a higher computational cost.
4. **Thinning**: the process to set as valid only a portion of intermediate samples within the chain during the algorithm optimization, aiming to decrease the Markov Chain correlation. A number of `thin_by` = 20 steps is used in order to store and yield every `thin_by` samples for each one of the parallel chains. Note that a trade-off exists: a higher value of `thin_by` may help to decrease the chain correlation but with an increase of the computational cost and time.

By using the configuration above, a number of `n_trace_samples`/`n_chains` = 20 iterations are obtained for each one of the parallel chains. Then, the result for all chains are also merged together, giving us a new final chain with `n_trace_samples`/`n_chains` $\cdot$ `n_chains` = 480 samples for the estimated posterior distribution. If a sub-set of k_s samples are required, then the last k_s samples can be selected from the final chain. Although the number of samples per chain (20) may seem rather low; in fact, there are the following aspects to consider: 1) There is a high number of parallel samplers. 2) There is a long burning period; therefore, we aim to decrease the sensibility to initial conditions. 3) There is a high thinning value: we are only storing a part (`n_trace_samples`) of the total number of effective iterations. In fact, the algorithm is internally computing almost `thin_by` times more iterations than the ones that we are actually saving and using.

In general, it is difficult to ensure convergence for the MCMC algorithm for each one of the possible light-curves in the dataset. Therefore, possible MCMC algorithm exceptions and divergences may appear for some exceptional cases. Whenever an MCMC chain does not converge, an alternative and simpler solution is used instead. This solution consists of replacing the SPM analytical function f_{sne} with a linear interpolation between empirical observations, i.e., the existence of a continuous-time function is assumed as the connection of two consecutive observations. Note that we are not explicitly verifying a proper convergence of the MCMC algorithm, instead, we are trying to solve conflict cases where the convergence is not reached. A more elaborate verification of the MCMC convergence is beyond the scope of this thesis.

[4] `https://emcee.readthedocs.io/en/stable/user/sampler/`.

3.2.2. Sampling Time Window

To evaluate the SPM analytical function at different observation-times, a Sampling Time Window (STW), consisting of a collection of L time values, is defined as $\Delta t^{(b)}_{\text{stw}i} = \left\{ t | t \sim \text{U}\left(t^{(b)}_{\text{init}i}, t^{(b)}_{\text{final}i}\right) \right\}_{j=1}^{L}$, where the STW consists of a temporal grid with L time values sampled from a uniform distribution. The size of the STW is defined by the number of empirical observations from the current single-band light-curve $\Phi^{(b)}_i$ ($L = L^{(b)}_i$).

The STW uniform distribution lower bound is defined as follows:

$$t^{(b)}_{\text{init}i} = \begin{cases} t_{i,1^{(b)}}, & \text{if } t_{i,1^{(b)}} < t^{(b)}_{\text{max}i}, \\ t_{i,1^{(b)}} - \Delta t, & \text{otherwise}, \end{cases} \tag{3.4}$$

where $t_{i,1^{(b)}}$ represents the first empirical observation-time from the single-band light-curve $\Phi^{(b)}_i$. The term $t^{(b)}_{\text{max}i}$ is the time value associated with the global maximum of the optimized SPM function evaluated using the optimal SPM parameters $\boldsymbol{\theta}^{(b)*}_i$. By setting $\Delta t = 10$ [days], the STW can be used to sample observation-times before the first empirical observation, e.g., when no observation is available close to the SN-peak (according to the SPM function maximum). This extension allows generating plausible observations over the poorly represented SN-rise and SN-peak regions, originally observed from the dataset $\mathcal{D}$.

The uniform distribution upper bound is defined as $t_{\text{final}i} = t_{i,-1^{(b)}}$, where $t_{i,-1^{(b)}}$ represents the last empirical observation-time from the single-band light-curve $\Phi^{(b)}_i$. This bound ensures that the sampled observation-times are bounded by the last empirical observation-time.

3.2.3. Generation of Synthetic Observations

Given an arbitrary SN multi-band light-curve Φ_i, the process to generate synthetic light-curves is shown in algorithm 3. In addition, Fig. 3.2 shows several examples of synthetic generation of multi-band light-curves for each SN type.

First, a set of optimal SPM parameters $\boldsymbol{\theta}^{(b)*}_i$ are sampled by using an MCMC posterior distribution estimation from the empirical observations. The MCMC sampling procedure allows us to introduce a moderate diversity of the SPM parameters when generating a new light-curve[5]. After this, the STW is generated, and the sampled time values are evaluated using the SPM analytical function f_{sne} and a set of optimal SPM parameters $\boldsymbol{\theta}^{(b)*}_i$. In this way, multiple synthetic observation-fluxes $\hat{\mu}_{i,j^{(b)}}$ are generated.

Next, the synthetic observation-errors are sampled from a conditional distribution $p(\sigma|\hat{\mu}_{i,j^{(b)}}, b)$, which describes the observation-error distribution σ, given an observation-flux $\hat{\mu}$ and a band b. Sampling from this distribution allows generating plausible observation-errors given an arbitrary observation-flux. To estimate this distribution, a collection of Gaussian distribu-

[5]Possible MCMC algorithm exceptions and divergences are handled by replacing the SPM analytical function f_{sne} with a linear interpolation between empirical observations. This strategy is also used in the case where there is no other band information.

Algorithm 3: SN multi-band synthetic light-curve generation based on SPM.

repeat
  for $b \in \{1, \ldots, B\}$ **do**
    $\boldsymbol{\theta}_i^{(b)*} \sim p\left(\Phi_i^{(b)} | \boldsymbol{\theta}_i^{(b)}\right) p\left(\boldsymbol{\theta}_i^{(b)}\right)$
    for $t_{i,j^{(b)}} \in \Delta t_{\text{stw}i}^{(b)}$ **do**
      $\hat{\mu}_{i,j^{(b)}} = f_{\text{sne}}\left(t_{i,j^{(b)}}; \boldsymbol{\theta}_i^{(b)*}\right)$
      $\hat{\sigma}_{i,j^{(b)}} \sim p(\sigma | \hat{\mu}_{i,j^{(b)}}, b)$
      $\hat{\mu}_{i,j^{(b)}} \leftarrow \hat{\mu}_{i,j^{(b)}} + k \cdot \hat{\sigma}_{i,j^{(b)}} \cdot \varepsilon, \varepsilon \sim \text{t-student}(\nu)$
  for $b \in \{1, \ldots, B\}$ **do**
    $t_{i,j^{(b)}} \leftarrow t_{i,j^{(b)}} - t_{i,1}, \forall j^{(b)}$ // `Observation-time re-offset`
until A number of k_s new light-curves are generated from Φ_i

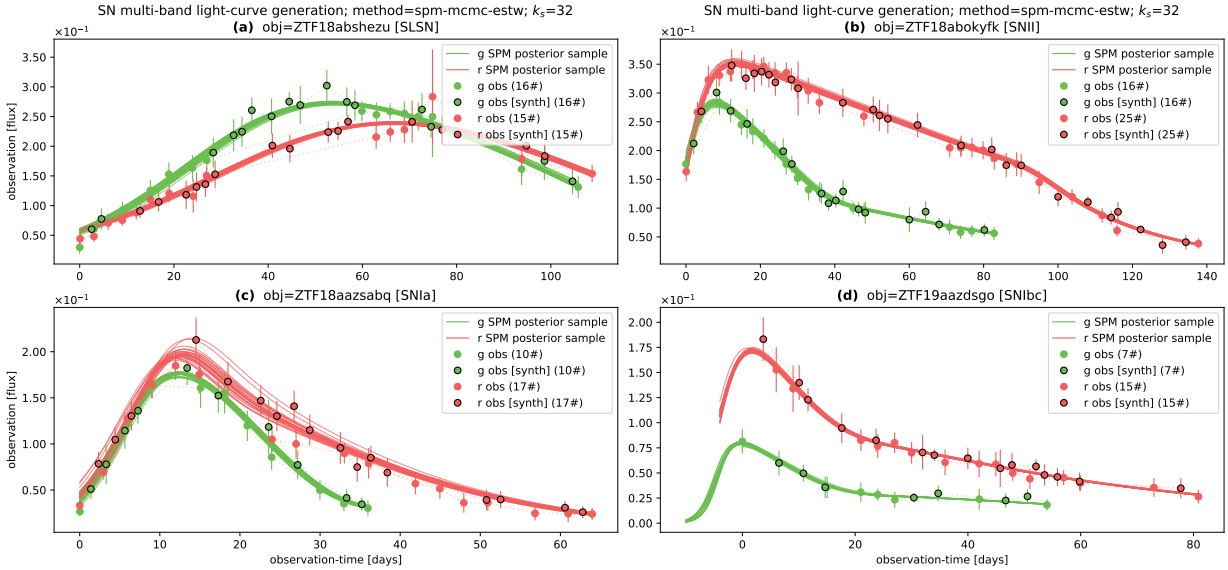

Figure 3.2: Examples of synthetic SN multi-band light-curves (before the observation-time re-offset). The SPM posterior samples ($k_s = 32$) from MCMC are shown as continuous-time curves for each band. Empirical observation-fluxes are shown as color circles with observation-error bars. Synthetic observation-fluxes, using a random SPM posterior sample, are shown as black outlined circles. (a) SLSN type. (b) SNII type. (c) SNIa type. (d) SNIbc type.

tions are fitted using a variable binning strategy over a transformed observation-flux versus observation-error distribution from the training-set $\mathcal{D}_{\text{training}}$ (see Appendix §B.3 for details).

A new synthetic observation-flux is then generated by re-sampling the synthetic observation-flux $\hat{\mu}_{i,j^{(b)}}$ by using a clipped t-student distribution scaled by $k \cdot \hat{\sigma}_{i,j^{(b)}}$. Several methods of light-curve observation-flux re-sampling have been explored in the literature using the Gaussian distribution (Moss, 2018; Naul et al., 2018; Gómez et al., 2020; Hosenie et al., 2020), but in this work, we explore the use of the t-student distribution as it shows higher dispersion over the distribution tails, generating a higher proportion of outlier observation-fluxes along the light-curve. The Gaussian distribution can be recovered by increasing $\nu \to \infty$, where ν

is the degree of freedom of the t-student distribution. For our dataset $\mathcal{D}$, we set $k = 5\text{e}{-1}$ and $\nu = 2$ to obtain a general well-behaved re-sampling dispersion.

3.2.4. Synthetic Training-Set Generation

Given an empirical SN multi-band light-curve, the proposed method can generate an arbitrary number k_s of new synthetic light-curves. Therefore, an augmented synthetic training-set $\mathcal{D}_{\text{training[s]}}$ is built by generating a number of $k_s = 32$ new synthetic light-curves for each empirical light-curve from the original training-set $\mathcal{D}_{\text{training}}$. The construction of this new synthetic training-set $\mathcal{D}_{\text{training[s]}}$ is an effort to increase both, the total number of samples and the diversity of the irregular cadence population observed in the original training-set $\mathcal{D}_{\text{training}}$.

3.3. BRF Baseline Classifier

In this section, the Balanced Random Forest (BRF) model is described as a baseline classifier. This model uses a set of features extracted from the light-curves to classify different types of SNe.

3.3.1. Photometric and Astrophysical Features

Given a multi-band light-curve Φ_i, several features can be extracted using irregular time-series related methods and astrophysical knowledge. These features aim to characterize the general behavior of a variable-length multi-band light-curve into a fixed-length feature vector. For the feature extraction, photometric and astrophysical features implemented by the ALeRCE broker[6] (Sánchez-Sáez et al., 2021) are used, which consist of a collection of 152 photometric features that are computed from light-curves. The ALeRCE broker proposed a vast set of novel features, but it also collected features from previous works (Nun et al., 2015). For instance, the ALeRCE broker proposed to fit the SPM parameters, based on an MLE estimation, as a novel approach to characterize SN light-curves.

In this work, we exclude some metadata-based features such as the ALLWISE colors or the galactic coordinates features, as these features seem to not influence the performance of the transient classifier as reported by the ALeRCE team (Sánchez-Sáez et al., 2021). Extra transient features, such as the Star Galaxy Separation score (SGS score) metadata or non-detection features, are not used because we aim to classify SNe based solely on the detected photometric information, i.e., the light-curves. This gives us a total of 144 features computed from each SN multi-band light-curve. The observation-fluxes, used for our methods, are consequently transformed into apparent magnitude to correctly compute the features.

[6] `https://github.com/alercebroker/lc_classifier`.

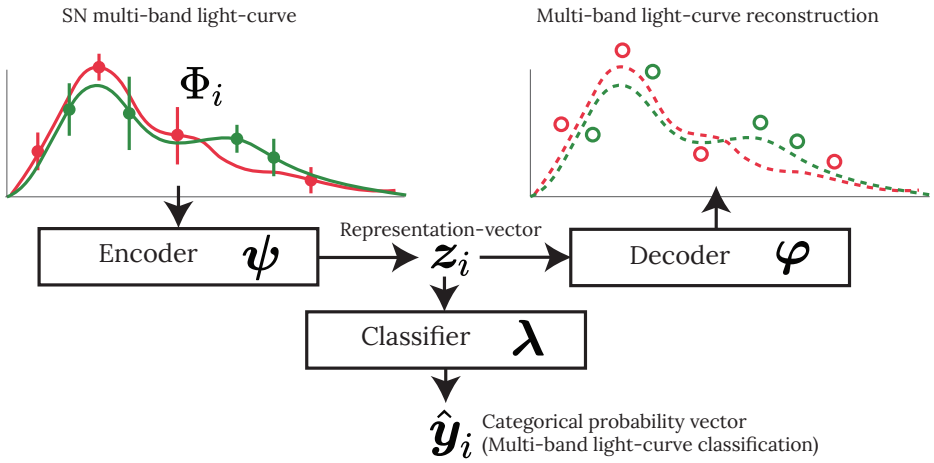

Figure 3.3: Proposed model architecture based on an autoencoder $\{\psi, \varphi\}$, which is composed of an encoder ψ and a decoder φ. The representation-vector z_i serves as input to a classifier λ.

3.3.2. Balanced Random Forest

For the baseline classifier, the Balanced Random Forest model (BRF; Chen et al., 2004) is used, which is a variation of the original Random Forest model (RF; Breiman, 2001). The main advantage of the BRF algorithm is that it can deal with the high class imbalance of the training-set $\mathcal{D}_{\text{training}}$. To deal with the imbalance, the BRF train each decision tree with a bootstrapped sample that is balanced in class samples, where the minority class is potentially well-represented.

To train and test the BRF model, the photometric features are computed from all the SN multi-band light-curves. Infinite and NaN features, which are produced due to incorrect feature extraction and programming code exceptions, are replaced with an especial value of -999 (Sánchez-Sáez et al., 2021). In addition, the best hyperparameter configuration is found by using a grid search over different values for the split quality criterion (e.g., gini, entropy) and tree maximum depth. The best selected configuration is the one associated with the best performance reported over the validation-set $\mathcal{D}_{\text{val}}$ by monitoring the maximum value of the balanced b-F_1score metric (see Appendix §A.2 for details).

3.4. TimeModAttn Model

In this section, the proposed TimeModAttn model for the SN multi-band light-curve classification is described. As shown in Fig. 3.3, this model is based on an autoencoder (encoder-decoder) and a classifier. A more detailed diagram is illustrated in Fig. 3.4. Note that this model can be used to process arbitrary multi-band light-curves; therefore, it is not limited to SN light-curves.

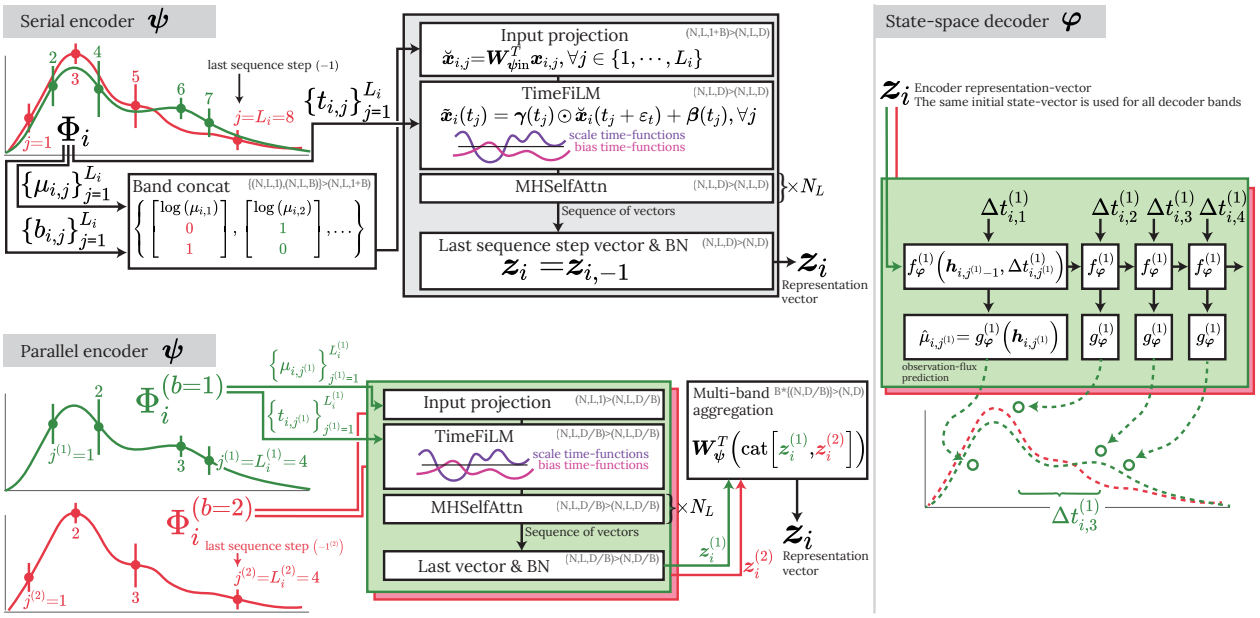

Figure 3.4: Diagram for the proposed autoencoder model (TimeModAttn), which is composed of an encoder $\boldsymbol{\psi}$ (shown on the left side of the diagram) and a decoder $\boldsymbol{\varphi}$ (shown on the right side of the diagram). There are two variations for the encoder: serial and parallel. For the band operator, the notation $\Phi^{(b=1)}$ is used for the band g, and $\Phi^{(b=2)}$ is used for the band r. An arbitrary number of N_L stacked layers is shown for the MHSelfAttn multi-layer sequence processing. Optional tensor dimensional information is also shown (e.g., (N, L, D)), which is related to the model implementation. N stands for the mini-batch size, L for the sequence steps tensor dimension, and D for the representation-vector dimension.

3.4.1. Loss Functions

First, given a multi-band light-curve Φ_i, a Mean Squared Error (MSE) reconstruction loss function is defined as follows:

$$\mathcal{L}_{\text{rec}_i} = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{L_i^{(b)}}\sum_{j^{(b)}=1}^{L_i^{(b)}}\left(\mu_{i,j^{(b)}} - \hat{\mu}_{i,j^{(b)}}\right)^2, \tag{3.5}$$

where $\mu_{i,j^{(b)}}$ are the observation-fluxes from the single-band light-curve $\Phi_i^{(b)}$ (with variable-length $L_i^{(b)}$). The observation-fluxes predicted by the decoder, for the single-band light-curve $\Phi_i^{(b)}$, are denoted as $\hat{\mu}_{i,j^{(b)}}$. Note that the final value of the loss consist of an average of the reconstruction losses from all the B bands[7].

Second, given a multi-band light-curve Φ_i and based on the Cross-Entropy (CE) $H(p|q)$, a categorical loss function is defined as follows:

$$\mathcal{L}_{\text{cat}_i} = H(p(y_i|\Phi_i), q(y_i|\Phi_i)), \tag{3.6}$$

[7]During the preparation of this thesis, we thought that we were using another definition for the reconstruction loss that was based on the Weighted MSE (WMSE) loss. After an inspection of our implementation, we finally decided that the shown loss (MSE) was the most theoretically correct way to define the used reconstruction loss. For more details regarding this issue, see Appendix §B.4

$$= \sum_{c=1}^{C} -p_c(y_i|\Phi_i) \cdot \log\left(\hat{y}_{i,c}\right), \tag{3.7}$$

where C is the total number of classes presented in the dataset, $p(y_i|\Phi_i)$ is the true class distribution, $q(y_i|\Phi_i)$ is the model estimated class distribution, and $\hat{y}_{i,c}$ is the model estimated probability for class c.

3.4.2. Proposed Model Formulation

To model the loss functions defined above, we use an autoencoder model (encoder-decoder). Following Jamal & Bloom (2020), two main architectures are implemented to deal with the multi-band light-curve processing: serial encoder and parallel encoder.

3.4.3. Serial Encoder

The main goal of the encoder is to generate an automatic fixed-length representation-vector $\boldsymbol{z}_i$ from the variable-length multi-band light-curve Φ_i as shown in Fig. 3.3. For the serial encoder $\boldsymbol{\psi}$, the formulation is as follows:

$$\breve{\boldsymbol{x}}_{i,j} = \boldsymbol{W}_{\boldsymbol{\psi}\text{in}}^{T} \boldsymbol{x}_{i,j}, \forall j \in \{1, \ldots, L_i\}, \tag{3.8}$$

$$\boldsymbol{z}_{i,j} = f_{\boldsymbol{\psi}}\Big(\{(\breve{\boldsymbol{x}}_{i,j'}, t_{i,j'})\}_{j' \leq j}\Big), \forall j \in \{1, \ldots, L_i\}, \tag{3.9}$$

$$\boldsymbol{z}_i = \boldsymbol{z}_{i,-1}, \tag{3.10}$$

where $\boldsymbol{x}_{i,j}$ is the encoder input vector, which is composed of photometric observations[8] that are associated with the arbitrary sequence step j. For the model input, the logarithm of the observation-flux is used as $\boldsymbol{x}_{i,j} = \left[\log\left(\mu_{i,j} + \varepsilon\right)\right]^T$. The logarithm function is used to attenuate large observation-flux values from the SN-peak, helping in the normalization of the input distribution for the Deep Learning models[9]. Note that no explicit time information is included in the input vector as the TimeModAttn model uses a temporal modulation strategy (see details in section §3.4.3.1).

For the serial encoder, a one-hot vector, which is defined as $\boldsymbol{b}_{i,j} = \left[0, \ldots, 1_{b_{i,j}=b}, \ldots, 0\right]^T \in \mathbb{R}^B$, is used as a band indicator and concatenated with the encoder input. This operation produces a new encoder input vector $\boldsymbol{x}_{i,j} \in \mathbb{R}^{(1+B)}$. Then, the input vector is projected into a higher $D = 128$ dimensional space $\breve{\boldsymbol{x}}_{i,j} \in \mathbb{R}^D$ by using the linear projection $\boldsymbol{W}_{\boldsymbol{\psi}\text{in}} \in \mathbb{R}^{(1+B)\times D}$, with shared parameters over all sequence steps.

In eq. (3.9), a causal sequence processing formulation is given. For an arbitrary sequence step j, a representation-vector $\boldsymbol{z}_{i,j}$ is computed using the current and previous vectors

[8] All model input vectors are normalized using a standardization method with statistics computed from synthetic training-set $\mathcal{D}_{\text{training[s]}}$ (see Appendix §B.5 for details).

[9] An inverse hyperbolic sine (asinh) function can be used if a forced photometry scenario is presented (with possible negative observation-fluxes).

and observation-times $\{(\breve{\boldsymbol{x}}_{i,1}, t_{i,1}), \ldots, (\breve{\boldsymbol{x}}_{i,j}, t_{i,j})\}$. We propose a temporal modulation (section §3.4.3.1) followed by an attention mechanism (section §3.4.3.2) to solve this formulation. The idea of this setting is to process a time-modulated sequence of representation-vectors by using the attention mechanism.

In eq. (3.10), the final representation-vector from the encoder is defined as the last representation-vector $\boldsymbol{z}_{i,-1}$ from the sequence $\{\boldsymbol{z}_{i,j}\}_{j=1}^{L_i}$. This vector is associated with the last sequence step $j = L_i$, where L_i is the variable-length of the multi-band light-curve Φ_i. Additionally, a Batch Normalization (BN; Ioffe & Szegedy, 2015) operation is performed over the final representation-vector $\boldsymbol{z}_{i,-1}$.

3.4.3.1. Temporal Modulation (TimeFiLM)

A method is required to induce the sequential information in attention mechanisms, as well as the observation-time information, to correctly process and characterize the SN light-curves with highly irregular cadence.

Inspired by the idea of the Feature-wise Linear Modulation operation (FiLM; Perez et al., 2018), we propose a temporal modulation (TimeFiLM) to induce the time information in the input sequence that is processed by the attention mechanism (see Fig. 3.4). The formulation of the proposed temporal modulation is as follows:

$$\tilde{\boldsymbol{x}}_i(t_{i,j}) = \phi_{\tanh}(\boldsymbol{\gamma}(t_{i,j})) \odot \breve{\boldsymbol{x}}_i(t_{i,j} + \varepsilon_t) + \boldsymbol{\beta}(t_{i,j}), \tag{3.11}$$

$$\tilde{\boldsymbol{x}}'_i(t_{i,j}) = \phi_{\mathrm{ReLU}}\big(\boldsymbol{W}^T \tilde{\boldsymbol{x}}_i(t_{i,j}) + \boldsymbol{b}\big), \tag{3.12}$$

$$\gamma_k(t) = \sum_{m=1}^{M} a'_{k,m} \sin\left(\frac{2\pi m}{T_{\max}} t\right) + b'_{k,m} \cos\left(\frac{2\pi m}{T_{\max}} t\right), \tag{3.13}$$

$$\beta_k(t) = \sum_{m=1}^{M} v'_{k,m} \sin\left(\frac{2\pi m}{T_{\max}} t\right) + w'_{k,m} \cos\left(\frac{2\pi m}{T_{\max}} t\right), \tag{3.14}$$

where the vector $\breve{\boldsymbol{x}}_{i,j}$ (associated with the observation-time $t_{i,j}$) stands for the vector time function evaluated on $t_{i,j}$: $\breve{\boldsymbol{x}}_{i,j} \equiv \breve{\boldsymbol{x}}_i(t_{i,j})$. Given an arbitrary input vector time function $\breve{\boldsymbol{x}}_i(t_{i,j} + \varepsilon_t) : \mathbb{R} \mapsto \mathbb{R}^K$, the result of the modulation operation $\tilde{\boldsymbol{x}}_i(t_{i,j}) : \mathbb{R} \mapsto \mathbb{R}^K$, in eq. (3.11), is defined as the element-wise product ($\odot$) followed by the element-wise addition ($+$) (FiLM operation) using the vector time functions $\boldsymbol{\gamma}(t_{i,j})$ and $\boldsymbol{\beta}(t_{i,j})$, respectively. An optional hyperbolic tangent function $\phi_{\tanh}$ is used to prevent explosive product values.

The vector time functions are constructed as $\boldsymbol{\gamma}(t) = [\gamma_1(t), \ldots, \gamma_K(t)]^T$ and $\boldsymbol{\beta}(t) = [\beta_1(t), \ldots, \beta_K(t)]^T$, where $\gamma_k(t) : \mathbb{R} \mapsto \mathbb{R}$ and $\beta_k(t) : \mathbb{R} \mapsto \mathbb{R}$ are the scale and bias time functions, respectively. We assume that these functions are continuously defined and can be evaluated at any arbitrary time value. In eq. (3.11), a new time-modulated vector function $\tilde{\boldsymbol{x}}_i(t_{i,j})$ is generated from the input vector function by using K different scale and bias time functions, each one associated with one dimension component of the modulator input vector $\breve{\boldsymbol{x}}_{i,j}$ (see Fig. 3.5).

The construction of both time functions, the scale, in eq. (3.13), and bias, in eq. (3.14), is based on a Fourier decomposition with M harmonic components. The term $m = 0$ is

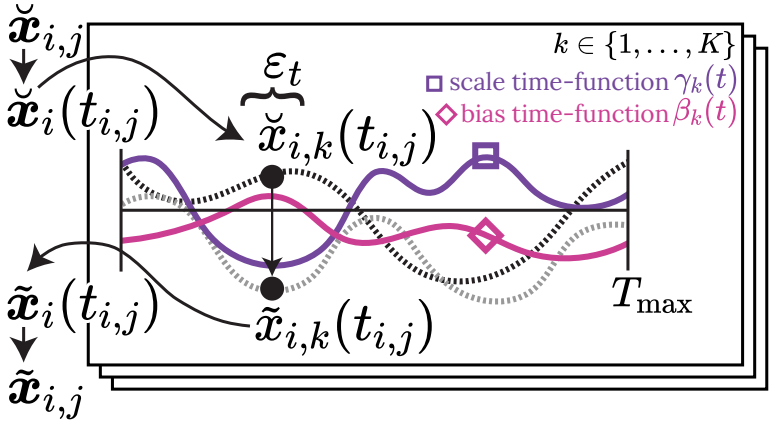

Figure 3.5: Proposed temporal modulation $\tilde{\boldsymbol{x}}_{i,j}$, where $\breve{x}_{i,k}(t_{i,j})$ is the k-th component of the vector time function $\breve{\boldsymbol{x}}_i(t_{i,j})$, which represents the vector $\breve{\boldsymbol{x}}_{i,j}$ associated with the time value $t_{i,j}$. The scale $\gamma_k(t)$ and bias $\beta_k(t)$ functions, represented with solid lines, can be evaluated at any arbitrary continuous-time value, giving a non-redundant and continuous-time modulation-range $[0, T_{\max}]$. Dashed lines represent empirical unobserved time functions, which are associated with the model input.

not included to attenuate the risk of constructing time functions that are invariant in time, which may be produced by significantly high linear term values. Then, the $K \cdot M$ learnable parameters of the modulation are $\left\{\left\{a'_{k,m}, b'_{k,m}, v'_{k,m}, w'_{k,m}\right\}_{k=1}^{K}\right\}_{m=1}^{M}$[10], where $K = D$ is used, corresponding with the dimensionality of the vector $\breve{\boldsymbol{x}}_{i,j}$.

This formulation allows the model to learn arbitrary smooth and continuous-time functions, inducing a non-redundant temporal modulation over a finite time interval $[0, T_{\max}]$, where $T_{\max} = k_T \cdot \max\left\{t_{i,-1}\right\}_{i=1}^{N}$ is defined as the maximum period, associated with the minimum harmonic frequency of the Fourier decomposition. This maximum period is arbitrarily defined such that it exceeds the maximum empirical last observation-time $t_{i,-1}$, found among the light-curves from the training-set $\mathcal{D}_{\text{training}}$, by setting $k_T = 1.5$.

The flexibility to learn any arbitrary time function[11], and not just a collection of periodic functions, as done in (Vaswani et al., 2017; Kazemi et al., 2019; Sousa et al., 2020), might be especially beneficial for non-periodic transient events such as SNe, as there could be potentially more informative time regions in the early explosion days: earlier than and close to the SN-peak, instead of periodically spaced informative zones. Therefore, the TimeModAttn model might learn to induce a highly expressive modulation over those SN time regions. The learning of periodic functions was tested in preliminary experiments, but the collapse of some learned periods was observed, increasing the risk of constructing time-invariant functions. This may be due to the fact that no hidden or intrinsic periodic behavior is expected in the SNe dataset.

[10]The parameters are randomly initialized with uniform Kaiming initialization (He et al., 2015). Also, higher harmonics are attenuated by using an exponential decay $e^{-k\cdot(m-1)}$, $k = .5$, stabilizing the early training epochs by starting with smooth and amplitude controlled modulation curves.

[11]Maximum flexibility and smoothness are constrained by the selected number of M harmonics components used.

As an optional and direct regularization technique, a noise term ε_t is added to each evaluated time value $t_{i,j}$ only during the training process. For the SNe, this noise can be dynamically sampled from a uniform distribution with values between -6 and 6 hours, which can induce a dynamic and moderate disturbance over the original light-curves.

In eq. (3.12), the final modulated vector function $\tilde{\boldsymbol{x}}'_i(t_{i,j})$ is computed using the linear projection $\boldsymbol{W} \in \mathbb{R}^{K\times K}$, plus a bias vector $\boldsymbol{b}$, and the ReLU function ϕ_{ReLU}, with shared parameters over all sequence steps. This operation allows the model to perform nonlinear interactions among the different dimension components of the time-modulated vector.

In summary, the proposed temporal modulation allows the model to properly capture the highly irregular cadence of light-curves by directly using the observation-times to induce an smooth and non-redundant modulation over the time interval $[0, T_{\max}]$. This allows us to avoid using missing-values assumptions, as well as any explicit imputation and interpolation methods. The latter methods might be detrimental because they can introduce artifacts and anomaly light-curve behaviors. Additionally, the construction of continuous-time defined functions using Fourier decomposition allows us to directly inspect the learned functions, exploring for possible and interpretable temporal modulation tendencies produced by the processing of transient events.

3.4.3.2. Multi-Head Self-Attention Mechanism (MHSelfAttn)

To complete the formulation of a solution for the causal sequence processing described in eq. (3.9), we propose the use of a multi-head self-attention (MHSelfAttn) mechanism[12]. This operation is performed over the output sequence vectors $\left\{\tilde{\boldsymbol{x}}'_{i,j}\right\}_{j=1}^{L_i}$ obtained from the temporal modulation method (TimeFiLM). The goal of the attention mechanism is to construct expressive and meaningful context vectors given the query, key, and value vectors as explained in section §2.2.12.

We highlight that the context vectors are computed over a sequence of vectors with induced time information from the proposed temporal modulation (TimeFiLM). Therefore, a meaningful context vector could contain automatic time representations computed along the light-curve, such as time differences, elapsed times, short and long-range trends and time dependencies, among others. This can be achieved by computing the correlations between the time-modulated versions of the query and key vectors, but also with the final vector aggregation using the time-modulated value vectors.

3.4.3.3. Temporal Offsets in Multi-Band Light-curves

Due to the irregular cadence affecting the multi-band light-curves, it is possible that time gaps between the first observations of the different bands are presented (a.k.a. temporal offsets), i.e., it is highly likely that one band starts being observed before or after another band.

[12]For simplicity, a single MHSelfAttn's layer is used ($N_L = 1$). Also, the number of units in the MHSelfAttn's MLP hidden-layer is reduced ($k_{\text{mlp}} = 1$). Note that more MHSelfAttn's layers could be used if required.

Ultimately, this can also be seen as a time shift of one band w.r.t. the other bands. As this offset may provide valuable information about the multi-band behavior of the light-curves, we expect that the proposed model will be able to capture this information automatically.

As it is shown in Fig. 3.3 (serial configuration of the model) and presented in section §2.1, the model input sequence is constructed in such a way that all observations, from all bands, are ordered causally and sequentially (where we use an indicator vector to denote the band information of each observation). This also affects the serial sequence of observation-times used for the model input. Additionally, note that our multi-band light-curves are re-offsetted in time, i.e., the first observation of any multi-band light-curve is forced to occur at the observation-time $t_{i,1} = 0$[13].

Given the proposed configuration of the model input, then any possible time offset between bands is explicitly exposed in the model input. Therefore, the model, ideally, should have access to and capture such information during the optimization thanks to the temporal modulation operation. For example, by assuming that the multi-band observations start in band g (with the corresponding band indicator vector), then the time offset of band r, relative to band g, could be captured as soon as the first band indicator vector associated with band r appears in the model input sequence.

3.4.4. Parallel Encoder

The formulation for the parallel encoder $\boldsymbol{\psi}$ model is as follows:

$$\breve{\boldsymbol{x}}_{i,j^{(b)}} = \boldsymbol{W}_{\boldsymbol{\psi}\text{in}}^{(b)^T} \boldsymbol{x}_{i,j^{(b)}}, \forall j^{(b)} \in \Big\{1, \ldots, L_i^{(b)}\Big\}, \tag{3.15}$$

$$\boldsymbol{z}_{i,j^{(b)}} = f_{\boldsymbol{\psi}}^{(b)}\left(\left\{\left(\breve{\boldsymbol{x}}_{i,j^{(b)'}}, t_{i,j^{(b)'}}\right)\right\}_{j^{(b)'} \leq j^{(b)}}\right), \tag{3.16}$$

$$\forall j^{(b)} \in \Big\{1, \ldots, L_i^{(b)}\Big\},$$

$$\boldsymbol{z}_i^{(b)} = \boldsymbol{z}_{i,-1^{(b)}}, \tag{3.17}$$

$$\boldsymbol{z}_i = \boldsymbol{W}_{\boldsymbol{\psi}}^T\Big(\text{cat}\Big[\boldsymbol{z}_i^{(1)}, \ldots, \boldsymbol{z}_i^{(B)}\Big]\Big), \tag{3.18}$$

where eqs. (3.15)-(3.17) follow similar ideas as those from the serial encoder. In general, in the parallel case, the sequence processing is performed independently for each of the B bands in the multi-band light-curve Φ_i.

In contrast to the serial encoder, in eq. (3.15) the one-hot vector $\boldsymbol{b}_{i,j}$ is not included in the encoder input vector $\boldsymbol{x}_{i,j^{(b)}} = \big[\log\big(\mu_{i,j^{(b)}} + \varepsilon\big)\big]^T$, as it is not required to include the band information for the parallel encoder. The high-dimensional space of the encoder is decreased, from D to (D/B), by using the linear projection $\boldsymbol{W}_{\boldsymbol{\psi}\text{in}}^{(b)} \in \mathbb{R}^{1\times(D/B)}$. This dimensionality reduction is performed in order to have a similar total number of learnable parameters for both, the serial and parallel encoders. Additionally, to keep an equal attention head dimensional space (D_q, D_k, D_v), we decrease the number of attention heads from H to H/B.

[13] As each multi-band light-curve is always re-offsetted in time, then there are no time gaps between light-curves within the dataset $\mathcal{D}$. Therefore, original time gaps between light-curves, due to differences in the starting dates (MJD) of the captured astronomical events, are not a problem given our methodology.

For the final representation-vector note that, in eq. (3.17), the final representation-vector $\boldsymbol{z}_{i,-1^{(b)}}$ is defined as the last representation-vector from the sequence $\left\{\boldsymbol{z}_{i,j^{(b)}}\right\}_{j^{(b)}=1}^{L_i^{(b)}}$. This vector is associated with the last sequence step $j^{(b)} = L_i^{(b)}$, where $L_i^{(b)}$ is the variable-length of the single-band light-curve $\Phi_i^{(b)}$.

In eq. (3.18), the final representation-vector $\boldsymbol{z}_i$ is projected by using the linear projection $\boldsymbol{W}_{\boldsymbol{\psi}} \in \mathbb{R}^{D\times D}$ from the joint space constructed with the concatenation of each of the B band representation-vectors: $\left\{\boldsymbol{z}_i^{(1)}, \ldots, \boldsymbol{z}_i^{(B)}\right\}$. This operation allows the model to capture relevant information, from each band representation-vector, into a final representation-vector with the same number of dimensions D as those of the serial encoder.

3.4.4.1. Temporal Offsets in Multi-Band Light-curves

As previously shown in Fig. 3.3 (parallel configuration of the model) and presented in section §2.1, the original model input sequence (serial configuration) is split into several single-band light-curves (each one associated with one band). Given this parallel processing of the model input sequence, any possible time gap between bands, w.r.t. the first observation-time $t_{i,1} = 0$, of the multi-band light-curve, is properly preserved in the input sequence for each parallel model. Note that no time re-offset is applied to the light-curves after the parallel split is performed, or, otherwise, the offset information would definitely be lost.

As the time offset information is presented in the model input of each parallel sequence processing model, then, ideally, each model should be capable to capture this information thanks to each temporal modulator. This could be possible as the first observation-time, for each single-band light-curve, contains the offset w.r.t. the first observation of the multi-band light-curve. Moreover, the time gap information between bands could be further processed in the last representation-vector that is projected from all the independent band representation-vectors (eq. (3.18)).

3.4.5. Parallel Decoder

The parallel decoder $\boldsymbol{\varphi}$ formulation is based on a state-space model as follows:

$$\boldsymbol{h}_{i,j^{(b)}} = \begin{cases} f_{\boldsymbol{\varphi}}^{(b)}\left(\boldsymbol{z}_i, \Delta t_{i,j^{(b)}}^{(b)}\right) & \text{if } j^{(b)} = 1, \\ f_{\boldsymbol{\varphi}}^{(b)}\left(\boldsymbol{h}_{i,j^{(b)}-1}, \Delta t_{i,j^{(b)}}^{(b)}\right), & \text{otherwise,} \end{cases}, \tag{3.19}$$

$$\hat{\mu}_{i,j^{(b)}} = g_{\boldsymbol{\varphi}}^{(b)}\left(\boldsymbol{h}_{i,j^{(b)}}\right), \tag{3.20}$$

where the initial state-vector $\boldsymbol{h}_{i,1^{(b)}}$ is defined as the representation-vector $\boldsymbol{z}_i$ computed by the encoder $\boldsymbol{\psi}$. In eq. (3.19), the transfer function $f_{\boldsymbol{\varphi}}^{(b)}$ is defined to model the evolution dynamic for the current sequence step state-vector $\boldsymbol{h}_{i,j^{(b)}}$ given both, the previous state-vector $\boldsymbol{h}_{i,j^{(b)}-1}$ and the time difference $\Delta t_{i,j^{(b)}}^{(b)}$ between both observations. In eq. (3.20), the function $g_{\boldsymbol{\varphi}}^{(b)}$ is defined to generate the single-band light-curve observation-flux estimations $\hat{\mu}_{i,j^{(b)}}$ given the current state-vector $\boldsymbol{h}_{i,j^{(b)}}$.

For simplicity and because our major research interest is the expressive capacity of the encoder $\boldsymbol{\psi}$, only the parallel approach is used for the decoder. This approach is also shared with all the encoder variations and baselines implemented in this work. As a remark, the same initial state-vector $\boldsymbol{z}_i$ is used for each of the B parallel decoders, where each decoder is associated with a particular band.

The function $f_{\boldsymbol{\varphi}}^{(b)}$ is modeled with a Gated Recurrent Unit (GRU; Chung et al., 2014), having a 1-dimensional input space for the time difference input and D dimensions for the hidden state-vector. The function $g_{\boldsymbol{\varphi}}^{(b)}$ is modeled with a 1-hidden-layer Multi-Layer Perceptron (MLP; Rumelhart et al., 1986), with a 1-dimensional output space and a linear activation function at the output. This MLP model shared parameters over all sequence steps.

3.4.6. Classifier

The proposed formulation for the classifier $\boldsymbol{\lambda}$ is as follows:

$$\hat{\boldsymbol{y}}_i = \phi_{\text{softmax}}(f_{\boldsymbol{\lambda}}(\boldsymbol{z}_i)), \tag{3.21}$$

where the function $f_{\boldsymbol{\lambda}}$ is modeled using a 2-hidden-layer MLP. The output dimension is set as the number of classes C. The softmax function ϕ_{softmax} is used to generate a final categorical probability vector for the discrete class prediction distribution $\hat{\boldsymbol{y}}_i = [\hat{y}_{i,1}, \ldots, \hat{y}_{i,C}]^T$. A dropout probability (Srivastava et al., 2014) of 50 % is used for the MLP model.

3.4.7. Optimization Problem

Given the aforementioned loss functions and the TimeModAttn model formulation, the complete optimization problem is defined as follows:

$$\mathcal{L}_{\text{pre-training}} = \frac{1}{N}\sum_{i=1}^{N} \underbrace{k_0 \cdot \mathcal{L}_{\text{rec}_i}}_{\substack{\text{multi-band}\\ \text{light-curve}\\ \text{reconstruction}}} + \underbrace{k_1 \cdot \mathcal{L}_{\text{cat}_i}}_{\substack{\text{cross-entropy}\\ \text{regularization}}}, \tag{3.22}$$

$$\mathcal{L}_{\text{fine-tuning}} = \frac{1}{N}\sum_{i=1}^{N} \underbrace{\mathcal{L}_{\text{cat}_i}}_{\text{cross-entropy}}, \tag{3.23}$$

where $k_0 = 1\text{e}4$ and $k_1 = 1$. This setting implies a higher relevance for the reconstruction loss term, i.e., the unsupervised learning term $\mathcal{L}_{\text{rec}_i}$. Here, N is the number of light-curves used to compute the loss functions (mini-batch size). The minimization optimization problem is split into the following two main steps:

1. **Pre-training**: first, the autoencoder $\{\boldsymbol{\psi}, \boldsymbol{\varphi}\}$ is trained to solve an auxiliary multi-band light-curve reconstruction task using a training-set composed of synthetic light-curves from $\mathcal{D}_{\text{training[s]}}$. The encoder $\boldsymbol{\psi}$ computes a fixed-length representation-vector $\boldsymbol{z}_i$ from

the variable-length multi-band light-curve Φ_i. This representation-vector $\boldsymbol{z}_i$ automatically summarize all the relevant aspects of the dynamics of the light-curve in order to estimate the correct light-curve reconstruction using the decoder $\boldsymbol{\varphi}$. The dynamics might include short and long-range trends and time dependencies; significant information about the first days of observations, SN-peak and SN-plateau regions; correlations and differences between bands; among others.

The representation-vector $\boldsymbol{z}_i$ serves as input to the classifier $\boldsymbol{\lambda}$ in order to discriminate among SN types. Additionally, in eq. (3.22), a semi-supervised learning scheme is used by including the labels of the synthetic light-curves on a cross-entropy regularization term over the representation-vector from the encoder. This regularization aims to improve the degree of nonlinear separation of the SN types over the representation space. The aim of the encoder is to automatically generate an informative representation-vector $\boldsymbol{z}_i$ useful for both, a multi-band reconstruction task and a categorical discrimination task.

The pre-training optimization step is performed using the Adam optimizer (Kingma & Ba, 2015) with the following setting: {`params`=$\{\boldsymbol{\psi}, \boldsymbol{\varphi}, \boldsymbol{\lambda}\}$, `batch_size`= 200, `betas`=(.9, .999), `weight_decay`=2e−4}. A linear learning rate warm-up schedule is implemented, increasing the learning rate lr, from $\text{lr}_{\text{min}} = 1\text{e}{-10}$ to $\text{lr}_{\text{max}} = 1.1\text{e}{-3}$, in $\Delta_{\text{epoch}} = 10$ train epochs (Vaswani et al., 2017; Lee et al., 2021).

2. **Fine-tuning**: after the pre-training process, a classification task is solved using a training-set composed only of empirical light-curves from $\mathcal{D}_{\text{training}}$. In this step, no synthetic light-curves are used. The fine-tuning process is performed as a domain adaptation technique, aiming to minimize the model's gaps and discrepancies between the synthetic distribution and the empirical distribution of light-curves. Discrepancies may arise as the synthetic light-curves behavior could be biased towards the SPM's overly smooth behavior, general incorrect SPM parameters fit, inter-band peak time shifts and time differences, among others.

   The fine-tuning optimization step is trained using the Stochastic Gradient Descent optimizer (SGD) with the following setting: {`params`= $\{\boldsymbol{\lambda}\}$, `batch_size`= 50, `lr`= 1e−3, `momentum`= .9}. In the fine-tuning process only the parameters of the classifier $\boldsymbol{\lambda}$ are re-optimized, while freezing the autoencoder parameters $\{\boldsymbol{\psi}, \boldsymbol{\varphi}\}$. Therefore, the encoder is used as a fixed-length representation-vector extractor from the multi-band light-curves.

3.4.7.1. Imbalance Learn and Regularization

To deal with class imbalance, the model is trained using mini-batches sampled from balanced auxiliary training-sets, which are dynamically and randomly constructed at each new training epoch using a stratified bootstrap strategy. This balancing strategy is applied during both optimization steps (see Appendix §B.6 for details).

In addition, as a regularization technique, an early stopping routine is executed during both optimization steps. This technique is performed by evaluating the losses over the imbalanced validation-sets $\mathcal{D}_{\text{val}}$. The losses of each light-curve Φ_i, of class c, are weighted according to the factor $k_c = \frac{1}{N_c \cdot C}$, where N_c is the number of samples from class c and C is the total

number of classes. This procedure allows us to evaluate a kind of balanced loss function over an imbalanced validation-set.

Finally, during both optimization steps, a dynamical data-augmentation procedure is also implemented by introducing observation-flux noise and removing random observations along the multi-band light-curves. For further comparison purposes, we implement three levels of data-augmentation: zero, weak, and strong data-augmentation levels (see Appendix §B.7 for details).

3.4.8. RNN Baseline Models

In addition to the proposed attention-based encoder, baseline models based on Recurrent Neural Networks (RNNs; Rumelhart et al., 1986; Zimmermann et al., 2012) were implemented.

For the serial encoder, two different RNN models are tested: the Long Short-Term Memory (LSTM; Hochreiter & Schmidhuber, 1997) and the Gated Recurrent Unit (GRU; Chung et al., 2014), which results in an alternative formulation for eq. (3.9). Similar to previous works (Naul et al., 2018; Carrasco-Davis et al., 2019; Tsang & Schultz, 2019; Gómez et al., 2020; Becker et al., 2020; Möller & de Boissière, 2020; Tachibana et al., 2020), the encoder input vector $\boldsymbol{x}_{i,j}$, shown in eq. (3.8), is re-defined as $\boldsymbol{x}_{i,j} = [\log(\mu_{i,j} + \varepsilon), \Delta t_{i,j}]^T$, where the time difference term $\Delta t_{i,j}$, for a multi-band light-curve Φ_i, is included. This term aims to describe the irregular cadence information for the encoder to correctly capture relevant time dependencies.

Likewise, for the parallel encoder, eq. (3.16) is modeled with LSTM and GRU models. Additionally, the encoder input vector, described in eq. (3.15), is re-defined to include the time information as $\boldsymbol{x}_{i,j^{(b)}} = \left[\log\left(\mu_{i,j^{(b)}} + \varepsilon\right), \Delta t^{(b)}_{i,j^{(b)}}\right]^T$, where $\Delta t^{(b)}_{i,j^{(b)}}$ is the time difference computed for the single-band light-curve $\Phi_i^{(b)}$.

3.4.8.1. Temporal Offsets in Multi-Band Light-curves

For the sake of fair comparisons between the proposed model and the RNN baselines, it is necessary that the offset information can also be captured by the models without the temporal modulator. As explained above, the RNN baselines use the time differences information, instead of the raw observation-times, as the temporal information. The definition of the time differences is fully described in section §2.1.3.

First, for the serial case, note that the RNN-based model, ideally, would capture the offset information by accumulating the time differences from the first observation of the multi-band light-curve up to the first observation of a particular band (this could be triggered by the band indicator vector). For the parallel case, the offset information of each band is explicit according to the definition proposed in §2.1.3. This definition was selected precisely so that this parallel configuration would not lose the offset information. It should be noted that the same explanation applies to the parallel decoder presented in section §3.4.5 since it also uses

the time difference information as model input.

Chapter 4

Results and Analyses

4.1. Results and Analyses

In this section, the experimental results are presented and the main analyses are performed. The notations S-{} and P-{} stand for the serial and parallel encoders, respectively.

Due to computational cost limitations, all SN light-curve observations beyond a threshold-day of 100 [days] were removed from all sets: $\mathcal{D}_{\text{train[s]}}$, $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, and $\mathcal{D}_{\text{test}}$. We found this threshold-day representative enough to properly study the SN events from the ZTF survey[1].

If not specified otherwise, all results are reported by evaluating the models over the test-set $\mathcal{D}_{\text{test}}$, which is composed of empirical light-curves only. The reported results consist of the aggregation of the results reported from all folds in the stratified 5-fold cross-validation. In addition, six random model's initializations (runs) per fold were performed, giving a total number of $N_{\text{runs}} = 30$ runs per model implementation.

Due to the general high variance found in the results, the p-values (p) from significance statistical tests are also included when necessary, complementing the analysis of results. Given the non-Gaussian distribution observed in the test-set results[2], a non-parametric statistical test is used: the permutation test[3]. We use a threshold of $p < .05$ to denote a statistical significance when comparing differences (Δ) between population means.

All Deep Learning models were implemented on Pytorch 1.8.1 (Paszke et al., 2019), using a GeForce GTX 1080 Ti GPU[4].

[1] Under this consideration, the extended non-redundant modulation-range is $[0, 150]$ [days].

[2] The non-Gaussianity (and high variance) of the aggregated 5-fold cross-validation results is produced due to differences in the reported classification performance among the test-set folds. These performance differences are usually influenced by the high class imbalance, the small number of samples, and the general quality of the light-curves presented in each test-set fold.

[3] `http://rasbt.github.io/mlxtend/user_guide/evaluate/permutation_test/`.

[4] `https://github.com/oscarpimentel/astro-lightcurves-classifier`.

4.1.1. Late-Classification Scenario

First, we report the late-classification performance for all models using multi-band light-curves with a maximum threshold-day of $t_{\text{th}} = 100$ [days], which are called 100-day light-curves in what follows. These light-curves are still of variable-length, where the maximum light-curve length found in this scenario is close to $L_i = 150$.

Table 4.1 shows the balanced metrics (b-{}) for the multi-class classification scenario: Precision, Recall, F_1score, Area Under the Receiver Operating Characteristic Curve (AUCROC), and Area Under the Precision-Recall Curve (AUCPR). These balanced metrics assume that each class is equally important despite the high class imbalance (see Appendix §A.2 for details). Additionally, for comparison purposes, the results for the zero, weak, and strong data-augmentation levels are shown. In the zero data-augmentation level, none of the data-augmentation procedures are used. In the weak level, a probability of 10 % is used to randomly remove observations in the light-curves during training. Consequently, this probability value produces a moderate disturbance in the irregular cadence observed during training. In contrast, in the strong data-augmentation level, a probability of 50 % is used, heavily affecting the cadence observed during training (see Appendix §B.7 for details). The results for a pre-training optimization step, using empirical light-curves from the training-set $\mathcal{D}_{\text{train}}$, are also shown for comparison purposes (empirical pre-training).

A significant performance improvement, w.r.t. the empirical pre-training, can be observed when using the synthetic pre-training settings (for all reported metrics): where the synthetic light-curves are used to perform the pre-processing optimization step. This improvement is achieved by all the tested Deep Learning models for both, the serial and parallel encoders. These results confirm that the use of synthetic light-curves is effectively beneficial to support the optimization of the Deep Learning models (RNN and attention-based models), producing a higher general performance than the BRF baseline performance (without synthetic data).

For all the synthetic pre-training settings, it can be observed that the proposed TimeModAttn model outperformed the BRF baseline (for all reported metrics). In particular, we highlight the weak data-augmentation level, where the following are the metric mean's differences, w.r.t. the BRF baseline, for the serial encoder: Δb-Precision=.0611***, Δb-Recall=.0719***, Δb-F_1score= .0703***, Δb-AUCROC=.0437***, and Δb-AUCPR= .0691***[5]. For the parallel encoder, the metric differences are as follows: Δb-Precision=.0530***, Δb-Recall=.0654***, Δb-F_1score= .0687***, Δb-AUCROC=.0446***, and Δb-AUCPR= .0865***. Additionally, no strong or consistent statistical evidence was found to conclude that either, the serial encoder or the parallel encoder, is the best alternative ($p \in [.021, .403]$[6]), implying that the type of encoder may be irrelevant in terms of general performance for the TimeModAttn model.

It can be observed that the level of the data-augmentation affected the performance of the RNN baselines for both the serial and parallel encoders. The strong data-augmentation level was detrimental to the performance of the RNN baselines. This effect may be explained because the data-augmentation dynamically influences the number of observations in the

[5] The statistical significance notation used is as follows:
$^{***}p \leq .001$, $^{**}p \leq .01$, $^{*}p \leq .05$, and $^{+}p \leq .1$.

[6] $p \in \{.021, .206, .403, .391, .036\}$.

Table 4.1: Late-classification performances for the BRF baseline, RNN baselines, and attention-based models (TimeModAttn) using 100-day multi-band light-curves. Both, the serial (S-{}) and parallel (P-{}) encoders are reported along with several pre-training and data-augmentation schemes (mean±std from 5-fold cross-validation).

Model	b-Precision	b-Recall	b-F_1score	b-AUCROC	b-AUCPR
BRF (fmode=all; training-set=[r])	.527±.030	.687±.052	.525±.039	.866±.020	.602±.051
Serial models	Empirical pre-training (zero data-augmentation)				
S-RNN+Δt (cell=GRU)	.520±.043	.626±.050	.528±.039	.852±.021	.577±.049
S-RNN+Δt (cell=LSTM)	.497±.030	.602±.044	.502±.034	.840±.019	.568±.031
S-TimeModAttn (M=12; H=8; ε_t=6/24)	.551±.034	.664±.058	.565±.040	.874±.024	.597±.036
	Synthetic pre-training (zero data-augmentation)				
S-RNN+Δt (cell=GRU)	.562±.051	.688±.058	.579±.049	.885±.036	.627±.062
S-RNN+Δt (cell=LSTM)	.561±.035	.680±.053	.578±.040	.884±.028	.619±.046
S-TimeModAttn (M=12; H=8; ε_t=6/24)	**.598±.030**	.736±.056	**.614±.036**	.904±.029	.665±.060
	Synthetic pre-training (weak data-augmentation)				
S-RNN+Δt (cell=GRU)	.545±.034	.706±.070	.556±.045	.879±.034	.610±.066
S-RNN+Δt (cell=LSTM)	.550±.031	.711±.070	.558±.040	.887±.033	.621±.070
S-TimeModAttn (M=12; H=8; ε_t=6/24)	.588±.023	**.759±.040**	.596±.033	.910±.020	**.671±.056**
	Synthetic pre-training (strong data-augmentation)				
S-RNN+Δt (cell=GRU)	.491±.024	.649±.068	.496±.036	.860±.032	.561±.063
S-RNN+Δt (cell=LSTM)	.497±.021	.657±.066	.494±.028	.864±.031	.565±.055
S-TimeModAttn (M=12; H=8; ε_t=6/24)	.582±.017	.754±.039	.584±.031	**.911±.019**	.665±.053
Parallel models	Empirical pre-training (zero data-augmentation)				
P-RNN+Δt (cell=GRU)	.521±.042	.613±.042	.527±.044	.849±.013	.561±.032
P-RNN+Δt (cell=LSTM)	.497±.034	.604±.049	.500±.041	.834±.016	.548±.027
P-TimeModAttn (M=12; H=4; ε_t=6/24)	.543±.026	.671±.053	.562±.029	.865±.022	.599±.038
	Synthetic pre-training (zero data-augmentation)				
P-RNN+Δt (cell=GRU)	.566±.038	.685±.056	.582±.040	.883±.027	.624±.047
P-RNN+Δt (cell=LSTM)	.567±.029	.683±.041	.580±.036	.881±.027	.645±.048
P-TimeModAttn (M=12; H=4; ε_t=6/24)	**.591±.021**	.729±.038	**.610±.026**	.897±.023	.676±.059
	Synthetic pre-training (weak data-augmentation)				
P-RNN+Δt (cell=GRU)	.547±.030	.697±.070	.552±.041	.879±.031	.610±.055
P-RNN+Δt (cell=LSTM)	.541±.022	.704±.061	.540±.032	.876±.029	.606±.051
P-TimeModAttn (M=12; H=4; ε_t=6/24)	.580±.020	**.753±.044**	.594±.035	**.911±.017**	**.689±.047**
	Synthetic pre-training (strong data-augmentation)				
P-RNN+Δt (cell=GRU)	.490±.020	.645±.057	.482±.024	.856±.032	.577±.064
P-RNN+Δt (cell=LSTM)	.499±.020	.660±.061	.484±.031	.857±.031	.573±.053
P-TimeModAttn (M=12; H=4; ε_t=6/24)	.581±.019	.750±.039	.585±.036	.907±.016	.679±.043

light-curves during the model optimization, directly affecting the values of the computed time differences $\Delta t_{i,j}$: the higher the probability of removing observations, the longer the computed time differences. Therefore, the data-augmentation produces a discrepancy between the time difference distributions of the training-set and the test-set, which may lead to a poor model generalization for unobserved light-curves presented in the test-set. Note that the maximum discrepancy between these distributions arises in the strong data-augmentation level, where

the worst performance was reported for the RNN baselines.

In contrast, high robustness against different data-augmentation levels can be observed for the TimeModAttn model. This could be explained because the encoder in the TimeModAttn model (TimeFiLM) directly uses the observation-times, without just relying on the time difference values. Therefore, the time representation used in the TimeModAttn model might be less sensitive to the general irregularity of the cadence, achieving a higher degree of model generalization for the unobserved light-curves in the test-set. This effect can be observed in the reported performances in Table 4.1 where, for all the tested data-augmentation levels and all reported metrics, the TimeModAttn model achieved higher performance than the RNN baselines. Note that the TimeModAttn model achieved high classification performances even in the strong data-augmentation level.

For further comparisons, we analyze the zero data-augmentation level because the RNN baselines achieved the best general performance in that setting. The TimeModAttn model outperforms the GRU baseline for both, the serial encoder ($p \leq .001, \forall p$) and the parallel encoder ($p \leq .001, \forall p$). Similar statistical evidence was also found when comparing the TimeModAttn model w.r.t. the LSTM baseline for both, the serial encoder ($p \leq .001, \forall p$) and the parallel encoder ($p \leq .001, \forall p$). Additionally, no strong or consistent statistical significance was found when comparing the GRU and LSTM baselines for both, the serial encoder ($p \in [.212, .459]$[7]) and the parallel encoder ($p \in [.009, .482]$[8]). Therefore, no major difference exists between both RNN models in the context of this work.

4.1.2. Early-Classification Scenario

Herein we study the case when a higher number of observations is gradually available in the test-set $\mathcal{D}_{\text{test}}$. With this aim, a moving threshold-day $t_{\text{th}} \in [1, 100]$ [days] is used in order to remove all observations, from test-set $\mathcal{D}_{\text{test}}$, above a given threshold. Note that if $t_{\text{th}} = 100$ [days]; then, the test-set $\mathcal{D}_{\text{test}}$ is equivalent to the set used in the late-classification scenario. For the feature extraction, algorithm instabilities arise when just a single observation is used to fit the SPM model. Therefore, for the BRF baseline, the results start being reported only since a minimum number of observations is reached: when all light-curves in the test-set $\mathcal{D}_{\text{test}}$ have at least one band with a number equal or higher than $L_i^{(b)} \geq 2$ observations.

For example, Fig. 4.1 shows the evolution of the b-AUCROC metric as a function of a moving threshold-day t_{th} in the weak data-augmentation level. As expected, the general performance for this metric increased with larger threshold-days. This is because the models have access to longer light-curves; hence, more information about the evolution of the SN transient event. From the b-AUCROC curves, we also observe that the performance of the TimeModAttn model tended to be higher than the rest of the tested baselines for most of the operation points. In particular, the TimeModAttn model achieved the BRF's maximum b-AUCROC performance several days earlier ($t_{\text{th}} \in (32, 40)$ [days]) than the BRF baseline ($t_{\text{th}} \in (52, 60)$ [days]). This result indicates that the TimeModAttn model can discriminate between SN types using light-curves with fewer observations than those used by the BRF

[7] $p \in \{.453, .212, .459, .440, .215\}$.
[8] $p \in \{.482, .447, .380, .370, .009\}$.

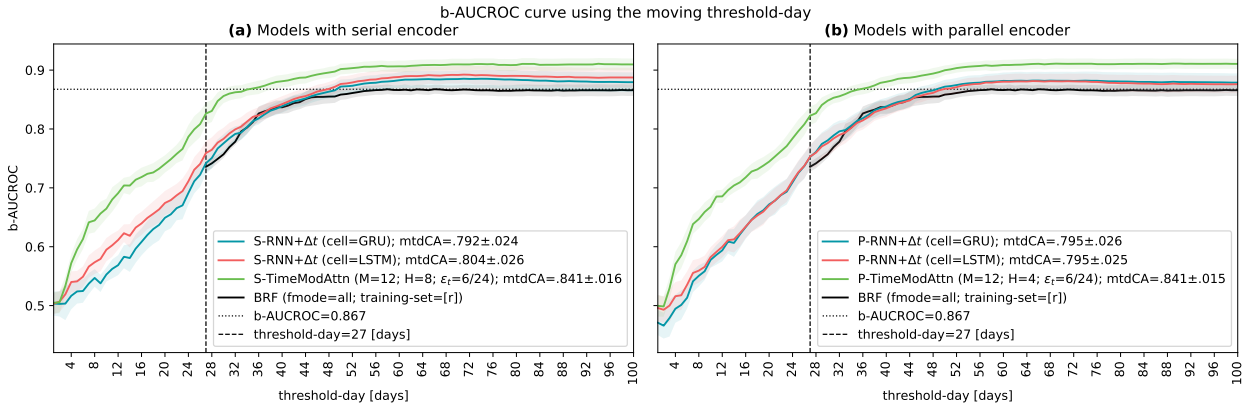

Figure 4.1: b-AUCROC metric-curve v/s moving threshold-day for the attention-based models, BRF baseline, and RNN baselines in the weak data-augmentation level (for the sake of better visualization, mean$\pm\frac{1}{2}$std curve and region are shown from 5-fold cross-validation). The moving threshold-day Curve Average (mtdCA) is shown for the TimeModAttn model and RNN baselines. Note that the horizontal axis (evolution of the threshold-day) relates with the observation-time since the first observation. (a) Models with serial encoder. (b) Models with parallel encoder.

baseline.

To summarize the early-classification results, the Curve Average (CA) is computed for the curves constructed by using the moving threshold-day (mtdCA). High values of the mtdCA are achieved if the performance of the model is consistently high along all the operation points defined by the moving threshold-day. Table 4.2 shows the computed mtdCA for all the reported balanced metrics.

Our findings in the early-classification scenario follow a similar trend as those previously reported in the late-classification scenario. As before, the use of synthetic light-curves (synthetic pre-training) was beneficial for all the tested Deep Learning models. Also, no strong or consistent strong statistical evidence was found to conclude which encoder alternative is the best (serial or parallel) for the TimeModAttn model ($p \in [.022, .458]$[9]) in the weak data-augmentation level.

For all the different pre-training settings and all reported metrics, the TimeModAttn model achieved higher performance than the RNN baselines. When comparing performances of the TimeModAttn model against the RNN baselines (GRU and LSTM), in the zero data-augmentation level (best late-classification setting for the RNN baselines), a significant difference was found for both, the serial encoder ($p \leq .001, \forall p$) and the parallel encoder ($p \leq .001, \forall p$). These results, along with Fig. 4.1, indicate that the TimeModAttn model outperforms the RNN baselines in the early-classification scenario of light-curves with few observations.

As before, general robustness against the level of the data-augmentation was again observed for the TimeModAttn model. Furthermore, the results shown in Table 4.2 suggest that the use of data-augmentation could be beneficial in the early-classification performance for

[9] $p \in \{.022, .034, .164, .458, .118\}$.

Table 4.2: Early-classification performances for the RNN baselines and attention-based models (TimeModAttn). The moving threshold-day Curve Average (mtdCA) is used (‡). Both, the serial (S-{}) and parallel (P-{}) encoders are reported along with several pre-training and data-augmentation schemes (mean±std from 5-fold cross-validation).

Model	b-Precision‡	b-Recall‡	b-F_1score‡	b-AUCROC‡	b-AUCPR‡
Serial models	Empirical pre-training (zero data-augmentation)				
S-RNN+Δt (cell=GRU)	.427±.025	.491±.029	.404±.034	.745±.020	.461±.022
S-RNN+Δt (cell=LSTM)	.412±.021	.489±.028	.395±.026	.749±.018	.450±.028
S-TimeModAttn (M=12; H=8; ε_t=6/24)	.474±.025	.535±.033	.448±.030	.806±.023	.509±.028
	Synthetic pre-training (zero data-augmentation)				
S-RNN+Δt (cell=GRU)	.471±.033	.536±.036	.440±.036	.780±.029	.505±.038
S-RNN+Δt (cell=LSTM)	.481±.035	.559±.035	.459±.042	.797±.024	.518±.038
S-TimeModAttn (M=12; H=8; ε_t=6/24)	.516±.022	.608±.033	**.497±.022**	.832±.024	.562±.038
	Synthetic pre-training (weak data-augmentation)				
S-RNN+Δt (cell=GRU)	.481±.030	.577±.044	.454±.031	.792±.024	.520±.039
S-RNN+Δt (cell=LSTM)	.480±.023	.590±.036	.457±.027	.804±.026	.527±.043
S-TimeModAttn (M=12; H=8; ε_t=6/24)	**.522±.022**	.630±.026	.495±.020	.841±.016	**.580±.040**
	Synthetic pre-training (strong data-augmentation)				
S-RNN+Δt (cell=GRU)	.447±.023	.556±.036	.430±.026	.783±.020	.491±.033
S-RNN+Δt (cell=LSTM)	.445±.019	.567±.035	.422±.021	.790±.021	.492±.031
S-TimeModAttn (M=12; H=8; ε_t=6/24)	**.522±.018**	**.632±.023**	.493±.020	**.845±.012**	.579±.039
Parallel models	Empirical pre-training (zero data-augmentation)				
P-RNN+Δt (cell=GRU)	.447±.028	.501±.029	.420±.030	.762±.018	.471±.018
P-RNN+Δt (cell=LSTM)	.422±.026	.494±.024	.400±.033	.749±.026	.458±.026
P-TimeModAttn (M=12; H=4; ε_t=6/24)	.461±.023	.516±.026	.438±.022	.789±.021	.495±.023
	Synthetic pre-training (zero data-augmentation)				
P-RNN+Δt (cell=GRU)	.492±.032	.549±.034	.464±.033	.788±.026	.519±.033
P-RNN+Δt (cell=LSTM)	.488±.025	.552±.024	.461±.029	.791±.030	.530±.034
P-TimeModAttn (M=12; H=4; ε_t=6/24)	**.516±.017**	.597±.018	**.499±.018**	.826±.020	.565±.027
	Synthetic pre-training (weak data-augmentation)				
P-RNN+Δt (cell=GRU)	.485±.023	.580±.041	.462±.028	.795±.026	.524±.040
P-RNN+Δt (cell=LSTM)	.476±.018	.586±.036	.451±.024	.795±.025	.516±.034
P-TimeModAttn (M=12; H=4; ε_t=6/24)	.514±.018	.621±.027	**.499±.019**	**.841±.015**	**.587±.029**
	Synthetic pre-training (strong data-augmentation)				
P-RNN+Δt (cell=GRU)	.440±.016	.561±.045	.422±.022	.777±.023	.496±.039
P-RNN+Δt (cell=LSTM)	.442±.016	.564±.043	.416±.018	.782±.020	.493±.032
P-TimeModAttn (M=12; H=4; ε_t=6/24)	.515±.015	**.623±.022**	.495±.020	**.841±.014**	.582±.034

the TimeModAttn model.

The above results show that the performance of the TimeModAttn model is not only higher in the late-classification scenario, but it is also consistently higher along with different early-classification operation points that are defined by changing the moving threshold-day. Moreover, the early-classification performance of the TimeModAttn model was higher than all the other tested baselines.

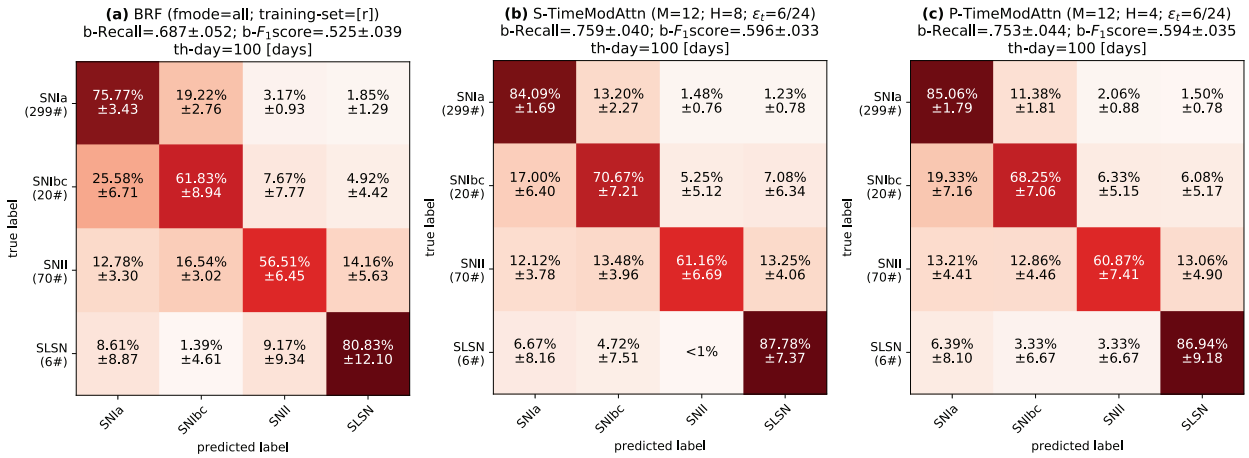

Figure 4.2: Confusion matrices for the SNe classification task using 100-day multi-band light-curves (mean±std from 5-fold cross-validation). The corresponding b-Recall and b-F_1score metrics are given on the top of each matrix. (a) BRF baseline model. (b) S-TimeModAttn model. (c) P-TimeModAttn model.

If not specified otherwise, the weak data-augmentation level is selected and explored as the main pre-training setting for the following experiments in this work. Additionally, examples of SN multi-band light-curve reconstructions can be found in Appendix §C.4 for the weak data-augmentation level.

4.1.3. Confusion Matrices and Operational Curves

Fig. 4.2 shows the confusion matrices for the classification of SN multi-band light-curves. As previously reported in the literature (Moss, 2018; Villar et al., 2019; Sánchez-Sáez et al., 2021), we can observe a common confusion between the SNIa and SNIbc types, in all confusion matrices, that may be related with intrinsic similarities of the mechanisms that cause the SN-peak: the diffusion of energy deposited by radioactive ^{56}Ni (Arnett, 2008). The TimeModAttn model decreased the confusion between the SNIa and SNIbc types w.r.t. the BRF baseline for both, the serial and parallel encoders. The TimeModAttn model achieved a maximum increment of the True Positive (TP) percentage for the SNIa type of $\Delta\text{TP}_{\text{SNIa}} = 9.2939^{***}$, for the SNIbc type of $\Delta\text{TP}_{\text{SNIbc}} = 8.8333^{***}$, for the SNII type of $\Delta\text{TP}_{\text{SNII}} = 4.6453^{***}$, and for the SLSN type of $\Delta\text{TP}_{\text{SLSN}} = 6.9444^{***}$[5].

Fig. 4.3 shows four light-curve examples that were correctly classified by the TimeModAttn model but incorrectly classified by the BRF baseline. Several misclassification errors arise among curves that, due to the irregular cadence, do not present observations in the SN-rise and/or SN-peak regions. This might cause instabilities in the SPM fitting, producing misleading features for the BRF baseline. For instance, features related with the rising time or the maximum brightness could be incorrectly estimated. Consequently, this may be especially detrimental for the discrimination between the SNIa and SNIbc types. In addition, multi-band light-curves having zero or few observations in one band tended to be misclassified by the BRF baseline too. As stated before, the scarcity of observations could lead to highly unstable SPM fittings and misleading features.

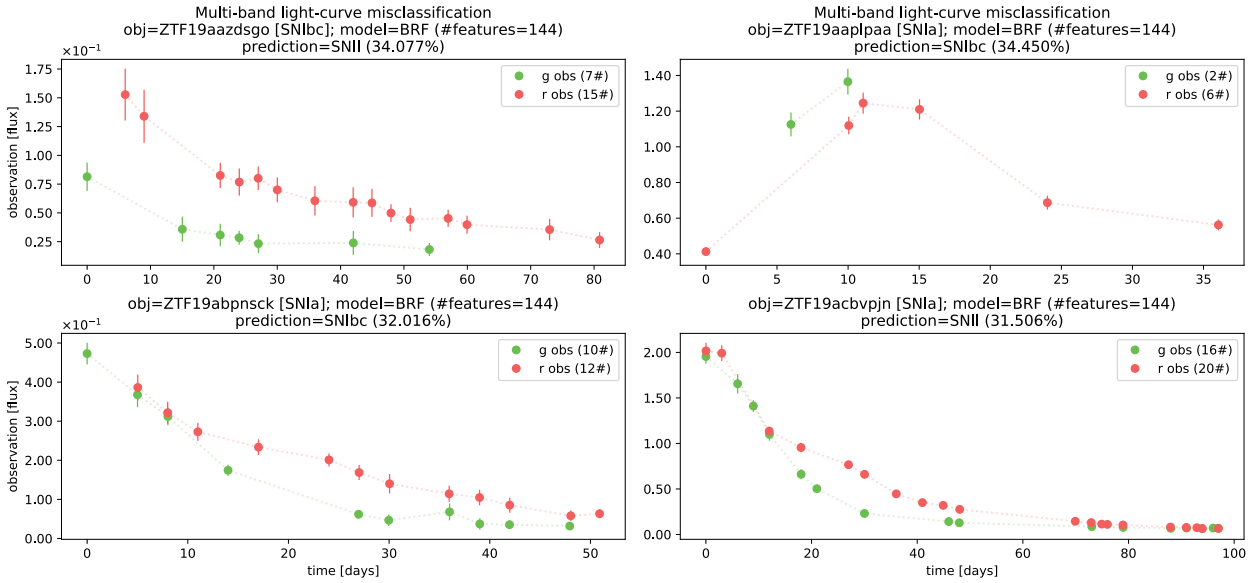

Figure 4.3: Four examples of misclassified SN multi-band light-curves by the BRF baseline that were correctly classified by the P-TimeModAttn model.

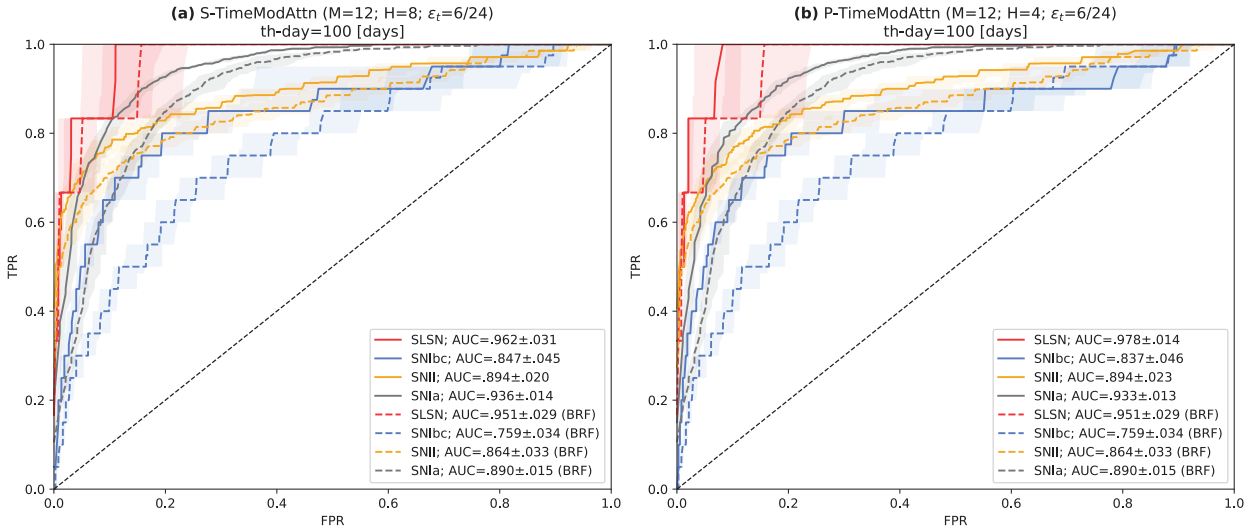

Figure 4.4: ROC curves for the SNe classification task using 100-day light-curves, where TPR and FPR stand for the True Positive Rate and False Positive Rate, respectively (50-percentile curve and 30-70-percentile region are shown from 5-fold cross-validation). In both plots, the BRF ROC curves are shown as dashed lines. (a) S-TimeModAttn. (b) P-TimeModAttn model.

Fig. 4.4 shows the ROC operational curves for each SN type. In general, it can be observed that, for all the SN types, the ROC curves reported from the TimeModAttn models are above those from the BRF baseline, leading to higher AUCROC scores per SN type for the TimeModAttn model. We highlight the ROC curves separation for the SNIa and SNIbc types, with maximum AUCROC differences of $\Delta\text{AUCROC}_{\text{SNIa}} = .0457^{***}$ and $\Delta\text{AUCROC}_{\text{SNIbc}} = .0877^{***}$[5], respectively. This fact correlates with the decrease of confusion errors found in the confusion matrices for these SN types.

Table 4.3: Late-classification performances for the BRF baseline and attention-based models (TimeModAttn) using 100-day multi-band light-curves and different target preserved single-bands. The value pb={g, r} indicates the target preserved single-band b^*. To avoid empty light-curve evaluations in the preserved single-band scenarios, light-curves with zero observations, in any of the B bands, were removed from the test-set $\mathcal{D}_{\text{test}}$. Both, the serial (S-{}) and parallel (P-{}) encoders are reported (mean±std from 5-fold cross-validation).

Model	b-Precision	b-Recall	b-F_1score	b-AUCROC	b-AUCPR
Multi-band models					
BRF (fmode=all; training-set=[r])	.530±.032	.692±.055	.530±.042	.867±.020	.605±.052
S-TimeModAttn (M=12; H=8; ε_t=6/24)	**.592±.022**	**.762±.039**	**.600±.033**	.911±.019	.675±.055
P-TimeModAttn (M=12; H=4; ε_t=6/24)	.586±.018	.757±.044	.599±.036	**.914±.017**	**.692±.047**
Single-band models					
P-TimeModAttn (M=12; H=4; ε_t=6/24; pb=g)	.518±.019	.661±.044	.508±.027	.848±.026	.584±.074
P-TimeModAttn (M=12; H=4; ε_t=6/24; pb=r)	.525±.019	.625±.057	.495±.036	.846±.024	.582±.031

4.1.4. Multi-band Effect in Classification

To study the effect of the multi-band information, we designed a setting to train and evaluate the TimeModAttn model using only a single-band b^*. With this aim, we use the parallel encoder where, given a target preserved single-band b^*, all the representation-vectors associated with the rest of the bands are replaced with zero vectors as $\boldsymbol{z}_i^{(b')} = \vec{0}, \forall b' \in \{1, \ldots, B\} - \{b^*\}$. Moreover, extra considerations were implemented to properly test this experiment, e.g., a zero reconstruction loss, in eq. (3.22), is used for all bands except for the target preserved single-band b^*; all information related with the observation-times is properly adjusted. Table 4.3 shows the reported performance for the TimeModAttn model with different target preserved single-bands.

The reported metrics show that the exclusive use of a single-band information (g or r) is significantly detrimental for the classification performance of the TimeModAttn model; therefore, using both bands is required to correctly characterize the SN transient events. Using all the available bands is especially beneficial when just a few observations are available in a particular band, where the model requires to support the classification task by using observations from the rest of the bands. Consequently, the use of serial or parallel encoders is recommended to properly capture all the information from a multi-band light-curve.

4.1.5. Interpretability Experiments

To explore, evaluate, and validate the automatic decisions of the TimeModAttn model, several experiments on interpretability are presented in this section. These experiments are based on the parallel encoder formulation, allowing us to explore the attention scores and the learned temporal modulation in each band.

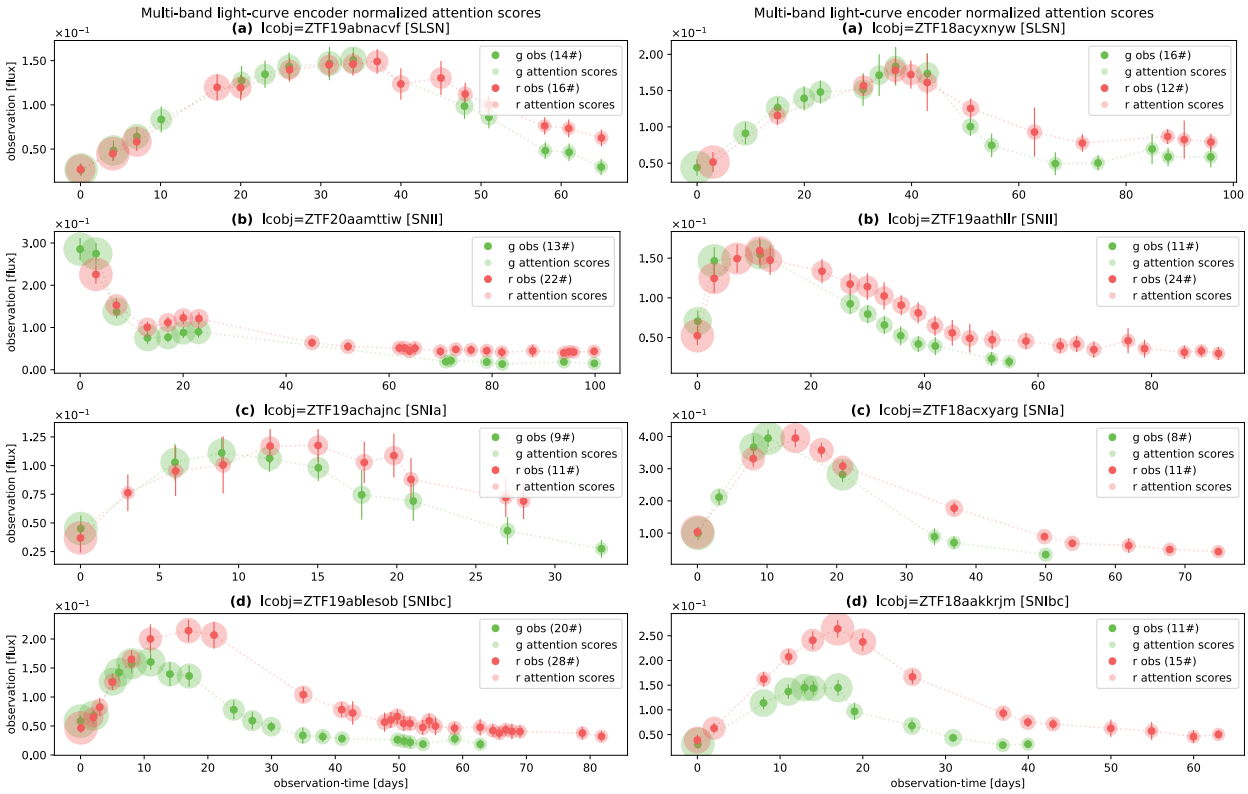

Figure 4.5: Examples of the normalized attention scores for the P-TimeModAttn model after pre-training. The bigger the shadow circle, the higher the attention score of an observation. (a) SLSN type. (b) SNII type. (c) SNIa type. (d) SNIbc type.

4.1.5.1. Attention Scores

Given a single-band light-curve $\Phi_i^{(b)}$, the attention scores $\left\{s_{i,j^{(b)}}\right\}_{j^{(b)}=1}^{L_i^{(b)}}$ are collected from the last MHSelfAttn's layer. In the multi-head attention scenario, the average score among the H heads is used: $s_{i,j^{(b)}} = \frac{1}{H}\sum_{h=1}^{H} s_{i,j^{(b)}}^{(h)}, \forall j^{(b)}$. Then, these attention scores are normalized as follows:

$$\bar{s}_{i,j^{(b)}} = \left(s_{i,j^{(b)}} - s_{i_{\min}}^{(b)}\right) / \left(s_{i_{\max}}^{(b)} - s_{i_{\min}}^{(b)}\right), \tag{4.1}$$

where $\bar{s}_{i,j^{(b)}} \in [0,1]$ is the normalized attention score given the original attention score $s_{i,j^{(b)}} \in \mathbb{R}^+$. The maximum and minimum attention scores, found in the band b, are denoted as $s_{i_{\max}}^{(b)} = \max\left\{s_{i,j^{(b)}}\right\}_{j^{(b)}=1}^{L_i^{(b)}}$ and $s_{i_{\min}}^{(b)} = \min\left\{s_{i,j^{(b)}}\right\}_{j^{(b)}=1}^{L_i^{(b)}}$, respectively. This procedure generates a maximum normalized attention score value of $\bar{s}_{i,j^{(b)}} = 1$ in the observation with the highest original attention score and a value of $\bar{s}_{i,j^{(b)}} = 0$ at the lowest.

Fig. 4.5 shows examples of the normalized attention scores for different SN multi-band light-curves. In general, the model tended to assign high normalized attention scores to (pay more attention to) early observations from the SN events, i.e., observations earlier than and close to the SN-peak.

We hypothesize that this attention behavior is produced because the first observations

seem to contain highly valuable information about the following evolution of the SN transient events. For example, the first observations can help the model to construct inner representations of the SN rising time (SN-rise region) and maximum brightness (SN-peak) that are relevant to discriminate among SNe, such as the SNIa and SNIbc types. In addition, the early SN observations (before the SN-peak) can be scarce due to the limited duration of the SN events and irregular cadence, which could also explain how the model handles the inner attention resources, prioritizing high attention scores on the early time region.

We highlight the high attention scores for the very first observations of a light-curve. For example, by using just the first observation, the model could construct an initial slope representation to distinguish if a SN light-curve started to be detected before the SN-peak (in the SN-rise region) or after the SN-peak (in the SN-fall region) due to the irregular cadence. The first observation could be used also as an observation-time offset, allowing the model to construct a representation of the elapsed time for each relevant and posterior observation. The first observation-time differences, among the bands, may offer relevant information of the multi-band behavior, which could be especially important when a specific band is started to be followed-up a long time after the rest of the bands.

4.1.5.2. Attention-Based Statistics

The main limitation of the attention score exploration presented above is that it heavily relies on a visual interpretation over a set of examples from a dataset $\mathcal{D}$. Based on a statistical approach, a new interpretability experiment is conducted to explore the attention behavior over a complete dataset $\mathcal{D}$. With this aim, we propose to use two simple and interpretable local features for a SN light-curve. Given a single-band light-curve $\Phi_i^{(b)}$, the following local SN-features are defined:

$$m^*_{i,j^{(b)}}, n^*_{i,j^{(b)}} = \underset{m_{i,j^{(b)}}, n_{i,j^{(b)}}}{\arg\min} \sum_{j^{(b)\prime} \in \Delta J_i^{(b)}} \left(\mu_{i,j^{(b)\prime}} - \hat{\mu}_{i,j^{(b)\prime}}\right)^2, \tag{4.2}$$

$$\Delta t^*_{i,j^{(b)}} = \frac{1}{L} \sum_{j^{(b)\prime} \in \Delta J_i^{(b)}} \left(t_{i,j^{(b)\prime}} - t_{i,j^{(b)}_{\max}}\right), \tag{4.3}$$

where the explanation is the following:

1. **SN-local-slope**: Given a linear function $\hat{\mu}_{i,j^{(b)\prime}} = m_{i,j^{(b)}} \cdot t_{i,j^{(b)\prime}} + n_{i,j^{(b)}}$, in eq. (4.2), a local slope value $m_{i,j^{(b)}}$ and an offset value $n_{i,j^{(b)}}$ are computed. The optimal values are estimated using a Mean Square Error (MSE) optimization, fitting the linear function over a group of empirical observation-fluxes defined by a window of local sequence steps $\Delta J_i^{(b)}$ centered in the sequence step $j^{(b)}$. A window size of $L = \#\left(\Delta J_i^{(b)}\right) = 3$ is used, i.e., the slope values are fitted using the observation-times $\left\{t_{i,j^{(b)}-1}, t_{i,j^{(b)}}, t_{i,j^{(b)}+1}\right\}$ and the observation-fluxes $\left\{\mu_{i,j^{(b)}-1}, \mu_{i,j^{(b)}}, \mu_{i,j^{(b)}+1}\right\}$. Positive values of $m^*_{i,j^{(b)}}$ are associated with observations with a local increase in brightness, while negative values are associated with observations with a local decrease in brightness.
2. **SN-peak-distance**: eq. (4.3) represents the average time difference (days) between the observation-times used to fit the SN-local-slope and the SN-peak time: the observation-

time associated with the empirical maximum brightness found along the light-curve. The maximum brightness observation-time is denoted as $t_{i,j^{(b)}_{\max}}$, where $j^{(b)}_{\max} = \arg\max_{j^{(b)}} \left\{\mu_{i,j^{(b)}}\right\}_{j^{(b)}=1}^{L_i^{(b)}}$ corresponds to the sequence step with the maximum observation-flux. Negative values of $\Delta t^*_{i,j^{(b)}}$ are associated with observations detected earlier than the SN-peak, while positive values are associated with observations detected after the SN-peak.

Given a dataset $\mathcal{D}$ with N light-curves Φ_i, we can gather a collection of local SN-features, for each observation, as $\left\{\left\{\left(m^*_{i,j^{(b)}}, \Delta t^*_{i,j^{(b)}}, b_{i,j^{(b)}}, \bar{s}_{i,j^{(b)}}\right)\right\}_{j^{(b)}=1}^{L_i^{(b)}}\right\}_{i=1}^{N}$, where $m^*_{i,j^{(b)}}$, $\Delta t^*_{i,j^{(b)}}$, $b_{i,j^{(b)}}$ and $\bar{s}_{i,j^{(b)}}$, are the SN-local-slope, the SN-peak-distance, the band, and the normalized attention score, respectively.

Using a probabilistic framework, let m^*, Δt^*, b, and $\bar{s}$ be discrete random variables. Fig. 4.6 shows the joint distribution $p(m^*, \Delta t^*, b) = \sum_{\bar{s}} p(m^*, \Delta t^*, b, \bar{s})$ (marginalizing over the normalized attention score $\bar{s}$), in plots (a.0) and (b.0), for the bands g and r, respectively. An expected SN behavior can be observed as the positive SN-local-slope values are distributed earlier than the SN-peak (SN-rise region), while negative SN-local-slope values are distributed after the SN-peak (SN-fall region). Small and zero values of the SN-local-slope are found in two scenarios: close to the SN-peak and in the SN-extinction region.

Fig. 4.6 shows the conditional joint distribution $p(m^*, \Delta t^*, b | \bar{s} \geq \bar{s}_{\text{th}})$, in plots (a.1) and (b.1), for the bands g and r, respectively. These distributions show the local SN-features that are related to high normalized attention scores using an attention threshold of $\bar{s}_{\text{th}} = .75$. When comparing the high distribution density w.r.t. the joint distribution, it can be observed that the high attention region is correlated with the attention score exploration shown in section §4.1.5.1, i.e., the model tended to pay more attention over observations earlier than and close to the SN-peak. The region of high attention was up to several days after the SN-peak, which might be an informative region to characterize the SN-plateau slope and duration[10].

Our findings suggest that early SN observations are the most important observations for the TimeModAttn model. Moreover, these observations can be directly captured by the attention-based models even in the case of long light-curves. We think this is important during training as early observations may be always available regarding the length of the light-curve, helping in the generalization of incomplete light-curves. In contrast, the RNN models may be biased to complete light-curves during training as their processing is forced to be sequential through all the sequence steps, which may hurt the generalization of incomplete light-curves. Moreover, RNN models may have difficulties when capturing information from early observations because the maximum path length could be large and composed of uninformative observations (SN-extinction region), especially in long-duration SN light-curves. This may explain why the TimeModAttn model achieved a general higher performance in early-classification than the RNN baselines.

[10]In general, this behavior is similar for both bands. Also, similar tendency was also observed when using more MHSelfAttn's layers ($N_L = 2$) or a different number of H attention heads.

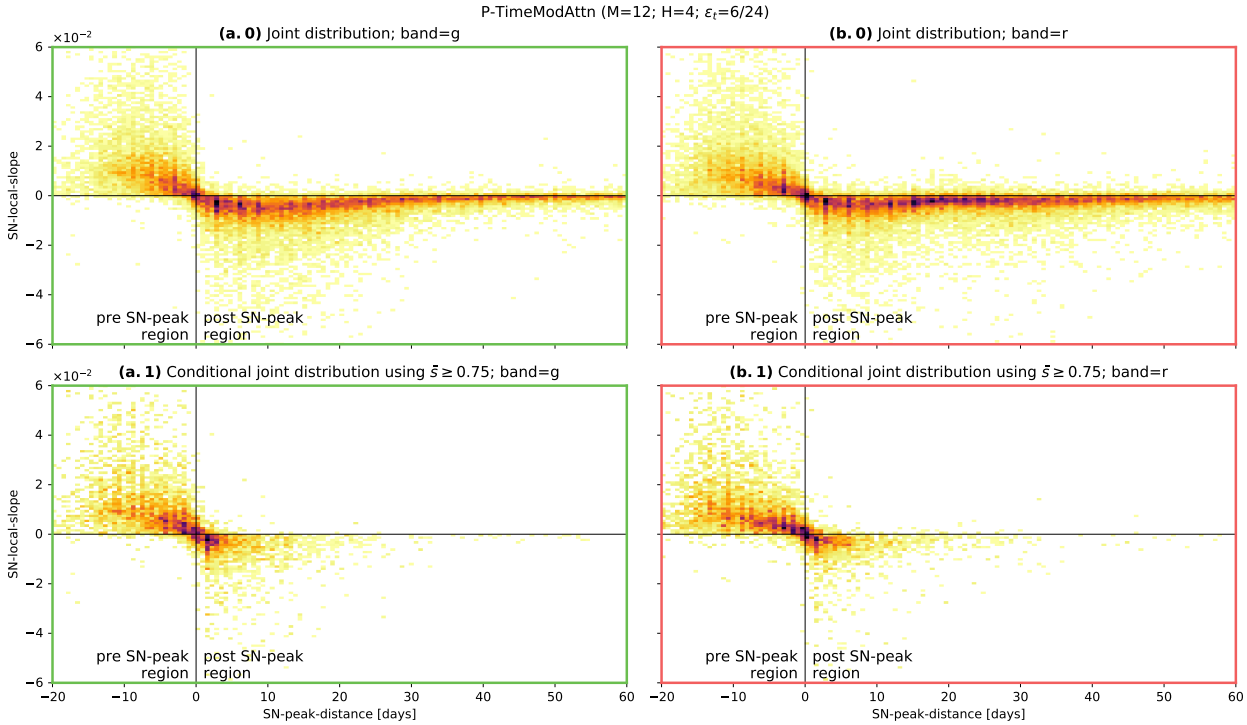

Figure 4.6: Attention-based statistics for the P-TimeModAttn model after pre-training. In the top row (plots (a.0) and (b.0) corresponding to the g and r band, respectively), the joint distribution is shown for both local SN-features, the SN-local-slope and the SN-peak-distance. In the bottom row (plots (a.1) and (b.1)), the conditional joint distribution is shown using a threshold for normalized attention scores of $\bar{s}_{\text{th}} = .75$. This allows highlighting the regions with higher attention scores. Visual guides, for zero SN-local-slope and zero SN-peak-distance, are shown as black lines. Green borders (plots (a.0) and (a.1)) correspond to the band g. Red borders (plots (b.0) and (b.1)) correspond to the band r.

4.1.5.3. Temporal Modulation Variability

The scale and bias variability time-functions of the proposed temporal modulation (TimeFiLM) are further analyzed. Given an arbitrary encoder associated with the band b, the temporal modulation variability time-functions are defined as follows:

$$\bar{\gamma}^{(b)}(t) = \frac{1}{K} \sum_{k=1}^{K} \left(\frac{\partial \gamma_k^{(b)}(t)}{\partial t} \right)^2, \tag{4.4}$$

$$\bar{\beta}^{(b)}(t) = \frac{1}{K} \sum_{k=1}^{K} \left(\frac{\partial \beta_k^{(b)}(t)}{\partial t} \right)^2, \tag{4.5}$$

where the functions $\bar{\gamma}^{(b)}(t)$ and $\bar{\beta}^{(b)}(t)$ are the variability time-functions for the scale and bias, respectively. These time-functions are defined as the average variability of the K modulation time-functions learned by the model. The variability is defined as the squared derivative of the modulation functions w.r.t. the time value t. Therefore, high values of the variability time-functions indicate a high average variability of the modulation over time. Fig. 4.7 shows the learned scale and bias variability time-functions for each run of the model.

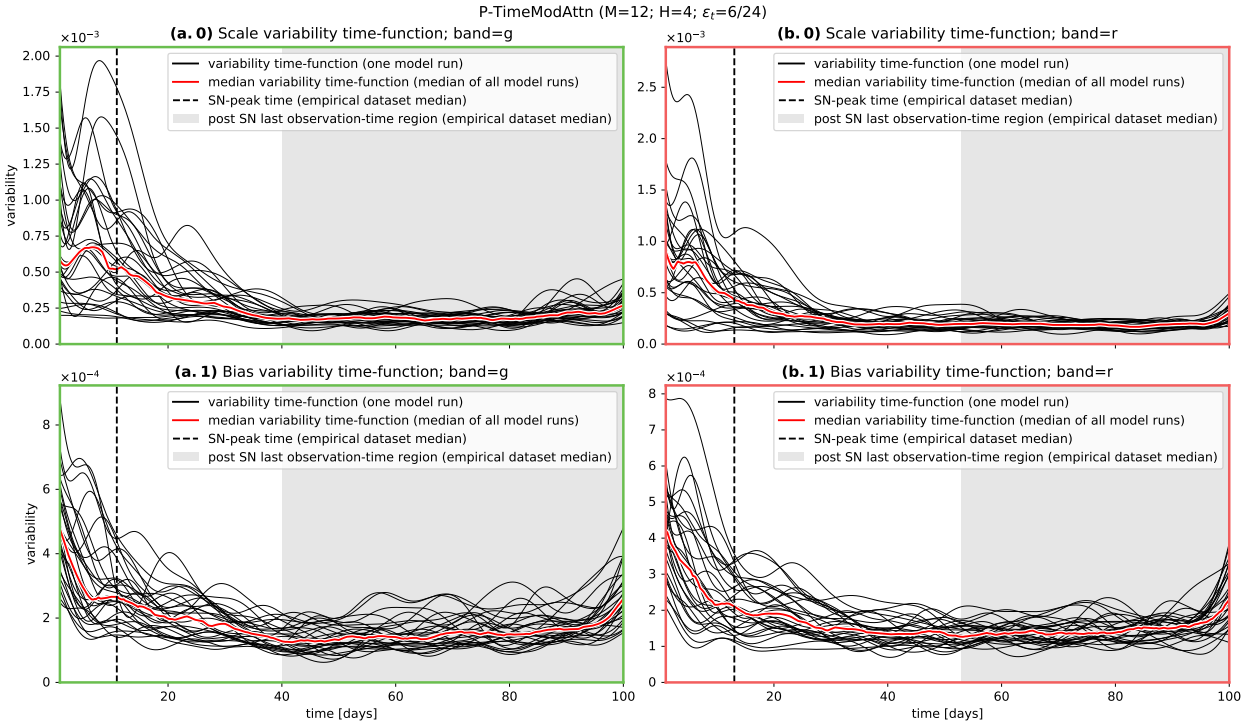

Figure 4.7: Variability functions for the scale time-function (plots (a.0) and (b.0)) and bias time-function (plots (a.1) and (b.1)) computed from the temporal modulations learned by the P-TimeModAttn model after the pre-training process for a time range $[0, 100]$ [days]. A high variability can be observed before the SN-peak time. Each model iteration (total of N_{runs}) is represented with a black curve. The red curves are computed as the median curve using all the N_{runs} model iterations. The SN-peak time is computed as the median empirical maximum brightness time from the original dataset $\mathcal{D}$. The gray region starts at the empirical median SN last observation-time computed from the dataset $\mathcal{D}$. Green borders (plots (a.0) and (a.1)) correspond to the band g. Red borders (plots (b.0) and (b.1)) correspond to the band r.

For both, the scale and bias variability time-functions, a general high variability over the early time range can be observed, i.e., earlier than the empirical median of the SN-peak time. This high variability could help the model to properly capture and differentiate small changes of the time values. This effect can be seen as a high temporal modulation resolution over the early time range[11]. This behavior is also correlated with the early high attention scores previously explored. We hypothesize that a high resolution is required to support the attention mechanisms, over the early time range, to correctly characterize the SN most important time regions. This high resolution can be beneficial when computing correlations between the time-modulated query and key vectors, as well as the final vector aggregation using the time-modulated value vectors.

The observed high variability, for both modulation functions, starts decreasing after passing the empirical median of the SN-peak time. A lower modulation resolution could be required in this time range, as the observation distribution becomes scarcer and sparser because the majority of light-curves have stopped being observed. Moreover, a large population

[11]Note that the scale function is a modulation operation that can offer a high degree of change over the modulator input vector, with direct attenuations and sign inversions.

Table 4.4: Total number of learnable parameters (#p) and empirical training times for RNN and attention-based models during the pre-training process (in GPU). We denote mbIT as the mini-batch Iteration Time measured for a complete forward and backward model operation.

Serial models	#p	mbIT [s]	mbIT/#p [µs]
S-RNN+Δt (cell=GRU)	267,270	.040±.007	.151±.026
S-RNN+Δt (cell=LSTM)	300,294	.041±.007	.135±.023
S-TimeModAttn (M=12; H=8; ε_t=6/24)	290,310	**.038±.006**	**.132±.020**
Parallel models			
P-RNN+Δt (cell=GRU)	234,246	.044±.008	.187±.033
P-RNN+Δt (cell=LSTM)	250,886	.045±.007	.179±.029
P-TimeModAttn (M=12; H=4; ε_t=6/24)	249,094	**.043±.007**	**.174±.027**

of low attention score observations can be found in this time range, suggesting the presence of irrelevant observations. A final high variability when reaching the time $t = 100$ [days] can be observed. This final variability rise might be influenced by long-duration light-curves (e.g., SNII, SLSN) or by modulation instabilities when reaching the time range $t > 100$ [days], where no empirical observations were found during training[12].

4.1.6. Empirical Computational Cost

Table 4.4 shows the training times for the RNN baselines and TimeModAttn models. In an effort to fairly compare different models capacities, we also include the metric $\text{IT}_{\text{mb}}/\text{P}$, which denotes the training time per total number of model learnable parameters. We do not include the BRF baseline because its optimization does not rely on GPU usage and it heavily depends on the CPU specifications and multi-threading strategies.

For a similar number of parameters, it can be observed that the TimeModAttn models achieved highly competitive empirical training times w.r.t. the RNN baselines. Note that the computational cost of the time modulation is also included in the reported training times. From our experiments, the use of the serial encoder leads to lower values of mbIT/#p for each of the tested Deep Learning models, suggesting that this encoder alternative is more efficient in terms of training time per total number of parameters. Even if the parallel encoder is parallel in our formulation, the optimization procedure was sequentially implemented in this work. This implementation handles the computation of all B parallel representation-vectors $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_B\}$ one after another, which may not be optimal in terms of GPU usage.

4.1.7. Additional Experiments

In order to further extend the results shown in this section, several additional experiments where conducted. In Appendix §C.3, we propose and implement additional methods for the

[12]In general, this variability behavior was similar for both bands. Also, similar tendency was also observed when using more MHSelfAttn's layers ($N_L = 2$) or a different number of M harmonic components for the temporal modulation.

generation of SN multi-band light-curves. For instance, we implement methods, based on a linear and a B-spline interpolation, as an alternative to the optimization of SPM parameters using MCMC. In Appendix §C.3.1, we also propose three different BRF optimization settings: only empirical data, only synthetic data, and empirical and synthetic data. Along with these configurations, we train the BRF model and study the performance obtained by the additional generation methods.

On the other hand, in Appendix §C.6, several ablation studies and additional models where implemented for the Deep Learning models. For example, in Appendix §C.6.1, we studied how the bypass of the temporal modulation or/and the attention mechanisms affects the overall performance of the TimeModAttn model. Also, in Appendix §C.6.2, we implement additional models: 1) TimeModRNN model: to study the effect of using the temporal modulation along with RNNs. 2) CatTimeAttn model: to study the effect of using a concatenated Temporal Encoding instead of the temporal modulation in the TimeModAttn model.

Chapter 5

Conclusions

5.1. Conclusions

In this work, a Deep Learning model (TimeModAttn), based on attention mechanisms (MHSelfAttn) with temporal modulation (TimeFiLM), was proposed to process and classify multi-band light-curves for different SN types. The proposed model avoids the requirement for hand-crafted feature computations, missing-values assumptions, and explicit light-curve imputation and interpolation methods. The training process was performed in two sequential steps. First, a pre-training process was performed, using synthetic SN multi-band light-curves, in a semi-supervised learning scheme with two simultaneous goals to solve: a multi-band light-curve reconstruction task and a SN type classification task. Second, a domain adaptation fine-tuning process was performed, using empirical multi-band light-curves, in a supervised learning scheme, to solve a classification task. Moreover, we proposed a method for the generation of synthetic SN multi-band light-curves, which is based on the SPM function. This helps to increase both, the number of samples and the diversity of the irregular cadence population.

By using SN multi-band light-curves from the ZTF survey, we first tested the proposed TimeModAttn model in the late-classification scenario using different performance metrics. From our experiments, we found that the TimeModAttn model outperformed the feature-based BRF baseline. By comparing the confusion matrices of the TimeModAttn model, w.r.t. the BRF baseline, it was observed that the TimeModAttn model obtained fewer confusions between the SN types, with maximum and significant increments of the True Positive (TP) percentages for the SNIa, SNIbc, SNII, and SLSN types. These results are correlated with the ROC curves separation between the TimeModAttn model and the BRF baseline. We highlight the confusion reduction obtained between the SNIa and SNIbc types, especially recalling the importance of the SNIa type for cosmology purposes.

In the early-classification scenario, we found that the TimeModAttn model achieved the maximum BRF's reported b-AUCROC performance several days earlier. This indicates that the TimeModAttn model can correctly discriminate between SN types using fewer observations, i.e., shorter SN multi-band light-curves. This early-classification capability of the

TimeModAttn model could be especially useful for future high-volume data surveys such as the LSST survey, where a fast and accurate classification of astronomical events must be performed with the fewest number of observations as possible. These findings show us that the TimeModAttn model can be effectively used to process and classify multi-band light-curves from different SN types, without relying on any costly hand-crafted feature computation.

In this work, the TimeModAttn model outperformed the tested RNN baselines (LSTM and GRU models). We obtained a higher, statistically significant, performance in both, the late-classification and early-classification scenarios for the TimeModAttn model w.r.t. the RNN baselines. For both types of Deep Learning models, we found that the generation of synthetic light-curves allowed to increase their general performance. Our conjecture is that attention-based models have the advantage of being able to access any observation from a light-curve regarding the total length or current sequence step. In contrast, in RNN models the processing is forced to be sequential through all the sequence steps (where many of them could be uninformative). Therefore, we think that the proposed model performs better as it can only focus and pay attention to observations that are relevant in the SN context: the early observations from the SN light-curves.

On the other hand, and by testing different levels of data-augmentation, we observed a high sensibility of the RNN baselines against the level of data-augmentation. Specifically, a strong data-augmentation was highly detrimental for the RNN baselines. This effect may be due to the existing gap of the time difference distributions between the training-set and test-set when using the proposed data-augmentation procedures. In contrast, we observed high robustness for the TimeModAttn model against the data-augmentation levels. This robustness can be helpful when a significant discrepancy in the irregular cadence, between the training-set and test-set, may be expected given the survey conditions. Additionally, we found that the TimeModAttn model was highly competitive w.r.t. the RNN baselines in terms of the reported empirical training cost.

From our experiments, we found that the use of all the available band information (multi-band light-curve) resulted in a significant benefit for the TimeModAttn model performance w.r.t. the use of a single-band information. Therefore, the serial or parallel encoder should be used in order to correctly solve the proposed SN classification task by capturing all the information from the multi-band light-curves. Additionally, no consistent or strong statistical evidence was found to conclude which encoder alternative is the best for the SN classification task. In this work, we used the parallel encoder to conduct interpretability experiments; however, by taking into account the reported empirical training cost of the serial encoder, this encoder could be more suitable when faster and efficient models (and no explicit multi-band interpretability) are required. The parallel encoder could be further explored in the near future when a higher number of bands will be available with surveys such as the future LSST survey, where six bands will be available (Ivezić et al., 2019).

Since the proposed model uses raw light-curves, it should not be a major problem to use it the context of ongoing and upcoming astronomical surveys based on light-curves. In fact, we expect that the use of this model may be well scalable in surveys where a large volume of data is expected per night (e.g., the current ZTF or the future LSST surveys), mainly because it is not necessary to perform a continuous and costly computation of features from light-curves. On the other hand, the proposed model can be easily parallelized given that its architecture

is based on the multi-head dot-attention mechanism. This can be a great advantage when optimizing pipelines for fast real-time inference of a large volume of data. Additionally, for class inference, the decoder can be discarded from the model to further reduce the inference time for real-time classification

We conducted several experiments on interpretability to explore the automatic decisions of the TimeModAttn model. We observed that the model tended to pay more attention to the first observations of the SN light-curves, i.e., the observations earlier than and close to the SN-peak. This behavior might be because the first observations offer highly valuable information about the evolution of SNe. This early attention behavior could allow the model to construct meaningfully inner representations to characterize a SN light-curve, e.g., the initial brightness slope; the SN brightness rising time, maximum peak, and early decay region; the elapsed time between the very first observation and the posterior observations. We found that the early high attention is correlated with a higher temporal modulation variability (or resolution) over the early time range. This increase in the variability could be beneficial to correctly induce the time information in the attention mechanisms operations: the computation of correlations between the time-modulated query and key vectors, as well as the final vector aggregation using the time-modulated value vectors.

In this thesis we also tested several alternative method for the generation of synthetic SN light-curves. In general, we cannot strongly assure that the proposed SPM-based methodology is superior to other simpler methods such as the linear interpolation. However, we noted that the selection of the generation method must be taken with care. For example, other methods, such as the B-spline interpolation, may not be completely appropriate due to artifacts presented in the resulting SN light-curves (e.g., an incorrect final brightness rising in the SN-extinction region)[1].

On the other hand, we compare the proposed model TimeModAttn w.r.t. all the training settings for the BRF baseline. In general, we highlight the fact that the use of synthetic data for training helps in the performance for both, the TimeModAttn model and the BRF baseline. In fact, although the TimeModAttn model shows a general better overall performance than all the tested BRF settings (early and late-classification), it is not possible to firmly conclude that this model is better for all the metrics used in this thesis (especially when using synthetic data for training). In any case, the use of the proposed model still has advantages over the use of feature-based models such as the BRF baseline. For example, we highlight: 1) Although the performance of both models improves when using synthetic data, this methodology is only scalable when using raw light-curves directly because the computation of features becomes extremely costly when including synthetic light-curves. 2) Possible useful data-augmentation techniques are only really scalable when using raw light-curves. 3) In contrast to the BRF baseline, the inference of the TimeModAttn model is straightforward for short light-curves with one o few observations. This effect can not be solved by using synthetic light-curves. 4) The use of features still heavily depends on expert knowledge which is a non-trivial and costly task. This task, moreover, should always be in constant revision, especially if new types of astronomical objects need to be studied[2].

[1]For extra details and discussion, see Appendix §C.3.
[2]For extra details and discussion, see Appendix §C.3.

As a final comment, the presented thesis generated the additional manuscript of a paper[3] for an astronomical journal (AJ). This manuscript was accepted, with minor revisions, by the referees and it is currently in the process of being corrected by the authors.

5.1.1. Future Work

No astrophysical external metadata (e.g., ALLWISE colors, galactic coordinates, SGS score, redshift) input capability was used in this work. To deal with optional and external metadata, we propose to extend our model to include metadata values using an extra modulation process. Given that metadata values could exhibit non-Gaussian distribution behaviors (e.g., multi-modal, clipped range, sparse distributions), we can directly use our proposed time modulation as a new «metadata modulation» over the corresponding non-redundant metadata-range associated with the handled metadata values. This modulation could be used, over the sequence input or directly over the encoder representation-vector, inducing the metadata information in the representation-vectors. This could be extended to multiple metadata values in a multi-layer metadata modulation architecture.

Given the space-state model formulation presented in this work (decoder), the forecasting of SN light-curves could be further explored as the space-state model can evolve arbitrarily over unobserved and future time values. Neural ODE decoders (Chen et al., 2018; Rubanova et al., 2019) could be also tested to perform continuous-time forecasting, avoiding the use of any explicit time differences information. Attention-based decoders could be also explored to implement an autoencoder model based solely on attention mechanisms. Moreover, a direct projection of the representation-vector (from the encoder) could be used as a decoding strategy, where the temporal modulation could be used to induce the information of the time values.

We believe that one of the major difficulties to face in a new survey may be the nature of its irregular cadence as well as the class imbalance. To further validate our proposed methodology, we propose to process data from other astronomical surveys, eventually testing it on the future LSST survey. Our hope is that our proposed methodology and the TimeModAttn model will be robust enough to achieve competitive results. Alternatively, we propose to classify other astronomical objects, such as stochastic events and periodic stars (using unfolded or folded light-curves). Given that periodic stars usually have a larger number of observations than SNe (therefore, longer light-curves), we expect that the use of the attention mechanisms could be beneficial given its natural long-term time dependencies learning capability. In addition, new interpretability experiments could be proposed for periodic stars, e.g., by exploring periodicities in the attention scores that could be better exposed by using folded light-curves.

Along with important projects such as the LSST survey, thousands of new observations will be detected every night in the future, which gives the scientific community the opportunity to discover unexpected and rare events. Therefore, it is critically important to design new algorithms that can process photometric information, allowing us the detection of anomalous light-curves. Several novel works have aimed to tackle this anomaly detection (Webb et al.,

[3] `https://arxiv.org/abs/2201.08482`.

2020; Villar et al., 2021; Malanchev et al., 2021; Muthukrishna et al., 2021). For example, Villar et al. (2021) proposes the use of a Variational AutoEncoder (VAE) in order to collapse the information from a light-curve into a latent space. Then, anomalous events can be detected using methods such as an Isolation Forest. The encoder proposed in this thesis could be used as an alternative when implementing an unsupervised VAE architecture, exploring anomalous light-curves along with the Isolation Forest. The exploration of the attention scores could help us to understand the local behaviors that cause a light-curve to be anomalous, i.e., ¿What are the properties of some observations that lead the model pay more attention to them?

In this thesis a simple expression for the reconstruction loss, based on the MSE loss, was used. As future work, we aim to experiment with reconstruction losses that introduce the observation-errors in their definition, i.e., the higher the observation-error, the lower the reconstruction penalty for that observation-flux estimation. An example of this is the Weighted MSE (WMSE) loss proposed by Naul et al. (2018). Additional changes for the reconstruction loss can be also further explored. In eq. (3.5), the empirical observation-flux $\mu_{i,j^{(b)}}$ can be refined as follows: $\mu_{i,j^{(b)}} \leftarrow \mu_{i,j^{(b)}} \sim \mathcal{N}\big(\mu_{i,j^{(b)}}, (k \cdot \sigma_{i,j^{(b)}})^2\big)$, which aims to capture the degree of uncertainty (by using the observation-error $\sigma_{i,j^{(b)}}$) directly in the loss function[4]. Note that this new observation-flux definition is stochastic and must be dynamically sampled during the training process. This stochasticity may allows us to induce an extra degree of regularization in the Deep Learning model optimization. This can also be tackled by using the reparametrization trick found in Variational Autoencoders (VAEs): $\mathcal{N}(\mu, \sigma^2) \approx \mu + \sigma \cdot \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, 1)$[5]. Note that the aforementioned approach can resemble a denoising autoencoder, a methodology that has been vastly and successfully used for image autoencoding.

The interpretability of the attention mechanisms can be extended. For example, although in this thesis we simplified the problem by taking the average attention score among all attention heads, we could also study the contribution of each attention head individually ¿Are there attention heads that specialize and focus on any particular behavior of the light-curve, e.g., early time regions, brightness peaks, light-curve outliers, low observation-error observations? ¿Are there attention heads that are redundant? ¿Can those attention heads be pruned without affecting the overall performance of the model? In order to carry out this study, inspiration could be taken from NLP-based works in the literature (Voita et al., 2019).

As described in the methodology section, the proposed model, thanks to the temporal modulation, is prepared to handle the temporal offsets that may exist in the bands of a multi-band light-curve. This information is accessible for both serial and parallel configurations; therefore, we expect that the model is able to capture the temporal offsets during optimization. For the RNN-based models, the time information is incorporated as temporal differences. It is worth noting that the characterization of the temporal offset between bands is neither straightforward nor trivial for the serial configuration of the RNN baselines: the model is expected to automatically be able to perform an accumulation of differences to obtain this information. How the offset affects the performance of the models, both the proposed and the RNN baselines, is outside the scope of this thesis. To study the impact,

[4]Extra considerations may be necessary if normalization methods are used for the observation-fluxes in the reconstruction process.

[5]Even if it is straightforward to implement, it is limited to the Gaussian distribution assumption.

we propose to use a data-augmentation method that only modifies or induce offsets between bands. With this, we expect to measure the impact of increasing the offset in the multi-band light-curves, studying which model characterize the temporal offsets the best.

Since real-time detection speed is critically important and a desired goal in the new paradigm of Big Data in astronomy, model optimization techniques can be explored to reduce the inference times of the proposed model. For example, we could further explore techniques such as model quantization. Model quantization aims to discretize the full precision of the model's tensors into low-bit tensors without much degradation of the model performance. The use of this technique has helped to to reduce the inference times in real-time text spotting and detection models (Liu et al., 2021).

5.1.2. Limitations

An hypothetical limitation of the proposed temporal modulation (TimeFiLM) is the selection of the non-redundant time modulation-range $[0, T_{\max}]$. In this work, the maximum period $T_{\max}$ was selected in order to be longer than a maximum fixed light-curve duration. Hypothetically and depending on the dataset properties (e.g., maximum time duration light-curves, sparsity of observation over time), the selection of $T_{\max}$ could be non-trivial and could even affect the performance. Note that this maximum non-redundant challenge is not presented in the RNN models as they use the time difference as the irregular cadence information; therefore, no maximum period must be assumed.

If a poor empirical performance is found due to a wrong selection of $T_{\max}$, a possible solution for the non-redundant time modulation problem is to perform a time sliding window methodology. Given a light-curve with an arbitrary time duration $T_{\max}$, a variable number of L' short time windows of duration $T_{\max}/L'$ can be used to split the original light-curve into L' new sub light-curves. Then, each sub light-curve[6] can be processed by the proposed TimeModAttn model to get a representation-vector. After this process, a variable number of L' representation-vectors will be obtained. Given that the sliding window methodology can be performed with regular and sequentially separated time intervals, the set of representation-vectors can be finally post-processed by using a simple variable-length average or even a memory-based model, such as the GRU or LSTM models.

Another conflicting scenario may occur in the case when regions of the modulation-range are not properly populated during the training process. Given this scenario, there may be problems when evaluating new light-curves that could have observations along time values never seen before during training (due to a poor model's generalization). A possible solution could be to pre-train the model using synthetic light-curves that are explicitly designed to cover the entire modulation-range. These light-curves can even have a random behavior since the main objective is to «familiarize» the model with different time values, which may be beneficial for the subsequent fine-tuning processes.

[6]Each sub light-curve still maintains properties such as the irregular cadence, variable-length, multi-band, etc.

Bibliography

Alcock, C., Allsman, R. A., Axelrod, T. S., et al. 1996, The Astrophysical Journal, 461, 84, doi: 10.1086/177039

Allam, T., & McEwen, J. D. 2021, arXiv e-prints, arXiv:2105.06178. https://arxiv.org/abs/2105.06178

Allevato, V., Paolillo, M., Papadakis, I., & Pinto, C. 2013, The Astrophysical Journal, 771, 9, doi: 10.1088/0004-637X/771/1/9

Alsabti, A. W., & Murdin, P., eds. 2017, Handbook of Supernovae (Cham: Springer International Publishing), doi: 10.1007/978-3-319-21846-5

Arévalo, P., Churazov, E., Zhuravleva, I., Hernández-Monteagudo, C., & Revnivtsev, M. 2012, Monthly Notices of the Royal Astronomical Society, 426, 1793, doi: 10.1111/j.1365-2966.2012.21789.x

Arnett, D. 2008, AIP Conference Proceedings, 1053, 237, doi: 10.1063/1.3009489

Ba, J. L., Kiros, J. R., & Hinton, G. E. 2016, arXiv e-prints, arXiv:1607.06450. https://arxiv.org/abs/1607.06450

Bahdanau, D., Cho, K., & Bengio, Y. 2014, International Conference on Learning Representations, ICLR 2015. https://arxiv.org/abs/1409.0473

Becker, I., Pichara, K., Catelan, M., et al. 2020, Monthly Notices of the Royal Astronomical Society, 493, 2981, doi: 10.1093/mnras/staa350

Bellm, E. C., Kulkarni, S. R., Graham, M. J., et al. 2019, Publications of the Astronomical Society of the Pacific, 131, doi: 10.1088/1538-3873/aaecbe

Boone, K. 2019, The Astronomical Journal, 158, 257, doi: 10.3847/1538-3881/ab5182

Breiman, L. 2001, Machine Learning, 45, doi: 10.1023/A:1010933404324

Brunel, A., Pasquet, J., Pasquet, J., et al. 2019, Electronic Imaging, 2019, 90, doi: 10.2352/ISSN.2470-1173.2019.14.COLOR-090

Carrasco-Davis, R., Cabrera-Vives, G., Förster, F., et al. 2019, Publications of the Astronomical Society of the Pacific, 131, doi: 10.1088/1538-3873/aaef12

Carrasco-Davis, R., Reyes, E., Valenzuela, C., et al. 2021, The Astronomical Journal, 162, 231, doi: 10.3847/1538-3881/ac0ef1

Chambers, K. C., Magnier, E. A., Metcalfe, N., et al. 2016, arXiv e-prints, arXiv:1612.05560. https://arxiv.org/abs/1612.05560

Charnock, T., & Moss, A. 2017, The Astrophysical Journal, 837, L28, doi: 10.3847/2041-8213/aa603d

Chen, C., Liaw, A., & Breiman, L. 2004, University of California, Berkeley

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. 2018, in Advances in Neural Information Processing Systems, ed. S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett, Vol. 31 (Curran Associates, Inc.). https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf

Chib, S., & Greenberg, E. 1995, The American Statistician, 49, 327, doi: 10.2307/2684568

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. 2014, NIPS 2014 Deep Learning and Representation Learning Workshop. http://arxiv.org/abs/1412.3555

Dark Energy Survey Collaboration, Abbott, T., Abdalla, F. B., et al. 2016, Monthly Notices of the Royal Astronomical Society, 460, 1270, doi: 10.1093/mnras/stw641

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. 2019, in Proceedings of the 2019 Conference of the North (Stroudsburg, PA, USA: Association for Computational Linguistics), 4171–4186, doi: 10.18653/v1/N19-1423

Donoso-Oliva, C., Cabrera-Vives, G., Protopapas, P., Carrasco-Davis, R., & Estevez, P. A. 2021, Monthly Notices of the Royal Astronomical Society, 505, 6069, doi: 10.1093/mnras/stab1598

Drake, A. J., Djorgovski, S. G., Mahabal, A., et al. 2009, The Astrophysical Journal, 696, 870, doi: 10.1088/0004-637X/696/1/870

Duchi, J., Hazan, E., & Singer, Y. 2011, Journal of Machine Learning Research, 12, 2121. http://jmlr.org/papers/v12/duchi11a.html

Eyheramendy, S., Elorrieta, F., & Palma, W. 2018, Monthly Notices of the Royal Astronomical Society, 481, 4311, doi: 10.1093/mnras/sty2487

Feigelson, E. D., Jogesh Babu, G., & Caceres, G. A. 2018, Frontiers in Physics, 6, doi: 10.3389/fphy.2018.00080

Foreman-Mackey, D., Hogg, D. W., Lang, D., & Goodman, J. 2013, Publications of the Astronomical Society of the Pacific, 125, 306, doi: 10.1086/670067

Förster, F., Maureira, J. C., Martín, J. S., et al. 2016, The Astrophysical Journal, 832, 155, doi: 10.3847/0004-637X/832/2/155

Förster, F., Cabrera-Vives, G., Castillo-Navarrete, E., et al. 2021, The Astronomical Journal, 161, 242, doi: 10.3847/1538-3881/abe9bc

Gaia Collaboration. 2016, arXiv e-prints, arXiv:1609.04153, doi: 10.1051/0004-6361/201629272

Gómez, C., Neira, M., Hernández Hoyos, M., Arbeláez, P., & Forero-Romero, J. E. 2020, Monthly Notices of the Royal Astronomical Society, 499, 3130, doi: 10.1093/mnras/staa2973

Goodman, J., & Weare, J. 2010, Communications in Applied Mathematics and Computational Science, 5, 65, doi: 10.2140/camcos.2010.5.65

Graham, M. J., Djorgovski, S. G., Drake, A. J., et al. 2017, Monthly Notices of the Royal Astronomical Society, 470, 4112, doi: 10.1093/mnras/stx1456

He, K., Zhang, X., Ren, S., & Sun, J. 2015, in 2015 IEEE International Conference on Computer Vision (ICCV) (IEEE), 1026–1034, doi: 10.1109/ICCV.2015.123

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. 2012. https://arxiv.org/abs/1207.0580

Hochreiter, S., & Schmidhuber, J. 1997, Neural Computation, 9, 1735, doi: 10.1162/neco.1997.9.8.1735

Hochreiter, S., Younger, A. S., & Conwell, P. R. 2001, in Artificial Neural Networks — ICANN 2001 (Springer Berlin Heidelberg), 87–94, doi: 10.1007/3-540-44668-0_13

Hockett, C. F., Shannon, C. L., & Weaver, W. 1953, Language, 29, 69, doi: 10.2307/410457

Horn, M., Moor, M., Bock, C., Rieck, B., & Borgwardt, K. 2019, International Conference on Machine Learning (ICML). https://arxiv.org/abs/1909.12064

Hosenie, Z., Lyon, R., Stappers, B., Mootoovaloo, A., & McBride, V. 2020, Monthly Notices of the Royal Astronomical Society, 493, 6050, doi: 10.1093/mnras/staa642

Huijse, P., Estévez, P. A., Förster, F., et al. 2018, The Astrophysical Journal Supplement Series, 236, 12, doi: 10.3847/1538-4365/aab77c

Hunter, J. D. 2007, Computing in Science & Engineering, 9, 90, doi: 10.1109/MCSE.2007.55

Ibsen, A., & Mann, B. 2020, in Astronomical Society of the Pacific Conference Series, Vol. 527, Astronomical Society of the Pacific Conference Series, ed. R. Pizzo, E. Deul, J. Mol, J. de Plaa, & H. Verkouter, 167

Ioffe, S., & Szegedy, C. 2015, in Proceedings of Machine Learning Research, Vol. 37, Proceedings of the 32nd International Conference on Machine Learning, ed. F. Bach & D. Blei (Lille, France: PMLR), 448–456. https://proceedings.mlr.press/v37/ioffe15.html

Ishida, E. E. O., & de Souza, R. S. 2013, Monthly Notices of the Royal Astronomical Society, 430, 509, doi: 10.1093/mnras/sts650

Ivezić, Ž., Kahn, S. M., Tyson, J. A., et al. 2019, The Astrophysical Journal, 873, doi: 10.3847/1538-4357/ab042c

Jamal, S., & Bloom, J. S. 2020, The Astrophysical Journal Supplement Series, 250, doi: 10.3847/1538-4365/aba8ff

Jayasinghe, T., Kochanek, C. S., Stanek, K. Z., et al. 2018, Monthly Notices of the Royal Astronomical Society, 477, 3145, doi: 10.1093/mnras/sty838

Karpenka, N. V., Feroz, F., & Hobson, M. P. 2012, Monthly Notices of the Royal Astronomical Society, 429, 1278, doi: 10.1093/mnras/sts412

Kazemi, S. M., Goel, R., Eghbali, S., et al. 2019, arXiv e-prints, arXiv:1907.05321. https://arxiv.org/abs/1907.05321

Kessler, R., Conley, A., Jha, S., & Kuhlmann, S. 2010, arXiv e-prints, arXiv:1001.5210. https://arxiv.org/abs/1001.5210

Kessler, R., Bernstein, J. P., Cinabro, D., et al. 2009, doi: 10.1086/605984

Kim, D.-W., Protopapas, P., Bailer-Jones, C. A. L., et al. 2014, Astronomy & Astrophysics, 566, A43, doi: 10.1051/0004-6361/201323252

Kim, D.-W., Protopapas, P., Byun, Y.-I., et al. 2011, The Astrophysical Journal, 735, 68, doi: 10.1088/0004-637X/735/2/68

Kim, S., & Kang, M. 2019, arXiv e-prints, arXiv:1902.10877. https://arxiv.org/abs/1902.10877

Kingma, D. P., & Ba, J. 2015, in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, ed. Y. Bengio & Y. LeCun. http://arxiv.org/abs/1412.6980

Lee, Y., Jun, E., & Suk, H.-I. 2021, arXiv e-prints, arXiv:2101.09986. https://arxiv.org/abs/2101.09986

Lin, H., Ye, Y., Leung, K.-C., & Zhang, B. 2020, in Genetic and Evolutionary Computing. ICGEC 2019. Advances in Intelligent Systems and Computing (Springer, Singapore.), 491–499, doi: 10.1007/978-981-15-3308-2_54

Liu, Y., Shen, C., Jin, L., et al. 2021, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1, doi: 10.1109/TPAMI.2021.3107437

Lochner, M., McEwen, J. D., Peiris, H. V., Lahav, O., & Winter, M. K. 2016, The Astrophysical Journal Supplement Series, 225, 31, doi: 10.3847/0067-0049/225/2/31

Loshchilov, I., & Hutter, F. 2017. https://arxiv.org/abs/1711.05101

Malanchev, K. L., Pruzhinskaya, M. V., Korolev, V. S., et al. 2021, Monthly Notices of the Royal Astronomical Society, 502, 5147, doi: 10.1093/mnras/stab316

Matheson, T., Stubens, C., Wolf, N., et al. 2021, The Astronomical Journal, 161, 107, doi: 10.3847/1538-3881/abd703

McInnes, L., Healy, J., & Melville, J. 2018. https://arxiv.org/abs/1802.03426

Mckinney, W. 2011, Python High Performance Science Computer

McLaughlin, M. A., Mattox, J. R., Cordes, J. M., & Thompson, D. J. 1996, The Astrophysical Journal, 473, 763, doi: 10.1086/178188

Möller, A., & de Boissière, T. 2020, Monthly Notices of the Royal Astronomical Society, 491, 4277, doi: 10.1093/mnras/stz3312

Mondrik, N., Long, J. P., & Marshall, J. L. 2015, The Astrophysical Journal, 811, L34, doi: 10.1088/2041-8205/811/2/L34

Moss, A. 2018, arXiv e-prints, arXiv:1810.06441. https://arxiv.org/abs/1810.06441

Müller, A. C., & Guido, S. 2017, Introduction to Machine Learning with Python: A Guide for Data Scientists

Muthukrishna, D., Mandel, K. S., Lochner, M., Webb, S., & Narayan, G. 2021, arXiv e-prints, arXiv:2111.00036. https://arxiv.org/abs/2111.00036

Muthukrishna, D., Narayan, G., Mandel, K. S., Biswas, R., & Hložek, R. 2019, Publications of the Astronomical Society of the Pacific, 131, 118002, doi: 10.1088/1538-3873/ab1609

Naul, B., Bloom, J. S., Pérez, F., & Van Der Walt, S. 2018, Nature Astronomy, 2, doi: 10.1038/s41550-017-0321-z

Naylor, T. 1998, Monthly Notices of the Royal Astronomical Society, 296, doi: 10.1046/j.1365-8711.1998.01314.x

Noebauer, U. M., Kromer, M., Taubenberger, S., et al. 2017, Monthly Notices of the Royal Astronomical Society, 472, 2787, doi: 10.1093/mnras/stx2093

Nun, I., Protopapas, P., Sim, B., et al. 2015, arXiv e-prints, arXiv:1506.00010. https://arxiv.org/abs/1506.00010

Pasquet, J., Pasquet, J., Chaumont, M., & Fouchez, D. 2019, Astronomy & Astrophysics, 627, A21, doi: 10.1051/0004-6361/201834473

Paszke, A., Gross, S., Massa, F., et al. 2019, in Advances in Neural Information Processing Systems 32, ed. H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, & R. Garnett (Curran Associates, Inc.), 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, Journal of Machine Learning Research, 12

Perez, E., Strub, F., de Vries, H., Dumoulin, V., & Courville, A. 2018, Proceedings of the AAAI Conference on Artificial Intelligence, 32. https://ojs.aaai.org/index.php/AAAI/article/view/11671

Petschek, A. G., ed. 1990, Supernovae, Astronomy and Astrophysics Library (New York, NY: Springer New York), doi: 10.1007/978-1-4612-3286-5

Pojmanski, G., Pilecki, B., Szczygiel, D. 2005, Acta Astronomica, 55, 97

Radford, A., Wu, J., Child, R., et al. 2019, OpenAI

Richards, J. W., Starr, D. L., Butler, N. R., et al. 2011, The Astrophysical Journal, 733, 10, doi: 10.1088/0004-637X/733/1/10

Riess, A. G., Filippenko, A. V., Challis, P., et al. 1998, The Astronomical Journal, 116, 1009, doi: 10.1086/300499

Rocklin, M. 2015, in Proceedings of the 14th Python in Science Conference, ed. K. Huff & J. Bergstra, 126–132, doi: 10.25080/Majora-7b98e3ed-013

Rosenblatt, F. 1958, Psychological Review, 65, 386, doi: 10.1037/h0042519

Rubanova, Y., Chen, R. T. Q., & Duvenaud, D. K. 2019, in Advances in Neural Information Processing Systems, ed. H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, & R. Garnett, Vol. 32 (Curran Associates, Inc.). https://proceedings.neurips.cc/paper/2019/file/42a6845a557bef704ad8ac9cb4461d43-Paper.pdf

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. 1986, Nature, 323, 533, doi: 10.1038/323533a0

Sainburg, T., McInnes, L., & Gentner, T. Q. 2020, arXiv e-prints, arXiv:2009.12981. https://arxiv.org/abs/2009.12981

Sánchez, E. 2006, AIP Conference Proceedings, 878, 213, doi: 10.1063/1.2409089

Sánchez-Sáez, P., Reyes, I., Valenzuela, C., et al. 2021, The Astronomical Journal, 161, 141, doi: 10.3847/1538-3881/abd5c1

Scargle, J. D. 1982, The Astrophysical Journal, 263, 835, doi: 10.1086/160554

Schmidt, B. P., Suntzeff, N. B., Phillips, M. M., et al. 1998, The Astrophysical Journal, 507, 46, doi: 10.1086/306308

Schmidt, K. B., Marshall, P. J., Rix, H.-W., et al. 2010, The Astrophysical Journal, 714, 1194, doi: 10.1088/0004-637X/714/2/1194

Shukla, S. N., & Marlin, B. M. 2021, International Conference on Learning Representations (ICLR). http://arxiv.org/abs/2101.10318

Smith, K. W., Williams, R. D., Young, D. R., et al. 2019, Research Notes of the AAS, 3, 26, doi: 10.3847/2515-5172/ab020f

Sousa, R. T., Pereira, L. A., & Soares, A. S. 2020, arXiv e-prints, arXiv:2003.09291. https://arxiv.org/abs/2003.09291

Spencer, B., & Reese, C. S. 2013, Utah Space Grant Consortium

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. 2014, Journal of Machine Learning Research, 15, 1929

Stellingwerf, R. F., & Donohoe, M. 1986, The Astrophysical Journal, 306, 183, doi: 10.1086/164331

Stokes, G. H., Evans, J. B., Viggh, H. E., Shelly, F. C., & Pearce, E. C. 2000, Icarus, 148, 21, doi: 10.1006/icar.2000.6493

Strobl, C., Boulesteix, A.-L., Zeileis, A., & Hothorn, T. 2007, BMC Bioinformatics, 8, 25, doi: 10.1186/1471-2105-8-25

Sukhbold, T., & Adams, S. 2020, Monthly Notices of the Royal Astronomical Society, 492, 2578, doi: 10.1093/mnras/staa059

Tachibana, Y., Graham, M. J., Kawai, N., et al. 2020, The Astrophysical Journal, 903, 54, doi: 10.3847/1538-4357/abb9a9

Tachibana, Y., & Miller, A. A. 2018, Publications of the Astronomical Society of the Pacific, 130, 128001, doi: 10.1088/1538-3873/aae3d9

Takase, S., & Okazaki, N. 2019. https://arxiv.org/abs/1904.07418

The PLAsTiCC team, Allam Jr., T., Bahmanyar, A., et al. 2018, arXiv e-prints, arXiv:1810.00001. https://arxiv.org/abs/1810.00001

Tsang, B. T.-H., & Schultz, W. C. 2019, The Astrophysical Journal, 877, L14, doi: 10.3847/2041-8213/ab212c

Udalski, A. 2003, Acta Astronomica, 53, 291

Varughese, M., Sachs, R., Stephanou, M., & Bassett, B. 2015, Monthly Notices of the Royal Astronomical Society, 453, doi: 10.1093/mnras/stv1816

Vaswani, A., Shazeer, N., Parmar, N., et al. 2017, in Advances in Neural Information Processing Systems, ed. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett, Vol. 30 (Curran Associates, Inc.). https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Villar, V. A., Cranmer, M., Berger, E., et al. 2021, The Astrophysical Journal Supplement Series, 255, 24, doi: 10.3847/1538-4365/ac0893

Villar, V. A., Berger, E., Miller, G., et al. 2019, The Astrophysical Journal, 884, 83, doi: 10.3847/1538-4357/ab418c

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., & Titov, I. 2019. https://arxiv.org/abs/1905.09418

Webb, S., Lochner, M., Muthukrishna, D., et al. 2020, Monthly Notices of the Royal Astronomical Society, 498, 3077, doi: 10.1093/mnras/staa2395

Wright, B. S., & Li, B. 2018, Physical Review D, 97, 083505, doi: 10.1103/PhysRevD.97.083505

Wright, E. L., Eisenhardt, P. R. M., Mainzer, A. K., et al. 2010, The Astronomical Journal, 140, 1868, doi: 10.1088/0004-6256/140/6/1868

York, D. G., Adelman, J., Anderson, Jr., J. E., et al. 2000, The Astronomical Journal, 120, 1579, doi: 10.1086/301513

Zeiler, M. D. 2012. https://arxiv.org/abs/1212.5701

Zhang, K., & Bloom, J. S. 2021, Monthly Notices of the Royal Astronomical Society, 505, 515, doi: 10.1093/mnras/stab1248

Zimmermann, H.-G., Tietz, C., & Grothmann, R. 2012, in Neural Networks: Tricks of the Trade: Second Edition (Springer Berlin Heidelberg), 687–707, doi: 10.1007/978-3-642-35289-8_37

Annexes

Annexed A

Theoretical Background Concepts

A.1. Additional Literature Summary Table

Table A.1 presents a brief summary of some of the literature works reviewed for the writing of this thesis. All these works consider the processing of light-curves or stamp-image sequences of different types of astronomical objects and events.

A.2. Balanced Multi-Class Performance Metrics

Given a multi-class dataset $\mathcal{D}$ and an arbitrary target class $c \in \{1, \ldots, C\}$, where C is the total number of classes in $\mathcal{D}$, a new binary class dataset $\mathcal{D}_{\text{c}}$ is constructed, where $\mathcal{D}_{\text{c}}$ has $C = 2$ classes: the positive class «c» and the negative class «$\bar{c}$». The new binary classes are assigned according to the original true class label c, and assigning the auxiliary negative class $\bar{c}$ to every sample from any other class different than the positive class c. Similarly, the new model binary class predictions are assigned according to the original model class prediction $c_i = \arg\max_c [\hat{y}_{i,1}, \ldots, \hat{y}_{i,C}]^T$, where c_i is the class associated with the highest predicted probability [1].

Given a binary class dataset $\mathcal{D}_{\text{c}}$, the Precision, Recall, and F_1score metrics are defined as follows:

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c}, \tag{A.1}$$

$$\text{Recall}_c = \frac{TP_c}{TP_c + FN_c}, \tag{A.2}$$

$$F_1\text{score}_c = 2 \cdot \frac{\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}, \tag{A.3}$$

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html`.

where TP_c, FP_c, and FN_c stand for the True Positive, False Positive, and False Negative binary class prediction scenarios given the dataset $\mathcal{D}_c$, respectively.

The Receiver Operating Characteristic (ROC)[2] and the Precision-Recall (PR)[3] curves are constructed by using the predicted probability $\hat{y}_{i,c}$ for the positive class «c» and the probability $1-\hat{y}_{i,c}$ for the negative class «$\bar{c}$». For the experiments, the Area Under the Curve (AUC) is reported for both, the ROC curve (AUCROC) and the PR curve (AUCPR).

All metrics computed along with binary datasets $\mathcal{D}_c$ can be aggregated into a new balanced metric. For example, the balanced F_1score is computed as follows:

$$\text{b-}F_1\text{score} = \frac{1}{C}\sum_{c=1}^{C} F_1\text{score}_c, \tag{A.4}$$

where each class performance is equally important for the final balanced metric result. The same process can be applied for each of the metrics used in this work, obtaining the following balanced metrics: b-Precision, b-Recall, b-F_1score, b-AUCROC, and b-AUCPR.

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html`.

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_curve.html`.

Table A.1: Additional summary for some of the main works that process photometric light-curves from different astronomical events. For each work, the processed SN types are shown. Several works can also/only process light-curves from different variable stars, stochastic events, and/or other general transient events (non-SN), where the details of these type of objects are not described for simplicity (∼). Information about the test model evaluation is also shown: evaluation over simulated data and evaluation over empirical (real) data. If the main contribution of the work is a light-curve feature extraction method such as SALT2, Gaussian Processes (GPs), Wavelet, SPM-based features, Auto-Regressive models (AR), hand-selected features, etc, then these are denoted as Conventional Feature Extraction (CFE). Deep Learning methods such as Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), Temporal CNN (TCNN), dilated TCNN (dTCNN), self-attention mechanisms, etc, are denoted as Deep Learning Feature Extraction (DLFE). †: the method presented can also process stamp-image sequences. ‡ : a function is used to model only the region previous to the SN-peak (SN-rise).

Reference	Dataset	Deals with multi-band	Eval over sim data	Eval over real data	SN types	CFE	DLFE	Uses SNe analytical model
Spencer & Reese (2013)	SNANA		✓		{Ia, non-Ia}	✓		
Karpenka et al. (2012)	SPCC	✓	✓		{Ia, non-Ia}	✓		✓
Ishida & de Souza (2013)	SPCC	✓	✓		{Ia, non-Ia}	✓		
Varughese et al. (2015)	SPCC	✓	✓		{Ia, non-Ia}	✓		
Lochner et al. (2016)	SPCC		✓		{Ia, II, Ibc}	✓		✓
Charnock & Moss (2017)	SPCC	✓	✓		{Ia, non-Ia}, {I, II, III}		✓	
Moss (2018)	SPCC	✓	✓		{Ia, non-Ia}, {Ia, II(IIn, IIP, IIL), Ibc}, {Ia, II, IIn, IIP, IIL, Ibc, Ib, Ic}		✓	
Naul et al. (2018)	ASAS, LINEAR, MACHO			✓	{∼}		✓	
Feigelson et al. (2018)	Kepler, HATS, ASAS			✓	{∼}	✓		
Villar et al. (2019)	Pan-STARRS	✓		✓	{SL, II, IIn, Ia, Ibc}	✓		✓
Pasquet et al. (2019)	SPCC, SNANA, SDSS	✓	✓	✓	{Ia, non-Ia}		✓	
Boone (2019)	PLAsTiCC	✓	✓		{Ia, Ibc, II, Ia-91bg, Ia-x, point-Ia, KN, SL, PI, ∼}	✓		
Carrasco-Davis et al. (2019)†	HiTS			✓	{SN(Ia, II), ∼}		✓	
Tsang & Schultz (2019)	ASAS-SN			✓	{∼}		✓	
Muthukrishna et al. (2019)	SNANA (ZTF), ZTF	✓	✓	✓	{Ia, Ibc, II, Ia-91bg, Ia-x, point-Ia, KN, SL, PI, ∼}		✓	✓‡
Brunel et al. (2019)	SNANA, SPCC	✓	✓		{Ia, non-Ia}		✓	
Möller & de Boissière (2020)	SNANA	✓	✓		{Ia, non-Ia}, {Ia, Ibc(Ib, Ic), II(IIn, IIL2, IIL1, IIP)}, {IIn, IIL2, IIL1, Ib, Ic, IIP, Ia}		✓	
Hosenie et al. (2020)	CRTS			✓	{∼}	✓		
Becker et al. (2020)	OGLE-III, Gaia, WISE			✓	{∼}		✓	
Jamal & Bloom (2020)	MACHO	✓		✓	{∼}		✓	
Gómez et al. (2020)†	CRTS(TAO)			✓	{SN, ∼}		✓	
Tachibana et al. (2020)	CRTS			✓	{∼}		✓	
Ibsen & Mann (2020)	PLAsTiCC	✓	✓		{Ia, Ibc, II, Ia-91bg, Ia-x, point-Ia, KN, SL, PI, ∼}		✓	
Sánchez-Sáez et al. (2021)	ZTF	✓		✓	{Ia, Ibc, II, SL}, {∼}	✓		✓
Zhang & Bloom (2021)	ASAS-SN, MACHO, OGLE-III			✓	{∼}		✓	
Donoso-Oliva et al. (2021)	ASAS, OGLE-III, MACHO, LINEAR, CRTS, Gaia, WISE			✓	{∼}		✓	
Allam & McEwen (2021)	PLAsTiCC	✓	✓		{Ia, Ibc, II, Ia-91bg, Ia-x, point-Ia, KN, SL, PI, ∼}		✓	

Annexed B

Methodology Details

B.1. Feature Extraction Implementation Details

In this work, the implementation for the feature extraction, created by the ALeRCE broker (Sánchez-Sáez et al., 2021), were used[1]. In order to properly use the ALeRCE's feature extractor, and for the sake of a fair comparison between the BRF baseline and the different Deep Learning models, some extra details of our implementation[2] are described below. In particular, we refer to the construction of the light-curve's DataFrame object that is handled and processed by the feature extractor object FeatureExtractorComposer[3] from the ALeRCE implementation. Therefore, the columns of the DataFrame used by the FeatureExtractorComposer are defined as follows:

1. `time`: the observation-time is used.
2. `magnitude`: the magnitude, transformed from the observation-flux, is used (see section §2.3.2.3).
3. `error`: the magnitude error, transformed from the observation-error, is used (see section §2.3.2.3).
4. `magpsf`: the magnitude, transformed from the observation-flux, is used (same as `magnitude`). It is required for the SNParametricModelExtractor object[4]
5. `sigmapsf`: the magnitude error, transformed from the observation-error, is used (same as `error`). It is required for the SNParametricModelExtractor object[4].

[1] `https://github.com/alercebroker/lc_classifier`.

[2] `https://github.com/oscarpimentel/astro-lightcurves-features/blob/master/lcfeatures/extractors.py`.

[3] `https://github.com/alercebroker/lc_classifier/blob/702451154c4482750f5395652117b296f392951c/lc_classifier/features/core/base.py`.

[4] `https://github.com/alercebroker/lc_classifier/blob/af94c49fac46b91ac2fd5eadc64bc1faee1b621f/lc_classifier/features/extractors/sn_parametric_model_computer.py`. Internally, the SNParametricModelExtractor object performs a flux-to-magnitude transformation in order to correctly use the SPM function.

6. `band`: the band index, from the photometric band, is used.

7. `isdiffpos`: it is set to 1[5]. Few and exceptional SN events, where values of -1 is presented in the light-curve's DataFrame column `isdiffpos` (wrong photometry process), are removed from the scope of this work (and from the dataset) as one of the very first steps of our pipeline.

We highlight that all the implemented Deep Learning models process the raw SN multi-band Light-curves, without any extra information such as metadata (e.g., `isdiffpos`). Therefore, for the sake of a fair comparison, the process of feature extraction is limited to the same information handled by the Deep Learning models, limiting the use of metadata and/or extra types of magnitude information (`magnitude`, `magpsf`). As future work, the Deep Learning models could be adapted in order to use metadata, as well as extra Light-curves constructed from additional flux information.

B.2. SPM Bounds and MCMC Prior Distribution

The Maximum Likelihood Estimation (MLE) optimization is performed using the curve-fit algorithm[6]. To ensure positive flux values and a general MLE fit stabilization, we impose valid bounds over the different SPM parameter values, which are shown in Table B.1. The initial MLE parameter guesses p_0 are also shown.

As mentioned in section §3.2.1, an isotropic multivariate Gaussian distribution is used for the MCMC prior distribution with a diagonal standard deviation matrix $\mathbf{\Sigma} \in \mathbb{R}^{6\times6}$. Each diagonal entry is proportional to the associated SPM bound range. For example, for the SPM parameter A, we define the standard deviation entry $\Sigma_{1,1}$ as $\sigma_{A_i^{(b)}} = k\Big(\sup\Big(A_i^{(b)}\Big) - \inf\Big(A_i^{(b)}\Big)\Big)$, where $k = .1$ is a scaling factor. Additionally, all the Gaussian distributions used in this work are truncated by the SPM bounds to avoid sampling any invalid SPM parameter values during the MCMC optimization.

B.3. Conditional Observation-Error Distribution Estimation

To estimate the conditional observation-error versus observation-flux distribution, we use the empirical samples from the training-set $\mathcal{D}_{\text{training}}$. We compute the maximum dispersion axis over the joint distribution $p(\mu, \sigma, b)$ using the principal component from a Principal Component Analysis (PCA) reduction[7]. Then, a rotation operation of the original space $p(\mu, \sigma, b)$ is performed by using the maximum dispersion axis slope for the construction of a rotation linear projection $\boldsymbol{W} \in \mathbb{R}^{2\times2}$ (see Fig. B.1 for an example of the rotated space

[5] `https://zwickytransientfacility.github.io/ztf-avro-alert/schema.html`.

[6] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html`.

[7] `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`.

Table B.1: SPM bounds and MLE initial guesses p_0, given an arbitrary single-band Light-curve's $\Phi_i^{(b)}$, for different SPM parameters (P). The sequence step $j_{\max}^{(b)} = \arg\max_{j^{(b)}} \left\{\mu_{i,j^{(b)}}\right\}_{j^{(b)}=1}^{L_i^{(b)}}$ corresponds to the sequence step with the maximum observation-flux (maximum brightness). The observation-times $\mathcal{T}_i^{(b)} = \left\{t_{i,j^{(b)}}\right\}_{\forall j^{(b)} | \mu_{i,j^{(b)}} \geq \frac{1}{3}\mu_{i,j_{\max}^{(b)}}}$ are the observation-times where the observation-fluxes are above a brightness threshold of $\frac{1}{3}\mu_{i,j_{\max}^{(b)}}$, where $\mu_{i,j_{\max}^{(b)}}$ is the maximum observation-flux. The observation-time $t_{i,1^{(b)}}$ is the first observation-time from the single-band Light-curve's $\Phi_i^{(b)}$.

P	Lower & upper SPM bounds	p_0
$A_i^{(b)}$	$\left(\frac{1}{5}\mu_{i,j_{\max}^{(b)}}, 5\mu_{i,j_{\max}^{(b)}}\right)$	$1.2\mu_{i,j_{\max}^{(b)}}$
$t_{\text{zero}i}^{(b)}$	$\left(t_{i,1^{(b)}} - 10, t_{i,j_{\max}^{(b)}} + 50\right)$	$t_{i,j_{\max}^{(b)}}$
$\gamma_i^{(b)}$	$(1, 120)$	$\max \mathcal{T}_i^{(b)} - \min \mathcal{T}_i^{(b)}$
$\beta_i'^{(b)}$	$(0, 1)$	$.5$
$t_{\text{rise}i}^{(b)}$	$(1, 50)$	$\frac{1}{2}\left(t_{i,j_{\max}^{(b)}} - t_{i,1^{(b)}}\right)$
$t_{\text{fall}i}^{(b)}$	$(1, 130)$	40

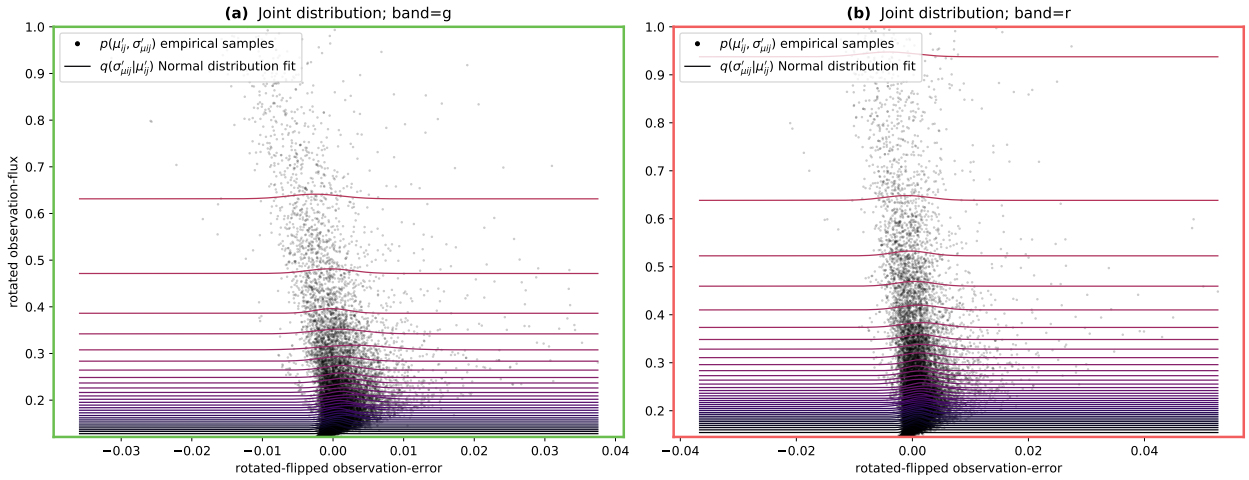

Figure B.1: Gaussian distribution fits for the conditional observation-error distribution (from the training-set $\mathcal{D}_{\text{training}}$ of an arbitrary fold split). Green border (a) corresponds to the band g. Red border (b) corresponds to the band r.

samples). Next, a collection of Gaussian distributions are fitted by MLE over the rotated space $p(\sigma'|\mu', b)$ by using several binned regions, where at least 50 empirical samples are found in each bin (see Fig. B.1 for examples of the Gaussian distribution fits).

In order to generate a new observation-error sample $\hat{\sigma}$, given the observation-flux μ and the band b, we first find the corresponding bin (target bin) associated with the observation-flux μ in the rotated space. Then, we sample a rotated observation-error $\hat{\sigma}'$ from the fitted Gaussian distribution associated with the target bin. Finally, the rotated observation-error

$\hat{\sigma}'$ is rotated back using the inverse rotation linear projection $\boldsymbol{W}^{-1}$ to obtain the required observation-error sample $\hat{\sigma}$.

B.4. Reconstruccion Loss Issue

During the preparation of this thesis, we thought that we were using another definition for the reconstruction w.r.t. the loss that was actually used. Based on the Weighted MSE (WMSE; Naul et al., 2018), our original loss definition is further described. Given a multi-band light-curve Φ_i, a reconstruction loss function is defined as follows:

$$\mathcal{L}_{\text{rec}_i} = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{L_i^{(b)}}\sum_{j^{(b)}=1}^{L_i^{(b)}}\frac{\left(\mu_{i,j^{(b)}} - \hat{\mu}_{i,j^{(b)}}\right)^2}{\gamma\cdot\sigma^2_{i,j^{(b)}} + \beta}, \tag{B.1}$$

where $\mu_{i,j^{(b)}}$ and $\sigma_{i,j^{(b)}}$ are the observation-flux and observation-error, from the single-band light-curve $\Phi_i^{(b)}$, respectively. The observation-flux predicted by the decoder, for the single-band light-curve $\Phi_i^{(b)}$, is denoted as $\hat{\mu}_{i,j^{(b)}}$. The values $\gamma = 1\text{e}{-3}$ and $\beta = 1$ are used to adjust the penalty induced by the observation-errors. Similar to Naul et al. (2018), a Weighted Mean Squared Error (WMSE) like-function is used for the reconstruction loss by also introducing the observation-errors $\sigma_{i,j^{(b)}}$, i.e., the higher the observation-error, the lower the reconstruction penalty for that observation-flux estimation.

In general, the motivation for the above definition is the following: the values γ and β are used to adjust the penalty induced by the observation-errors. Our conjecture is that, without this adjustment, the lowest observation-errors (presented in the training-set) can drag the optimization to sub-optimal local minimum regions over the loss surface, leading to a convergence biased towards these observations.

After reviewing our implementation, we observed that the value chosen for γ was too low considering the empirical values achieved by the observation-errors. This finally produces that the values for $\gamma\cdot\sigma_{i,j^{(b)}}$ tend to 0 and, therefore, the expression $\gamma\cdot\sigma^2_{i,j^{(b)}} + \beta$ tends to 1, yielding to a practical use of the MSE instead of the WMSE-like loss. Note that this behaviour heavily depends on the dataset used and the distribution of the observation-errors. For this reason, we finally decided that the MSE loss was the most theoretically correct way to define the used reconstruction loss in this thesis.

Note that nothing stops us from experimenting with more and well-defined values for γ, but this is beyond the scope of this thesis. Therefore, it remains as a future task the study of the impact of using this type of WMSE loss, as well as the selection of the values for γ. Our final conjecture is that the effective use of the WMSE loss does not necessarily imply an improvement in the results presented in this thesis, but this must be corroborated with the appropriate experiments.

Algorithm 4: Dynamic training-set class balancing strategy.

$\mathcal{D}_c = \{\Phi_i | y_i = c\}_{i=1}^{N} \subset \mathcal{D}_{\text{training}}$// Collect the light-curves, of class c, into a new auxiliar subset $\mathcal{D}_{(c)}$

$N_{\max} = \max\{N_c = \#(\mathcal{D}_c)\}_{c=1}^{C}$// Find the maximum population $N_{\max}$ associated with the most populated class c

for *epoch* $\in$ *epochs* **do**

  $\mathcal{D}_{\text{training}}^{\text{balanced}} = \{\varnothing\}$// Init a new empty balanced training-set

  for $c \in \{1, \ldots, C\}$ **do**

    for $i \in \{1, \ldots, N_{\max}\}$ **do**

      $\Phi_i' \sim \mathcal{D}_{(c)}$// Randomly choose a light-curve of class c with probability $p = \frac{1}{N_c}$

      $\mathcal{D}_{\text{training}}^{\text{balanced}} \overset{\cup}{\leftarrow} \{\Phi_i'\}$// Append the selected light-curve into the balanced training-set

B.5. Model Input Normalization

For the serial formulation, the model input vectors are normalized as follows:

$$\boldsymbol{x}_{i,j} \leftarrow \frac{\boldsymbol{x}_{i,j} - \text{mean}(\boldsymbol{x})}{\text{std}(\boldsymbol{x}) + \varepsilon}, \tag{B.2}$$

where the mean and standard deviation vectors are computed over the auxiliary vector set $\boldsymbol{x} = \left\{\{\boldsymbol{x}_{1,j}\}_{j=1}^{L_1}, \ldots, \{\boldsymbol{x}_{N,j}\}_{j=1}^{L_N}\right\}$, which is a variable-length vector collection consisting of all the samples from the synthetic training-set $\mathcal{D}_{\text{training[s]}}$.

For the parallel formulation, the normalization follows as follows:

$$\boldsymbol{x}_{i,j^{(b)}} \leftarrow \frac{\boldsymbol{x}_{i,j^{(b)}} - \text{mean}\big(\boldsymbol{x}^{(b)}\big)}{\text{std}(\boldsymbol{x}^{(b)}) + \varepsilon}, \tag{B.3}$$

where the auxiliary vector set is collected, given the band b, as
$\boldsymbol{x}^{(b)} = \left\{\left\{\boldsymbol{x}_{1,j^{(b)}}\right\}_{j^{(b)}=1}^{L_1^{(b)}}, \ldots, \left\{\boldsymbol{x}_{N,j^{(b)}}\right\}_{j^{(b)}=1}^{L_N^{(b)}}\right\}$.

This method is used to normalize the input for both, the encoder model and the decoder model (time difference values). It is also used over the observation-fluxes evaluated in the reconstruction loss shown in section §3.4.1.

B.6. Class Balance Strategy

Algorithm 4 describes the strategy used to dynamically construct an auxiliary and balanced training-set $\mathcal{D}_{\text{training}}^{\text{balanced}}$. This strategy is used during both, the pre-training and fine-tuning processes to deal with the class imbalance.

Algorithm 5: Dynamic data-augmentation strategy for a multi-band light-curve Φ_i.

if model is training **then**
  for $b \in \{1, \ldots, B\}$ **do**
    $\Phi_i^{(b)} \leftarrow f_{\text{lcrss}}\left(\Phi_i^{(b)}\right)$ // LCRSS
    $\Phi_i^{(b)} \leftarrow f_{\text{lcrod}}\left(\Phi_i^{(b)}\right)$ // LCROD
    for $j^{(b)} \in \left\{1, \ldots, L_i^{(b)}\right\}$ **do**
      $\mu_{i,j^{(b)}} \leftarrow \mu_{i,j^{(b)}} + k \cdot \sigma_{i,j^{(b)}} \cdot \varepsilon, \varepsilon \sim \text{t-student}(\nu)$ // LCORE
  $\Phi_i \leftarrow f\left(\left\{\Phi_i^{(1)}, \ldots, \Phi_i^{(B)}\right\}\right)$ // Re-define the multi-band light-curve using the new single-band light-curves
  for $b \in \{1, \ldots, B\}$ **do**
    $t_{i,j^{(b)}} \leftarrow t_{i,j^{(b)}} - t_{i,1}, \forall j^{(b)}$ // Observation-time re-offset

B.7. Multi-Band Light-Curve Data-Augmentation

Algorithm 5 describes the data-augmentation strategy used to dynamically construct new multi-band light-curves during the training processes, allowing us to induce a degree of variability in the light-curves processed by the models. This strategy is used during both, the pre-training and fine-tuning processes.

The data-augmentation explanation is as follows:

1. **Light-Curve Random Sub-Slide (LCRSS)**: given a single-band light-curve $\Phi_i^{(b)}$, a random light-curve sub-slide is selected from $\Phi_i^{(b)}$, re-defining the original single-band light-curve. The sub-slide is performed by randomly sampling both, an initial sequence step and a new variable-length $L_i^{(b)}$.
2. **Light-Curve Random Observation Dropout (LCROD)**: given a single-band light-curve $\Phi_i^{(b)}$, random individual observations are removed from the $\Phi_i^{(b)}$, re-defining the original single-band light-curve. A dropout probability of p_{lcrod} is used.
3. **Light-Curve Observation-flux Re-Estimation (LCORE)**: given a single-band light-curve $\Phi_i^{(b)}$, and following the same method shown in section §3.2.3, a clipped t-student distribution is used to re-sample each observation-flux $\mu_{i,j^{(b)}}$.
4. **Observation-time re-offset**: to avoid ill-defined multi-band light-curves Φ_i, the first observation-time $t_{i,1}$, from the resulting multi-band light-curve, is subtracted from all the observation-times. This is performed to construct new multi-band light-curves where the first observation-time is zero: $t_{i,1} = 0$. This procedure is also performed whenever changes in the observation-times occur, e.g., when applying pre-processing methods (section §3.1.2), when generating synthetic light-curves (section §3.2).

Three main levels of data-augmentation are used in this work: zero, weak, and strong data-augmentation levels. In the zero data-augmentation level, all the aforementioned procedures

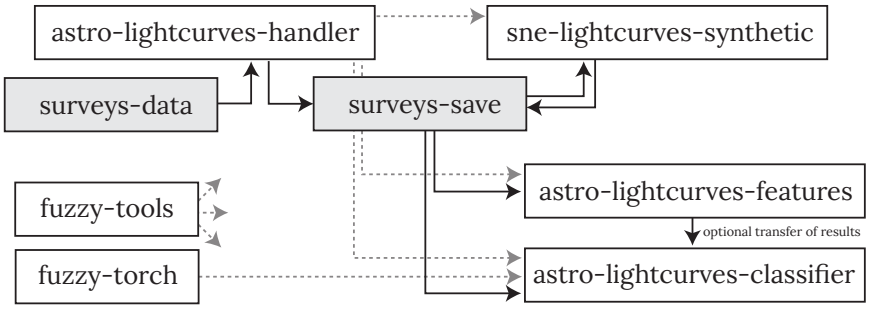

Figure B.2: Relationship between the different repositories implemented for this thesis. Dashed lines indicate the dependencies of packages. Note that fuzzy-tools is used in all the repositories as it is an auxiliary repository. The load and save of data is shown using solid lines. The folders (not repositories) surveys-data and surveys-save are shown, where files (light-curve datasets) are loaded and saved by different repositories.

are ignored, returning the original multi-band light-curves. In the weak data-augmentation level, a dropout probability of $p_{\text{lcrod}} = .1$ is used. In the strong data-augmentation level, a probability of $p_{\text{lcrod}} = .5$ is used. Because some of the data-augmentation procedures remove observations, a minimum single-band light-curve length threshold of $L_i^{(b)} \geq 5$ is imposed as a new augmented light-curve requirement in order to avoid problems of short or empty light-curves.

B.8. Repository Brief Roadmap and Used Software

The following is a list of repositories used for this thesis. In addition, the relationship between the different repositories are shown in Fig. B.2.

1. **fuzzy-tools**: used as an auxiliary repository for multiple and diverse purposes, such as file handling (Pickle-based); plot visualizations (Matplotlib-based); string, list, dictionary, and other data-structures handling; different data science methods; among many others. Url: `https://github.com/oscarpimentel/fuzzy-tools`.

2. **fuzzy-torch**: used as an auxiliary repository for training Pytorch based models. It also contains several basic models, functions, methods, and routines used for the implementation of other Deep Learning models, such as the ones shown in this thesis. Url: `https://github.com/oscarpimentel/fuzzy-torch`.

3. **astro-lightcurves-handler**: used as an auxiliary repository for the creation and handling of multi-band light-curves objects and light-curve datasets. It is also used to apply the pre-processing methods over the light-curve datasets, e.g., sigma clipping, train/-val/test split. Url: `https://github.com/oscarpimentel/astro-lightcurves-handler`.

4. **sne-lightcurves-synthetic**: used for the generation of synthetic SN multi-band light-curves and datasets. Url: `https://github.com/oscarpimentel/sne-lightcurves-synthetic`.

5. **astro-lightcurves-features**: used for the extraction of features from the multi-band light-curves. It is also used for the training of the BRF baseline. Url: `https://github.com/oscarpimentel/astro-lightcurves-features`.

6. **astro-lightcurves-classifier**: used for the implementation and training of all the tested Deep Learning models. Url: `https://github.com/oscarpimentel/astro-lightcurves-classifier`.

Software: Pytorch (Paszke et al., 2019), Jupyter[8], Dask (Rocklin, 2015), Matplotlib (Hunter, 2007), Pandas (Mckinney, 2011), Python[9], Scikit-learn (Pedregosa et al., 2011), Emcee (Foreman-Mackey et al., 2013).

[8] `https://jupyter.org/`.
[9] `https://www.python.org/`.

Annexed C

Experiments and Results

C.1. Relevance Ranking of the BRF model

After the optimization of a BRF model, a relevance ranking can be obtained: an ordered list that indicates the importance that each feature has for the final predictions of the BRF model. Table C.1 shows two examples of relevance rankings for the BRF model. Similar to the findings in Sánchez-Sáez et al. (2021) for the ZTF survey, the top position of the rankings are usually dominated by the SPM features for the classification of transient events such as SNe. Other common features, such as linear trends (e.g., `LinearTrend`), Mexican Hat Power Spectrum (MHPS) features, multi-band features (e.g., `g-r_max`, `g-r_mean`), etc, can also be found at top positions in the ranking.

We highlight the high variability found for the reported relevance rankings, where a different ranking is usually obtained from each BRF model run. This may be produced due to the intrinsic difficulty of the studied ZTF survey (e.g., the high irregular cadence and imbalance for the SN light-curves), but also due to the notably low percentages obtained for the top features ($< 5\,\%$). Additionally and similar to the findings in Sánchez-Sáez et al. (2021), features designed for periodic stars can be found in the relevance rankings, for transient events, as all available features are used, e.g., `Power_rate` (from periodogram). Given that no intrinsic periodic behavior is expected from transient events, those periodic features could be misleading for the BRF model, increasing the risk of overfitting. Therefore, a feature selection method could be utilized to improve the BRF model performance by removing extra and unnecessary features for the SN event.

C.2. 2D Projections for Synthetic Light-Curves

A dimensionality reduction method is performed for visual inspection purposes over the astrophysical and hand-crafted features extracted from both, the empirical and the synthetic SN multi-band light-curves. This reduction is unsupervised and its pipeline steps are

Table C.1: Two examples of relevance rankings obtained from the optimized BRF model. Only the top 30 features of the ranking are shown. For single-band features, the name notation {feature name}_{b} is used to denote in which band b the feature was computed.

Feature name	Rank	Relevance [%]
SPM_t0_g	1	4.563
SPM_gamma_g	2	4.507
SPM_t0_r	3	4.026
g-r_max	4	2.589
first_mag_g	5	2.499
SPM_tau_rise_r	6	2.359
delta_mjd_fid_g	7	2.353
Power_rate_3	8	2.239
Power_rate_4	9	2.049
delta_mjd_fid_r	10	2.042
SPM_beta_r	11	1.936
LinearTrend_r	12	1.933
g-r_mean	13	1.788
MHPS_ratio_g	14	1.766
SPM_gamma_r	15	1.705
Skew_g	16	1.642
Harmonics_mag_1_r	17	1.552
Power_rate_2	18	1.534
first_mag_r	19	1.478
MedianAbsDev_r	20	1.401
LinearTrend_g	21	1.399
iqr_r	22	1.356
SPM_tau_fall_g	23	1.317
SPM_tau_rise_g	24	1.273
MHPS_high_r	25	1.182
Q31_r	26	1.171
Power_rate_1/2	27	1.147
delta_period_r	28	1.072
MHPS_high_g	29	1.028
AndersonDarling_g	30	.981

Feature name	Rank	Relevance [%]
SPM_t0_r	1	3.658
SPM_t0_g	2	3.031
SPM_gamma_g	3	2.882
LinearTrend_r	4	2.753
delta_mjd_fid_g	5	2.363
SPM_tau_rise_r	6	2.302
g-r_max	7	2.023
g-r_mean	8	1.866
first_mag_r	9	1.860
first_mag_g	10	1.735
SPM_gamma_r	11	1.596
SPM_beta_r	12	1.593
SPM_tau_fall_g	13	1.575
Power_rate_4	14	1.535
Power_rate_3	15	1.505
MHPS_ratio_g	16	1.474
MHPS_high_g	17	1.471
delta_mjd_fid_r	18	1.381
Power_rate_2	19	1.359
LinearTrend_g	20	1.314
iqr_r	21	1.294
SPM_tau_fall_r	22	1.123
MHPS_high_r	23	1.119
SPM_tau_rise_g	24	1.066
Skew_g	25	1.015
AndersonDarling_g	26	1.014
Period_band_g	27	.957
delta_mag_fid_r	28	.873
Q31_r	29	.866
SF_ML_amplitude_r	30	.804

described as follows[1]:

1. A quantile normalization[2] is used, transforming the distribution of the input to follow a standard Gaussian distribution.
2. A standard scaling normalization[3] is applied. This step is used to remove the mean from the data, pre-processing the subsequent PCA algorithm.

[1]Note that those pipeline steps are applied sequentially over an arbitrary input vector $\boldsymbol{x}_i$.

[2]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html.

[3]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

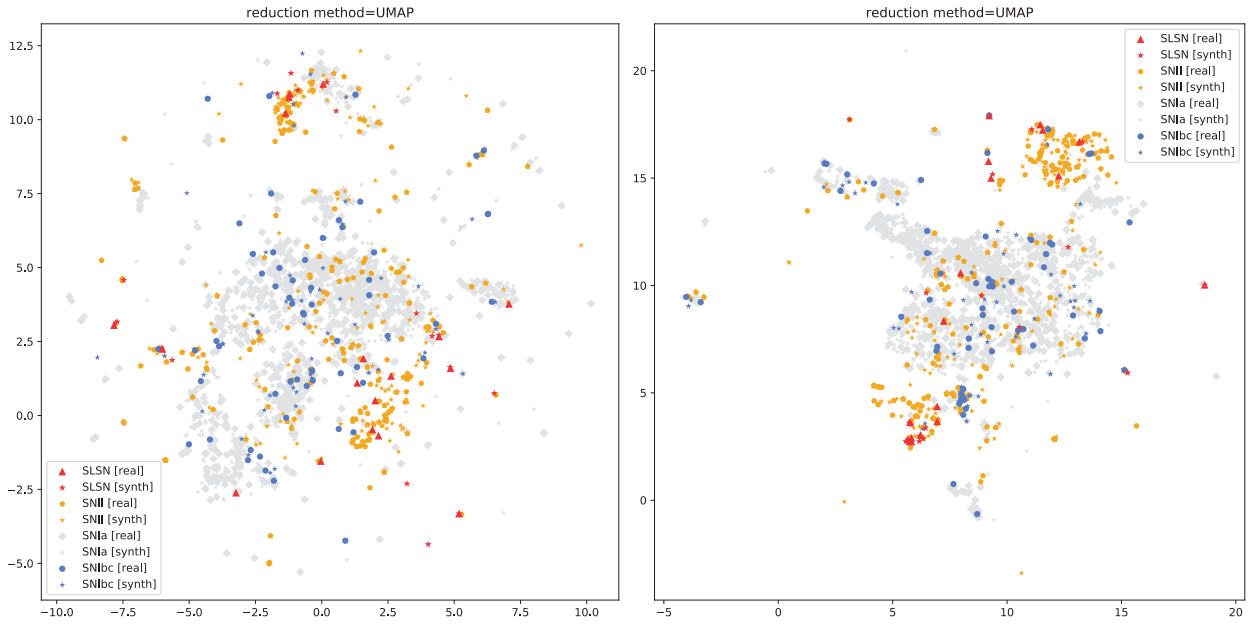

Figure C.1: Examples of dimensionality reductions using astrophysical and hand-crafted features (from the same arbitrary fold split). Both, empirical SN (real) and synthetic SN (synth) are shown. Left) Using all available features. Right) Using only SPM features.

3. A PCA reduction is performed: $f_{\text{pca}}(\boldsymbol{x}_i) : \mathbb{R}^{D_x} \mapsto \mathbb{R}^{10}$, where D_x is the dimensionality of the input vector $\boldsymbol{x}_i$.

4. A UMAP reduction is performed: $f_{\text{umap}}(\boldsymbol{x}_i) : \mathbb{R}^{10} \mapsto \mathbb{R}^{2}$. Empirically, we select the following setting for the UMAP algorithm: {`metric`=`euclidean`, `n_neighbors`= 10, `min_dist`= .01}.

Additionally, we tested two main scenarios for this dimensionality reduction: 1) Using all the available astrophysical features, with a dimensionality of the input vectors of $D_x = 144$. 2) Using only the features associated with the SPM, with a dimensionality of the input vectors of $D_x = 14$[4]. Fig. C.1 shows both, the UMAP reduction when using all the available features and the reduction when only using the SPM features. For comparison purposes, all reductions are obtained from the same arbitrary fold split.

In general, we observe that the use of only SPM features leads to a more class-separable projection w.r.t. the use of all the features, with a notable separation between classes SNIa and SNII. Also, a major part of the SLSN instances are clustered together in a common region. On the other hand, a high sparsity is observed for the SNIbc instances, where only few instances are clustered together. This behavior could correlate with the BRF classification performance. Therefore, we hypothesize that the use of only the SPM features may be beneficial when performing this type of dimensionality reduction tasks for SN light-curves, where meaningless features (e.g., periodic features) could induce a misleading variability that can be detrimental for these unsupervised projection methods.

[4]A fit error feature is also included as implemented in `https://github.com/alercebroker/lc_classifier`.

In order to check for possible outliers and/or incorrect fits in the generation of the synthetic light-curves, we also search for maximum distances in the projected UMAP hyperplane. These distances are defined as euclidean distances between an arbitrary empirical SN multi-band light-curve w.r.t. the associated and generated synthetic light-curves. Fig. C.2 show some empirical-synthetic computed distances for each SN type. Fig. C.3 shows examples of the synthetic generation for some of the reported objects with maximum distance.

By inspecting the high distance objects in the UMAP hyperplane, we found wrong generation fits in cases where the light-curves present a uncommon general behavior (outliers-like light-curves). On the other hand, we also found some SN light-curves with few observations, which could produce instabilities and highly variable results in the extraction of the SPM features. Even if this method could be used as an outlier searching tool, we highlight that this reduction methodology for the search of outliers may be tricky as distances in the projected UMAP reduction can be misleading. In any case, a simpler and linear method, such as a linear PCA reduction, can be also performed.

C.2.1. Limitations

We emphasize that performing this dimensionality reduction is non-trivial for two important reasons. 1) The optimal design of optimization and selection of the hyperparameters of the pipeline is highly non-trivial, leaving plenty of room for experimentation. 2) This reduction methodology is unsupervised; consequently, the class information is not used at any step (e.g., the optimization of the UMAP algorithm). This can be especially detrimental when a high imbalance of the studied dataset is presented, like in this work. In their standard version, these reduction algorithms do not deal with high class imbalances, so a risk of over representation could exist biased towards the majority classes (e.g., SNIa, SNII). This could explain the separability found for the class SLSN, which, while acceptable (as all instances fall into a close and common region), this separability could be even better. Finally, these reductions method must be used with caution in dataset like the one studied in this work, using them as complementary tools.

C.3. Additional Synthetic Light-Curves Generation Methods

For the sake of comparison, additional methods for the generation of synthetic SN light-curves were tested in this work:

1. `mcmc-estw`: it is the main method used and proposed in this work (see section §3.2 for details). This method is based on fitting the SPM function $f_{\text{sne}}\left(t_{i,j^{(b)}}; \boldsymbol{\theta}_i^{(b)*}\right)$ by founding the optimal SPM parameters $\boldsymbol{\theta}_i^{(b)*}$. Then, the generated synthetic observations can be sampled from an extended Sampling Time Window (STW) $t_{i,j^{(b)}} \sim \Delta t_{\text{stw}i}^{(b)}$. This extended STW allows us to sample synthetic observations previous to the first observation.

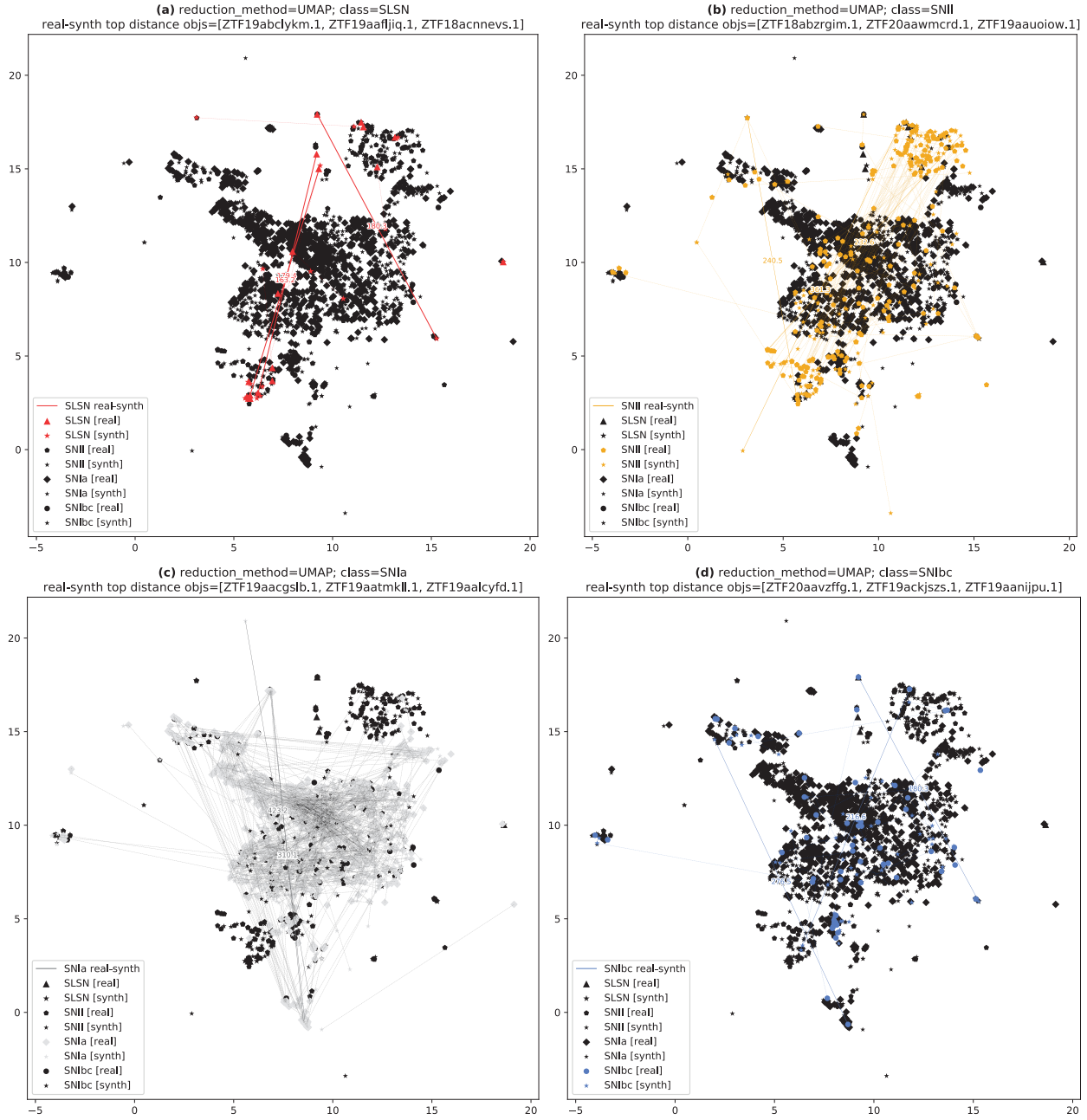

Figure C.2: Examples of dimensionality reductions using astrophysical and hand-crafted features using only SPM features (from the same arbitrary fold split). Both, empirical SN ([real]) and synthetic SN ([synth]) are shown. Some maximum distance relationships, between an arbitrary empirical instance and the related synthetic instances, are shown as line connections (real-synth). (a) SLSN type. (b) SNII type. (c) SNIa type. (d) SNIbc type.

2. `mcmc-fstw`: the same methodology of the `mcmc-estw` is followed but, instead, a fixed STW is used. This fixed STW is defined by fixing $t^{(b)}_{\text{init}i} = t_{i,1^{(b)}}$. In contrast to `mcmc-estw`, in this method, the synthetic observations can only be sampled between the first and last observations of a light-curve.

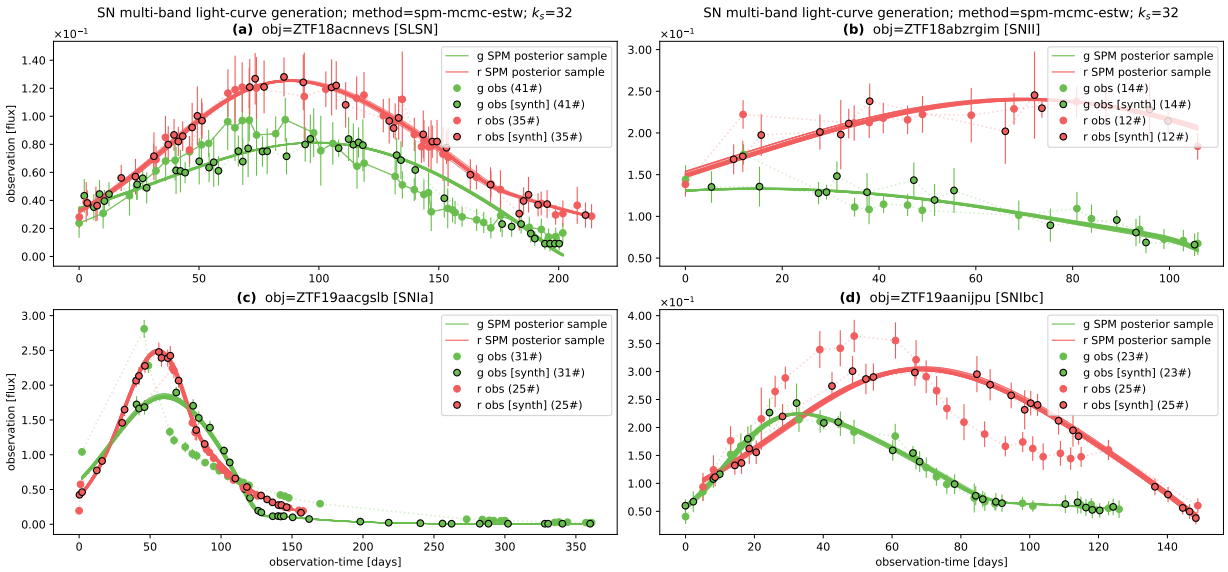

Figure C.3: Examples of synthetic SN multi-band light-curves (before the observation-time re-offset). The SPM posterior samples ($k_s = 32$) from MCMC are shown as continuous-time curves for each band. Empirical observation-fluxes are shown as color circles with observation-error bars. Synthetic observation-fluxes, using a random SPM posterior sample, are shown as black outlined circles. (a) SLSN type. (b) SNII type. (c) SNIa type. (d) SNIbc type.

3. `linear-fstw`: it is a simple linear interpolation[5] of each empirical observation of the light-curve with its subsequent observation. Therefore, this method re-define the function f_{sne} as a linear interpolation that can still be evaluated at any arbitrary time value. To avoid possible artifact synthetic observations, a fixed STW is used.

4. `bspline-fstw`: the B-spline representation[6] of a light-curve is found over the empirical observations of a light-curve, re-defining the function f_{sne} as a smooth and continuous B-spline interpolation. To avoid possible artifact synthetic observations, a fixed STW is used.

Fig. C.4 shows examples for the results of the different synthetic generation methods aforementioned. For comparison purposes, these methods are optimized by using the same arbitrary empirical SN light-curve.

C.3.1. Performance Comparison

To compare the different synthetic generation methods, we study the performance achieved by using the BRF classifier baseline. Additionally and recalling the optimization settings shown in section §3.3, we also propose three BRF optimization settings:

1. **Only empirical data ([r])**: it is the original setting described in section §3.3. Only

[5] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.interp1d.html`.
[6] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.splrep.html`.

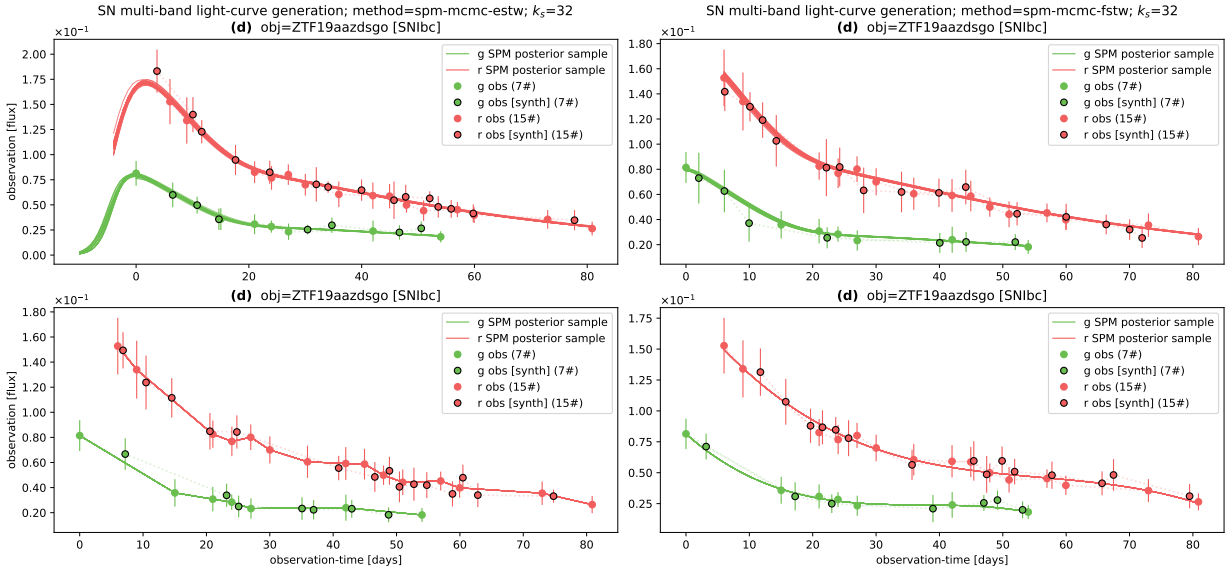

Figure C.4: Examples of synthetic SN multi-band light-curves (before the observation-time re-offset). The SPM posterior samples ($k_s = 32$) from MCMC are shown as continuous-time curves for each band. Empirical observation-fluxes are shown as color circles with observation-error bars. Synthetic observation-fluxes, using a random SPM posterior sample, are shown as black outlined circles. Given the same arbitrary and empirical SN light-curve, the results of different generation methods are shown: `mcmc-estw`, `mcmc-fstw`, `linear-fstw`, `bspline-fstw`. Note that only the methods based on the MCMC optimization present some degree of variability on the estimated function f_{sne}.

empirical light-curves, from the training-set $\mathcal{D}_{\text{training}}$, are used to optimize the BRF baseline.

2. **Only synthetic data ([s])**: in this setting, only the synthetic light-curves, from the training-set $\mathcal{D}_{\text{training[s]}}$, are used to optimize the BRF baseline.

3. **Empirical and synthetic data ([s+r])**: in this setting, light-curves from both, the empirical training-set $\mathcal{D}_{\text{training[s]}}$ and the synthetic training-set $\mathcal{D}_{\text{training[s]}}$, are used to optimize the BRF baseline. The same proportion of each type of light-curve is used; therefore, the each empirical light-curve from the training-set $\mathcal{D}_{\text{training[s]}}$ is repeated a number of $k_s = 32$ times in the new combined training-set $\mathcal{D}_{\text{training[r+s]}}$, i.e., $\mathcal{D}_{\text{training[r+s]}} = k_s \times \{\mathcal{D}_{\text{training}}\} \cup \mathcal{D}_{\text{training[s]}}$.

It should be noted that, for each fold split, exactly the same validation-set $\mathcal{D}_{\text{val}}$ and test-set $\mathcal{D}_{\text{test}}$ are used for all the aforementioned settings, which consist of only empirical light-curves. Therefore, note that the selection of the type of light-curves (empirical or synthetic) only affects the training-set that is used to optimize the BRF baseline.

Table C.2 shows the late-classification performances achieved by training the BRF baseline under the aforementioned training settings ([r], [s], [r+s]) and using the different generation methods (`mcmc-estw`, `mcmc-fstw`, `linear-fstw`, `bspline-fstw`) to build those training-set. In addition, and by following the same methodology implemented for the early-classification in section §4.1.2, Fig. C.5 shows the evolution of the performance for the b-AUCROC metric

Table C.2: Late-classification performances for the BRF baseline using 100-day multi-band light-curves. Results of the TimeModAttn model are also included for both, the serial and the parallel encoders (weak data-augmentation level). All the tested training settings ([r], [s], [r+s]) and different generation methods (`mcmc-estw`, `mcmc-fstw`, `linear-fstw`, `bspline-fstw`) are shown (mean±std from 5-fold cross-validation).

Model	b-Precision	b-Recall	b-F_1score	b-AUCROC	b-AUCPR
Feature-based models					
BRF (fmode=all; training-set=[r])	.527±.030	.687±.052	.525±.039	.866±.020	.602±.051
BRF (fmode=all; training-set=spm-mcmc-estw[s])	.592±.032	.719±.048	.594±.047	.890±.018	.654±.053
BRF (fmode=all; training-set=spm-mcmc-fstw[s])	.591±.030	.719±.045	.595±.043	.890±.018	.655±.051
BRF (fmode=all; training-set=linear-fstw[s])	.583±.012	.732±.028	.586±.021	.891±.017	.652±.037
BRF (fmode=all; training-set=bspline-fstw[s])	.564±.016	.693±.047	.558±.026	.878±.023	.621±.049
BRF (fmode=all; training-set=spm-mcmc-estw[r+s])	.610±.041	.726±.050	.619±.050	.894±.016	.646±.054
BRF (fmode=all; training-set=spm-mcmc-fstw[r+s])	.612±.045	.725±.052	.621±.053	.894±.016	.647±.055
BRF (fmode=all; training-set=linear-fstw[r+s])	.597±.027	.722±.047	.608±.037	.894±.017	.646±.050
BRF (fmode=all; training-set=bspline-fstw[r+s])	.585±.030	.705±.045	.593±.036	.887±.018	.637±.052
Attention-based models					
S-TimeModAttn (M=12; H=8; ε_t=6/24)	.588±.023	.759±.040	.596±.033	.910±.020	.671±.056
P-TimeModAttn (M=12; H=4; ε_t=6/24)	.580±.020	.753±.044	.594±.035	.911±.017	.689±.047

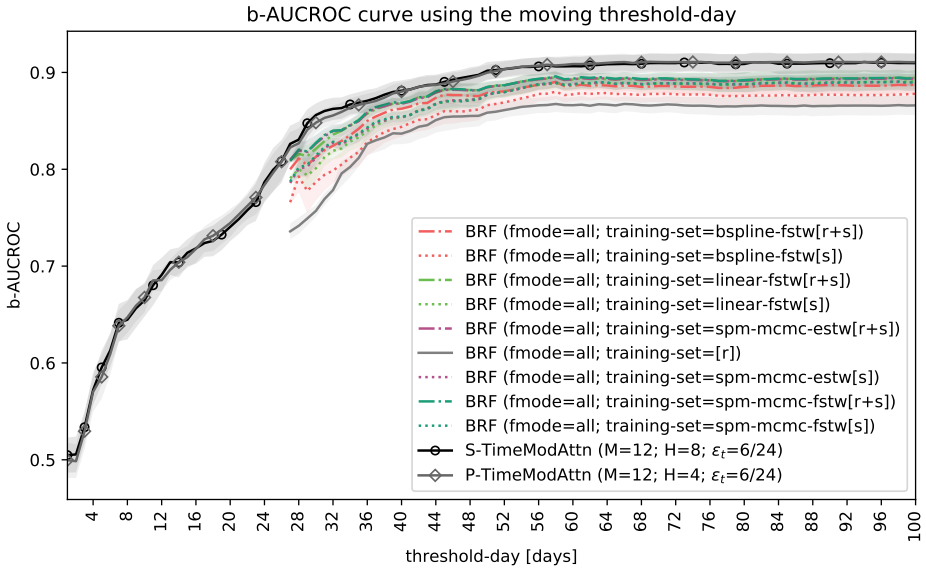

Figure C.5: b-AUCROC metric-curve v/s moving threshold-day for different training settings of the BRF baseline. Results of the TimeModAttn model are also included for both, the serial and the parallel encoders (weak data-augmentation level). All the tested training settings ([r], [s], [r+s]) and different generation methods (`mcmc-estw`, `mcmc-fstw`, `linear-fstw`, `bspline-fstw`) are shown. For the sake of better visualization, mean$\pm\frac{1}{10}$std curve and region are shown from 5-fold cross-validation. Note that the horizontal axis (evolution of the threshold-day) relates with the observation-time since the first observation.

as a function of the moving threshold-day t_{th}.

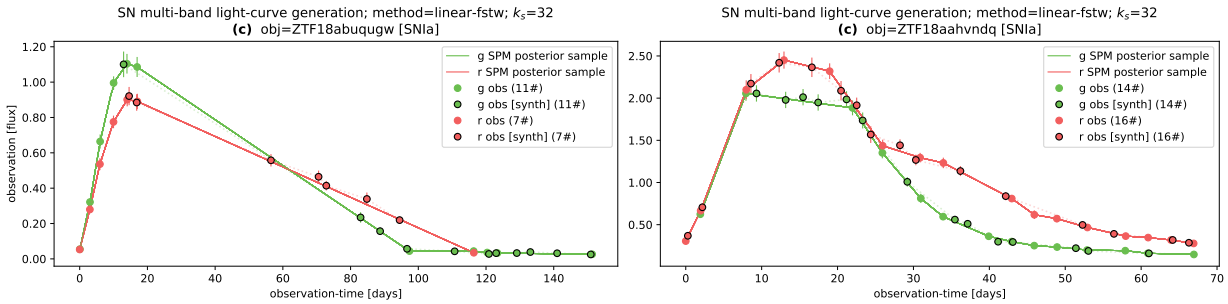

Figure C.6: Examples of synthetic SN multi-band light-curves using the linear interpolation method: `linear-fstw` method (before the observation-time re-offset). The SPM posterior samples ($k_s = 32$) from MCMC are shown as continuous-time curves for each band. Empirical observation-fluxes are shown as color circles with observation-error bars. Synthetic observation-fluxes, using a random SPM posterior sample, are shown as black outlined circles.

From our experiments, we observe that the use of any type of generation method achieved a higher performance for both, the late-classification scenario and the early-classification scenario (according to the b-AUCROC metric). In general, it is not possible to firmly conclude, for all the tested metrics, which BRF optimization setting is the best: the only synthetic data ([s]) or the empirical and synthetic data ([r+s]). On the other hand, a highly competitive performance can be observed between three generation methods: `spm-mcmc-estw`, `spm-mcmc-fstw`, and `linear-fstw`. Contrary to our original conjecture, we found that the use of the ESTW does not necessarily relate with an increment in the overall classification performance. In any case, extra experiments could be designed as future work to further study the effect of the Sampling Time Window (STW).

For the `linear-fstw` method, a highly competitive performance was found w.r.t. the SPM-based methods. This was unexpected because the linear interpolation still can produce artifacts in the behavior of the light-curve. Fig. C.6 shows examples of light-curves generated with the `linear-fstw` method. In particular cases, it can be observed how missing observations can induce different type of artifacts, e.g., missing observations in the SN-peak can produce a «flat SN-peak», missing observations in the SN-rise and SN-fall regions can produce linear rising and/or falling behaviors. Even if our initial conjecture was that this type of artifacts could affect the performance, we empirically found that this is not the case: the model still can correctly perform a classification task, even if these types of artifacts are presented in the training-set. As future work, more experiments could be conducted to further study this phenomena, testing at which degree the linear interpolation starts being detrimental[7].

On the other hand, the `bspline-fstw` method achieved the worst general performance for both classification scenarios. This may be due to notable artifact behaviors presented in the generated light-curves. To illustrate this, Fig. C.7 shows examples of the generated light-curves for the `bspline-fstw` method, where an incorrect final rising could be observed

[7]As a side note, we highlight that the linear interpolation could be specially detrimental for other type of astronomical events such as periodic stars, where the period could be radically altered when interpolating between observations that are too separated in time.

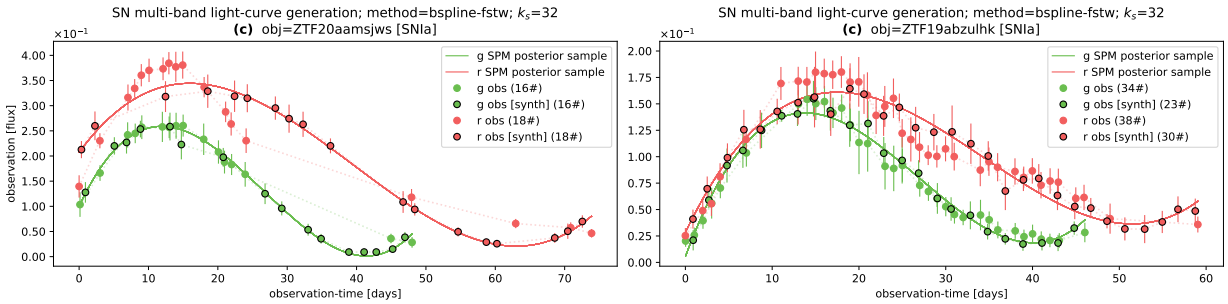

Figure C.7: Examples of synthetic SN multi-band light-curves using the B-spline interpolation method: `bspline-fstw` method (before the observation-time re-offset). The SPM posterior samples ($k_s = 32$) from MCMC are shown as continuous-time curves for each band. Empirical observation-fluxes are shown as color circles with observation-error bars. Synthetic observation-fluxes, using a random SPM posterior sample, are shown as black outlined circles.

close to the SN-extinction region. This effect could be misleading given the context of the SN event.

C.3.1.1. Comparison Against the TimeModAttn Model

To further study the late-classification scenario performance, for the BRF baseline settings w.r.t. the TimeModAttn model, Table C.3[8] includes the additional BRF baseline setting training with only synthetic data ([s]). Additionally, Fig. C.8 shows the confusion matrices associated with all the previous tested settings for the BRF baseline. We can observe that the confusion matrices for the TimeModAttn models (previously shown in section §4.1.3) show a general higher and well-distributed TP percentage along the diagonal, ensuring a good performance for all classes. This is important because all the classes must be, in principle, equally relevant in the classification task.

We can also further study the evolution of the performance w.r.t. the general number of observations in the test-set (by using the moving threshold-day). As shown before, Fig. C.5 shows the evolution of the performance, of the BRF baseline settings, for the b-AUCROC metric as a function of the moving threshold-day t_{th}. Note that we also include the best configuration of the TimeModAttn model. Additionally, Fig. C.9 shows the same type of evolution but in the frame of the Deep Learning models. We observe that the TimeModAttn model achieved higher performance w.r.t. to the BRF baseline settings in both: early-classification and late-classification.

In general, we highlight the fact that the use of synthetic data for training helps in the performance for both, the TimeModAttn model and the BRF baselines. In fact, although the TimeModAttn model shows a general better overall performance than all the tested BRF settings (early and late-classification), it is not possible to firmly conclude that this model is better for all the metrics used in this thesis (when using synthetic data for training). In any case, the use of the proposed model still has advantages over the use of feature-based models

[8]It is essentially a copy of Table 4.1.

Table C.3: Late-classification performances for the TimeModAttn model and the RNN baselines using 100-day multi-band light-curves. Results of the BRF baseline are also included for: 1) training using only real data ([r]), and 2) training using only synthetic data ([s]). Both, the serial (S-{}) and parallel (P-{}) encoders are reported along with several pre-training and data-augmentation schemes (mean±std from 5-fold cross-validation).

Model	b-Precision	b-Recall	b-F_1score	b-AUCROC	b-AUCPR
Feature-based models					
BRF (fmode=all; training-set=[r])	.527±.030	.687±.052	.525±.039	.866±.020	.602±.051
BRF (fmode=all; training-set=spm-mcmc-estw[s])	.592±.032	.719±.048	.594±.047	.890±.018	.654±.053
Serial models	Empirical pre-training (zero data-augmentation)				
S-RNN+Δt (cell=GRU)	.520±.043	.626±.050	.528±.039	.852±.021	.577±.049
S-RNN+Δt (cell=LSTM)	.497±.030	.602±.044	.502±.034	.840±.019	.568±.031
S-TimeModAttn (M=12; H=8; ε_t=6/24)	.551±.034	.664±.058	.565±.040	.874±.024	.597±.036
	Synthetic pre-training (zero data-augmentation)				
S-RNN+Δt (cell=GRU)	.562±.051	.688±.058	.579±.049	.885±.036	.627±.062
S-RNN+Δt (cell=LSTM)	.561±.035	.680±.053	.578±.040	.884±.028	.619±.046
S-TimeModAttn (M=12; H=8; ε_t=6/24)	**.598±.030**	.736±.056	**.614±.036**	.904±.029	.665±.060
	Synthetic pre-training (weak data-augmentation)				
S-RNN+Δt (cell=GRU)	.545±.034	.706±.070	.556±.045	.879±.034	.610±.066
S-RNN+Δt (cell=LSTM)	.550±.031	.711±.070	.558±.040	.887±.033	.621±.070
S-TimeModAttn (M=12; H=8; ε_t=6/24)	.588±.023	**.759±.040**	.596±.033	.910±.020	**.671±.056**
	Synthetic pre-training (strong data-augmentation)				
S-RNN+Δt (cell=GRU)	.491±.024	.649±.068	.496±.036	.860±.032	.561±.063
S-RNN+Δt (cell=LSTM)	.497±.021	.657±.066	.494±.028	.864±.031	.565±.055
S-TimeModAttn (M=12; H=8; ε_t=6/24)	.582±.017	.754±.039	.584±.031	**.911±.019**	.665±.053
Parallel models	Empirical pre-training (zero data-augmentation)				
P-RNN+Δt (cell=GRU)	.521±.042	.613±.042	.527±.044	.849±.013	.561±.032
P-RNN+Δt (cell=LSTM)	.497±.034	.604±.049	.500±.041	.834±.016	.548±.027
P-TimeModAttn (M=12; H=4; ε_t=6/24)	.543±.026	.671±.053	.562±.029	.865±.022	.599±.038
	Synthetic pre-training (zero data-augmentation)				
P-RNN+Δt (cell=GRU)	.566±.038	.685±.056	.582±.040	.883±.027	.624±.047
P-RNN+Δt (cell=LSTM)	.567±.029	.683±.041	.580±.036	.881±.027	.645±.048
P-TimeModAttn (M=12; H=4; ε_t=6/24)	**.591±.021**	.729±.038	**.610±.026**	.897±.023	.676±.059
	Synthetic pre-training (weak data-augmentation)				
P-RNN+Δt (cell=GRU)	.547±.030	.697±.070	.552±.041	.879±.031	.610±.055
P-RNN+Δt (cell=LSTM)	.541±.022	.704±.061	.540±.032	.876±.029	.606±.051
P-TimeModAttn (M=12; H=4; ε_t=6/24)	.580±.020	**.753±.044**	.594±.035	**.911±.017**	**.689±.047**
	Synthetic pre-training (strong data-augmentation)				
P-RNN+Δt (cell=GRU)	.490±.020	.645±.057	.482±.024	.856±.032	.577±.064
P-RNN+Δt (cell=LSTM)	.499±.020	.660±.061	.484±.031	.857±.031	.573±.053
P-TimeModAttn (M=12; H=4; ε_t=6/24)	.581±.019	.750±.039	.585±.036	.907±.016	.679±.043

such as the BRF baseline:

1. It is worth remembering that the computation and extraction of features, from the

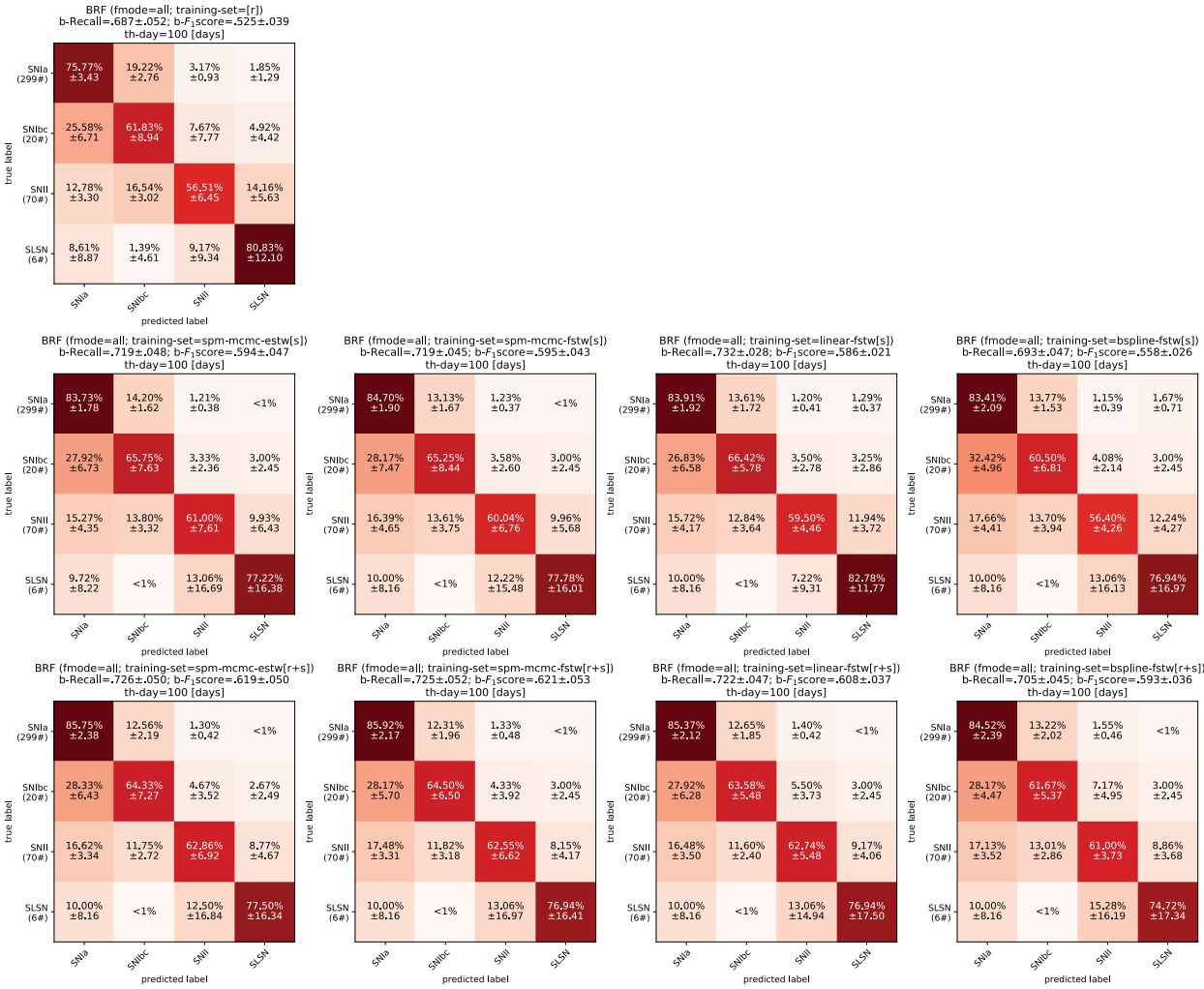

Figure C.8: Confusion matrices for the SNe classification task using 100-day multi-band light-curves. All the tested training settings ([r], [s], [r+s]) and different generation methods (`mcmc-estw`, `mcmc-fstw`, `linear-fstw`, `bspline-fstw`) are shown (mean±std from 5-fold cross-validation). The corresponding b-Recall and b-F_1score metrics are given on the top of each matrix.

light-curves, is mandatory when using the BRF model. Therefore, in order to optimize the BRF using synthetic light-curves, the features of the latter are required. Because the computation of features is a costly process (CPU-based), the optimization of the BRF model using synthetic light-curves incurs in an even higher computational cost w.r.t. the use of only empirical light-curves.

2. Note that including light-curves obtained with a data-augmentation method (such as the one described in section §B.7) may be too demanding, incurring in an even higher higher computational cost due to the feature computation: the features of every possible augmented light-curve must be extracted.

3. It should be remembered that the inference of the Deep Learning models is straightforward for short light-curves with one o few observations. This is not the case for the feature-based BRF model as the features extracted with one or few observations is

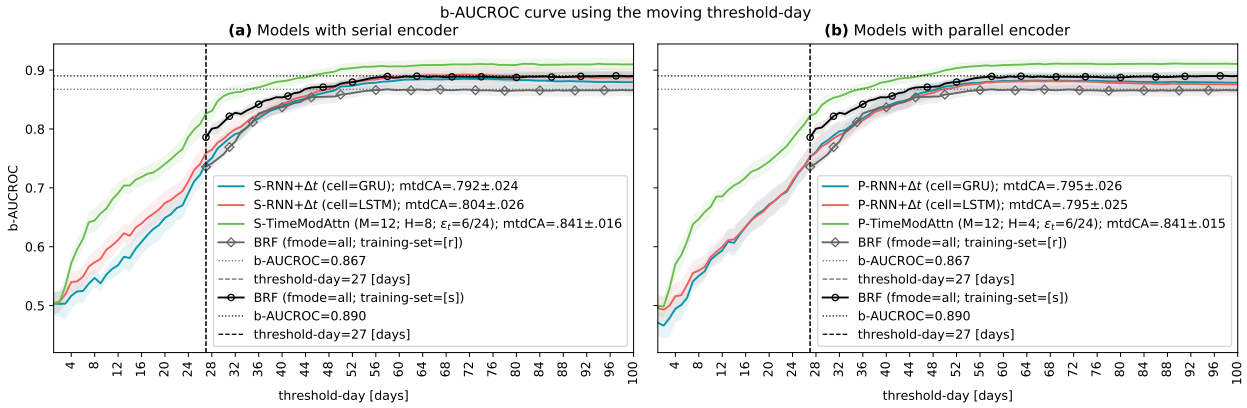

Figure C.9: b-AUCROC metric-curve v/s moving threshold-day for the TimeModAttn model and the RNN baselines (weak data-augmentation level). Results of the BRF baseline are also included for: 1) training using only real data ([r]), and 2) training using only synthetic data ([s]). For the sake of better visualization, mean$\pm\frac{1}{2}$std curve and region are shown from 5-fold cross-validation. The moving threshold-day Curve Average (mtdCA) is shown for the TimeModAttn model and RNN baselines. Note that the horizontal axis (evolution of the threshold-day) relates with the observation-time since the first observation. (a) Models with serial encoder. (b) Models with parallel encoder.

highly unstable. This effect can not be solved by using synthetic light-curves.

4. The use of features still heavily depends on expert knowledge which is a non-trivial and costly task (and may even induce undesired expert biases). This task, moreover, should always be subject to revision, especially if new types of astronomical objects need to be studied.

Even though the reported metrics were useful in this work (b-Precision, b-Recall, b-F_1score, b-AUCROC, b-AUCPR), we must highlight that it may be still tricky to draw solid conclusions by using this type of balanced metrics. For example, in some cases, a high b-F_1score metric may not directly relate with a well-distributed TP percentages in the diagonal of the confusion matrix. Therefore, as future work, a more suitable and robust set of balanced metrics could be explored to better study and compare the performance of the implemented models. This is especially important in the current context due to the critical class imbalance.

Finally, we highlight that the previous comparison using different generation methods was not performed using the Deep Learning framework and models. It was mainly due to the excessive computational costs associated with testing so many possible different configurations of methods and Deep Learning models. In any case, we selected the `mcmc-estw` method as it has a theoretical SN background (SPM) and it is the only method that allows us to extend the generation of observations to early days (offering us a better opportunity to further explore early observations). Nevertheless, we hypothesize that the final selection of the generation method may not be too critical as the proposed Deep Learning framework uses the synthetic light curves in a pre-training step. Therefore, possible domain gaps (between synthetic and empirical data) may still be fixed in the following fine-tuning step with empirical light-curves only.

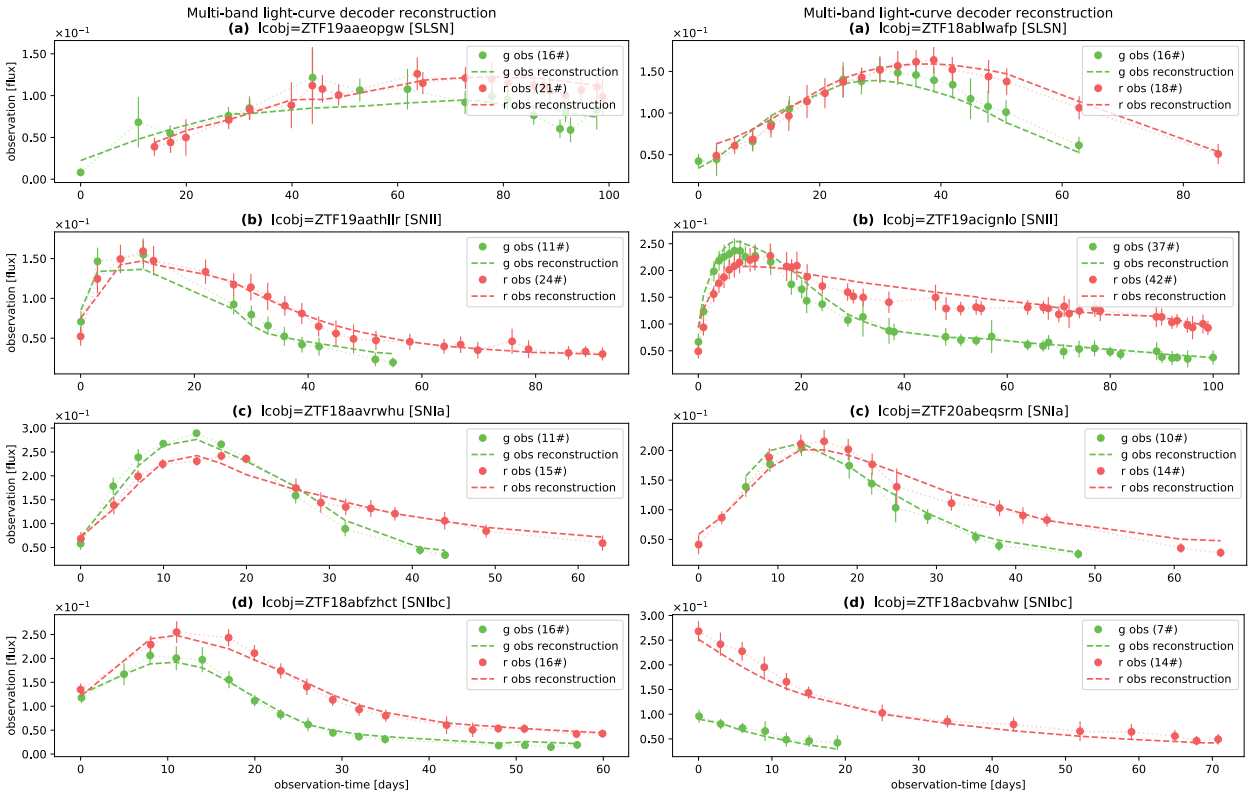

Figure C.10: Examples of multi-band light-curve reconstructions for the P-TimeModAttn model after the pre-training process. Dashed lines are used for the reconstructed light-curves. (a) SLSN type. (b) SNII type. (c) SNIa type. (d) SNIbc type.

C.4. Multi-band Light-Curve Reconstruction

Fig. C.10 shows examples of multi-band light-curve reconstructions for different SN types using the TimeModAttn model. These examples show that the decoder can correctly estimate, given the representation-vector $\boldsymbol{z}_i$ generated by the encoder, the observation-fluxes for the reconstruction of SN multi-band light-curves. This reconstruction is well-performed despite the existence of long time gaps without any observation due to the irregular cadence.

It is worth noticing that the representation of the time information, handled by the TimeModAttn model, is different for the encoder and decoder. For the encoder, a temporal modulation based on the raw time values is induced in the input; while for the decoder, the time difference values are used as the input. In general, this dual representation of the time values produces a highly challenging scenario for the learning of time dependencies, forcing the autoencoder model to transform raw time value representations into time difference representations. Given a correct optimization process, this dual time representation may ultimately lead to a highly meaningful representation space for both, the reconstruction task and the classification task. This could explain the high classification performance obtained by the TimeModAttn model, but further studies and experiments must be performed, which are left for future work.

C.5. 2D Projections for TimeModAttn Model

A dimentionality reduction method is performed for visual inspection purposes over an arbitrary embedding extracted from the TimeModAttn model. This reduction is unsupervised and its pipeline steps are described as follows[9]:

1. A standard scaling normalization[10] is applied. This step is used to remove the mean from the data, pre-processing the subsequent PCA algorithm.
2. A PCA reduction is performed: $f_{\text{pca}}(\boldsymbol{x}_i) : \mathbb{R}^{D_x} \mapsto \mathbb{R}^{10}$, where D_x is the dimensionality of the input vector $\boldsymbol{x}_i$.
3. An optional min-max scaling normalization[11] is applied. This pipeline step is mainly used to avoid warnings from the API of the Parametric UMAP algorithm.
4. A Parametric UMAP reduction is performed: $f_{\text{umap}}(\boldsymbol{x}_i) : \mathbb{R}^{10} \mapsto \mathbb{R}^{2}$. Empirically, we select the following setting for the Parametric UMAP algorithm: {`metric`=`euclidean`, `n_neighbors`= 50, `min_dist`= .01}.

We use the Parametric UMAP because it allows us to solve the intrinsic stochasticity of the UMAP algorithm, i.e., the Parametric UMAP correctly produces the same output embeddings given the same data inputs. This is relevant when reducing the dimensionality of vectors, extracted from an arbitrary layer of the TimeModAttn model, when a higher number of observations is gradually available in the test-set $\mathcal{D}_{\text{test}}$, such as the methodology followed in the early-classification scenario (in section §4.1.2). In this scenario, several light-curves may not change given a small change in the moving threshold-day; therefore, the Parametric UMAP allows us to solve the stochasticity problem when those cases occur (this is especially useful to create stable animations of the projections).

We study two embeddings extracted from the TimeModAttn model: 1) The representation-vector $\boldsymbol{z}_i$ (output of the encoder). 2) The last high-dimensional embedding of the classifier (before the last linear projection with the softmax function activation). In both cases, the dimensionality of the input vectors, for the reduction method, are $D_x = 128$. Fig. C.11 and Fig. C.12 show the 2D projections of the representation-vectors and the last classifier's embeddings, respectively. For comparison purposes, both reductions are obtained from the same arbitrary fold split and model run.

For both sequences of projections, we can observe how both, the sparsity of the instances and the class separability, gradually increase with the increment of the moving threshold-day:

1. **Case** $t_{\text{th}} = 1$ [days]: an initial collapse of the representation can be observed, which is expected as the information of the light-curves is minimal.

[9] Note that those pipeline steps are applied sequentially over an arbitrary input vector $\boldsymbol{x}_i$.

[10] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`.

[11] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html`.

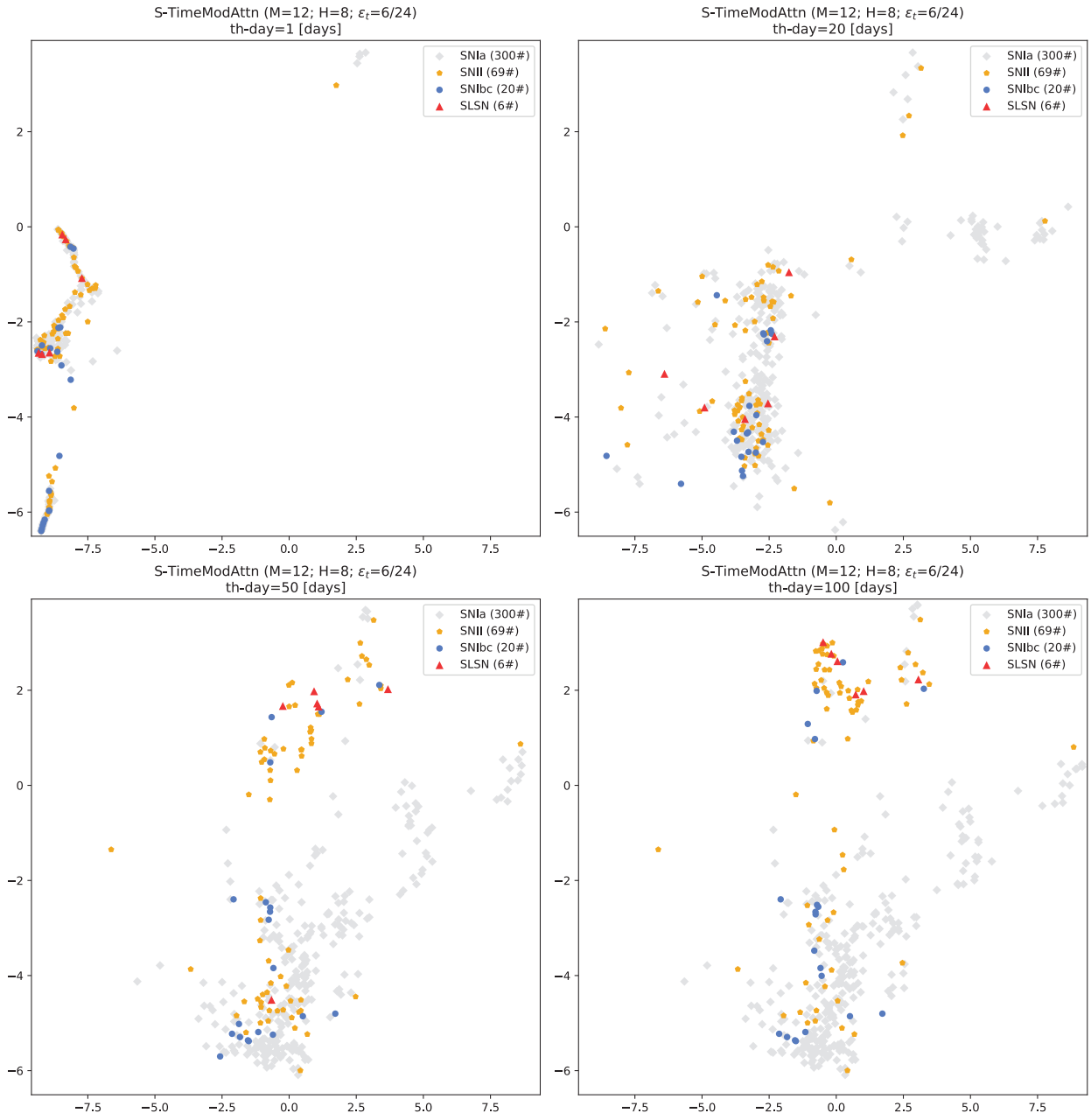

Figure C.11: Examples of dimensionality reductions for the S-TimeModAttn model using the representation-vectors $\boldsymbol{z}_i$ (from the same arbitrary fold split and model run). Several moving threshold-day values are shown: 1, 20, 50, and 100 [days]. See the complete animation in `https://drive.google.com/file/d/1SBOHeEC5mZ5BvIhHfGfMY_V6kl7oEiqp/view?usp=sharing`.

2. **Case** $t_{\text{th}} = 20$ [days]: the projections gradually start to sparse as more information about the light-curve is fed to the model.

3. **Case** $t_{\text{th}} = 50$ [days]: a stable and sparse configuration of the projections starts taking form. For the representation-vector case (Fig. C.11), a class separability starts emerging. In contrast, for the last classifier's embedding (Fig. C.12), this class separability is more evident, showing a notable separation between the classes SNIa, SNII, and SNIbc.

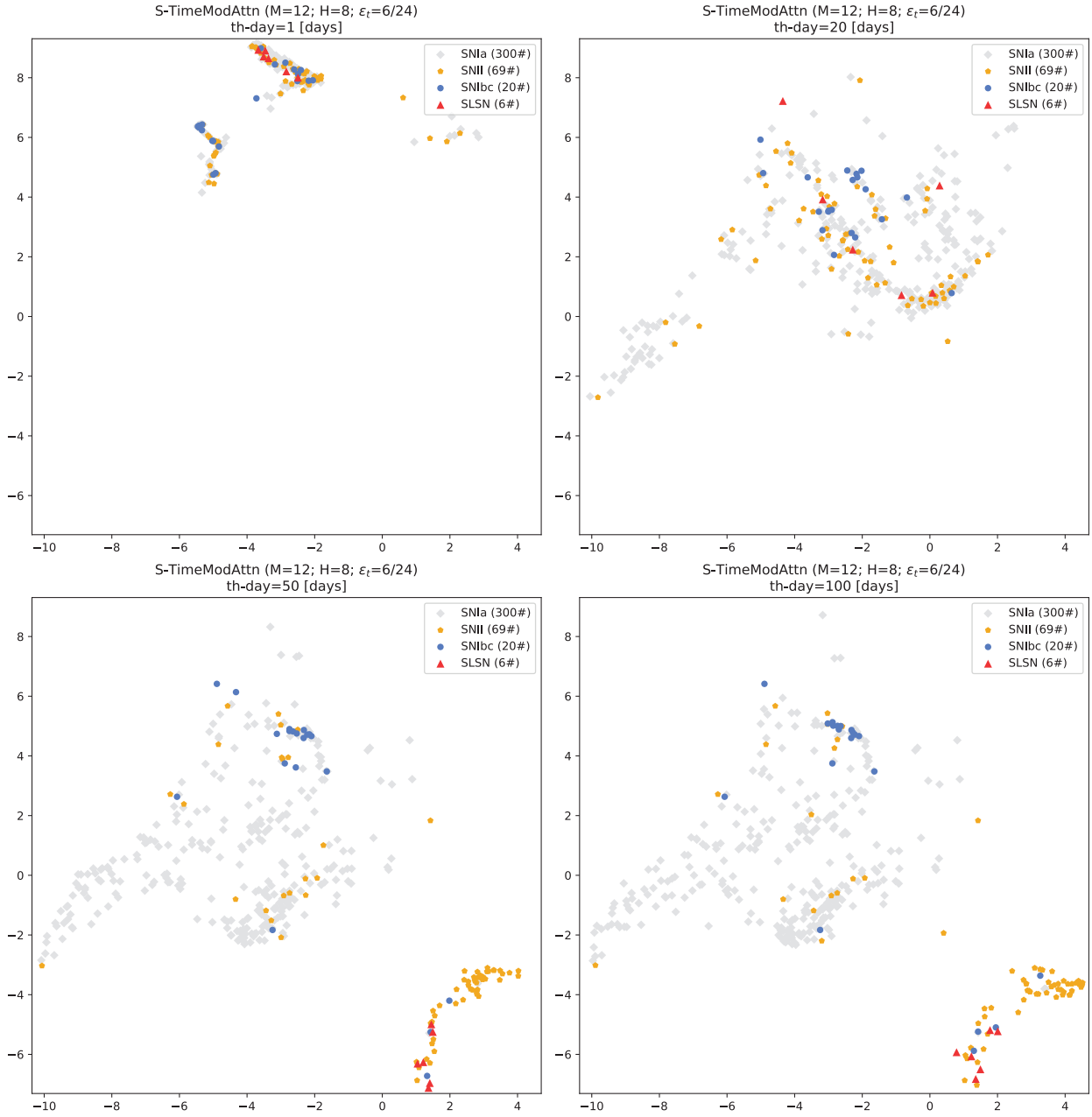

Figure C.12: Examples of dimensionality reductions for the S-TimeModAttn model using the last high-dimension embeddings of the classifier (from the same arbitrary fold split and model run). Several moving threshold-day values are shown: 1, 20, 50, and 100 [days]. See the complete animation in `https://drive.google.com/file/d/1dMsFKz--PfxH7FwXIGPu1vJk-5XKxrn4/view?usp=sharing`.

4. **Case** $t_{\text{th}} = 100$ [days]: the final configuration of the projections is reached. For the representation-vector case (Fig. C.11), a notable separation between the classes SNIa and SNII can be observed. For the last classifier's embedding (Fig. C.12), a small difference w.r.t. $t_{\text{th}} = 50$ [days] projection is observed, slightly improving the previous class separability.

In general, the class separability for the last classifier's embeddings is more pronounced and, moreover, starts to appear with smaller values of t_{th}. This can be explained as the representation-vector shares information to solve both: a classification task and a reconstruction task, entangling the information for just the class separability. On the other hand, the last classifier's embedding, processed from the representation-vector, only handles information to solve the classification task; therefore, all the information may be useful for the class separability. However and despite the difficulty of the case, we still highlight the class separability obtained by using the representation-vectors, especially between the classes SNIa and SNII.

Finally, we highlight that performing this dimensionality reduction is non-trivial for two main reasons: the selection of hyperparameters and the class imbalance. For more details, see the discussion in section §C.2.1.

C.6. Ablation Studies and Additional Models

C.6.1. Number of Fourier Decomposition Harmonics and Attention Heads

To study the influence and relevance of key components of the TimeModAttn model, the following architecture settings are proposed:

1. **Case** $M = 0$: by recalling eq. (3.11), a fully invariant temporal modulation is directly induced by setting an invariant scale and bias time-functions: $\gamma_k(t) = 1 \wedge \beta_k(t) = 0, \forall k \in \{1, \ldots, K\}, \forall t$. This setting allows the encoder to still access the observation-flux information of the light-curves, but without any meaningful information of the observation-time.
2. **Case** $H = 0$: by recalling eq. (2.86), the self-attention mechanism, in the MHSelfAttn layers, is bypassed by imposing $\boldsymbol{c}_{i,j} = \vec{0}$. Therefore, in this case, the encoder only can access to the information of the last observation-flux and the last observation-time of the light-curves, as the last observation is still modulated by the TimeFiLM.
3. **Case** $M = 0$; $H = 0$: in this extreme case, the encoder can only access to the last observation-flux information from the light-curves, but without the last observation-time information.

Table C.4 shows the results associated with the aforementioned architecture settings. Note that, in all settings, the achieved performance is notably lower w.r.t. the BRF baseline performance, suggesting that the temporal modulation and the attention mechanism are required and critical components for the TimeModAttn model.

The use of a fully invariant temporal modulation in the encoder ($M = 0$) is detrimental for the performance because the irregular cadence (temporal information), from the light-curves, are completely lost for the encoder. Nevertheless, a degree of classification can still

Table C.4: Late-classification performances for the BRF baseline and several ablation case studies for the attention-based models (TimeModAttn) using 100-day multi-band light-curves. Both, the serial (S-{}) and parallel (P-{}) encoders are reported (mean±std from 5-fold cross-validation).

Model	b-Precision	b-Recall	b-F_1score	b-AUCROC	b-AUCPR
BRF (fmode=all; training-set=[r])	.527±.030	.687±.052	.525±.039	.866±.020	.602±.051
Serial models					
S-TimeModAttn (M=0; H=0; ε_t=6/24)	.291±.046	.333±.043	.194±.052	.575±.034	.305±.020
S-TimeModAttn (M=0; H=8; ε_t=6/24)	.410±.027	.561±.058	.386±.024	.787±.036	.480±.032
S-TimeModAttn (M=12; H=0; ε_t=6/24)	.337±.014	.379±.042	.307±.016	.698±.028	.365±.018
Parallel models					
P-TimeModAttn (M=0; H=0; ε_t=6/24)	.304±.015	.328±.063	.163±.018	.571±.037	.294±.015
P-TimeModAttn (M=0; H=4; ε_t=6/24)	.401±.018	.547±.042	.375±.021	.773±.029	.469±.032
P-TimeModAttn (M=12; H=0; ε_t=6/24)	.366±.021	.453±.050	.326±.024	.724±.017	.383±.017

be achieved as the encoder can access to the observation-flux information from the complete light-curves. In fact, the attention mechanism could learn to compute flexible statistics from the observation-flux distribution even if no temporal information is available. For example, the attention mechanism could learn to compute the mean and standard deviation of the observation-flux distribution, weighted sums to ignore the SN-extinction region, getting information about the SN-peak by paying attention to the maximum observation-fluxes, etc.

On the other hand, the bypass of the attention mechanism in the encoder ($H = 0$) results in a even worst classification performance as only the information of the last observation is available for the encoder. This confirms that the use of the attention mechanism is required to capture time dependencies from the light-curves and, therefore, meaningful information from the astronomical event. Note that a degree of classification can still be achieved even in this case.

To further study the aforementioned phenomenon, Fig. C.13 shows the confusion matrices for the TimeModAttn model. For both encoders, serial and parallel, a high TP percentage is achieved for the SNIa class. This could be explained by observing the joint distribution in Fig. C.14, where a large number of SNIa instances are clustered together in a common and sparse region due to the high variability of the time duration of their light-curves, w.r.t. other classes. Therefore, a Deep Learning model could be able to trace an hyperplane separation to correctly discriminate a large number of SNIa instances.

As expected, the combination of both cases ($M = 0$; $H = 0$) produces a model that achieved the worst classification performance. This could be explained as the information obtained from the light-curves, and handled by the encoder, is minimal and insufficient to correctly characterized the light-curves and, therefore, to solve the classification task. As an extra note, the worst possible performance (completely random classification) can be achieved by also inducing $\boldsymbol{x}_{i,j} \leftarrow \vec{0} \odot \boldsymbol{x}_{i,j}$ in the input vector of the modulation operation, which results in a dummy encoder.

Finally, it is worth mentioning that these types of ablation studies, usually called «sanity

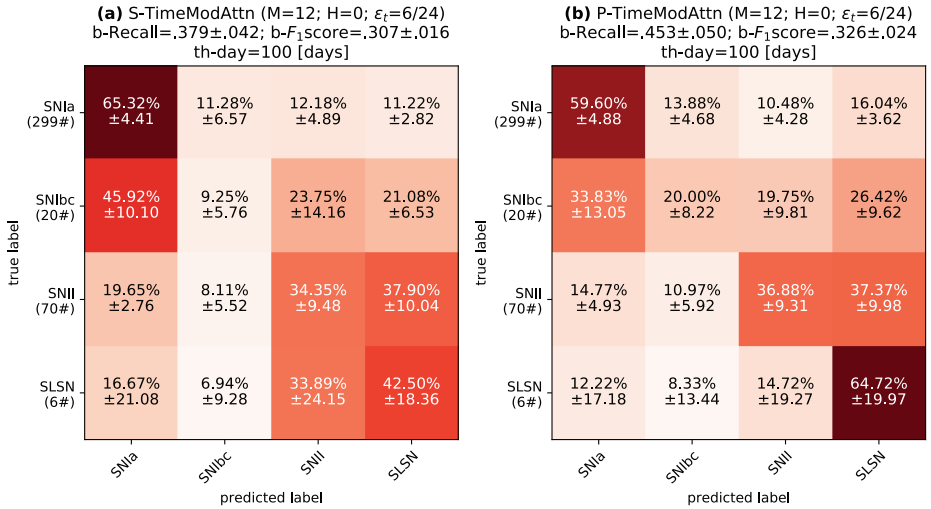

Figure C.13: Confusion matrices for the SNe classification task using 100-day multi-band light-curves when bypassing the attention mechanism ($H = 0$) (mean±std from 5-fold cross-validation). The corresponding b-Recall and b-F_1score metrics are given on the top of each matrix. (a) S-TimeModAttn model. (b) P-TimeModAttn model.

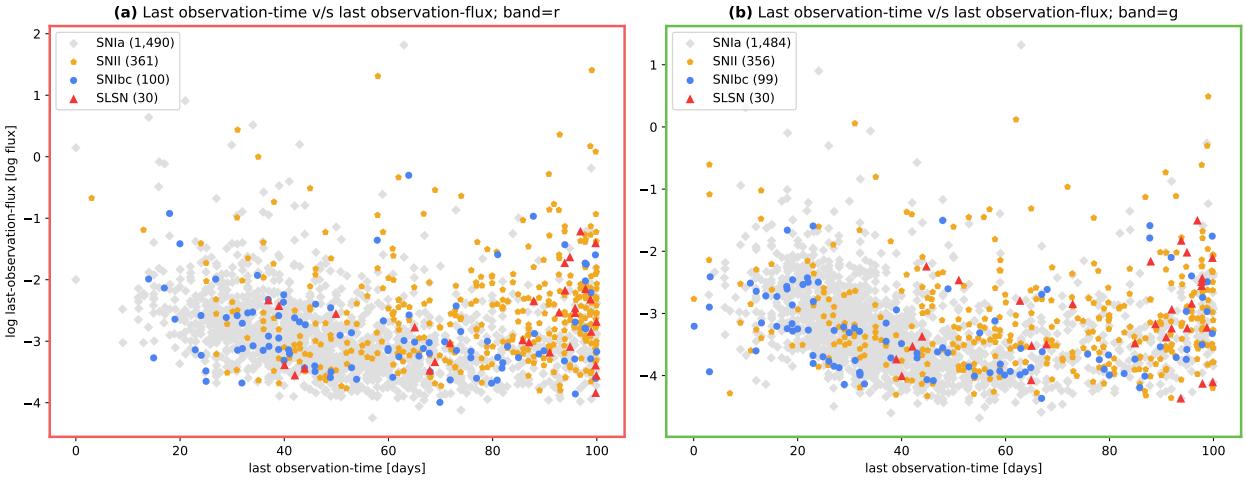

Figure C.14: The joint distribution of the last observation-time versus the last observation-flux (from the original dataset $\mathcal{D}$). Note that the observations associated with empty single-band light-curves ($L_i^{(b)} = 0$) are removed. Green border (a) corresponds to the band g. Red border (b) corresponds to the band r.

checks», can be really useful when testing and validating different code implementations in complex pipelines and models. Moreover, we highly encourage software developers to perform these types of tests whenever possible in order to periodically check for a healthy code implementation.

C.6.2. TimeModRNN and CatTimeAttn Models

In this section, additional model settings are implemented and studied. First, we study the effect of using the temporal modulation (TimeFiLM) along with RNNs. Recalling the proposed TimeModAttn described in section §3.4, a new architecture setting is used (TimeModRNN) as follows: the number of N_L HMSelfAttn's layers of the encoder are replaced with a number of N_L RNN's layers (e.g., GRU, LSTM).

Second, we compare the use of the temporal modulation (TimeFiLM) w.r.t. the use of a Temporal Encoding (TE) to induce the temporal information in the TimeModAttn model. A new architecture setting is used (CatTimeAttn), where two changes are implemented: 1) The temporal modulation is completely removed (or bypassed) from the encoder. 2) The encoder input vector $\boldsymbol{x}_{i,j}$, used in the TimeModAttn model, is re-defined in order to include the values of a juxtaposed TE: $\boldsymbol{x}_{i,j} \leftarrow \text{cat}[\boldsymbol{x}_{i,j}, \boldsymbol{t}_{i,j}]$, where the TE vector is defined as follows:

$$\boldsymbol{t}_{i,j} = f_{\text{TE}}(t_{i,j}) = \begin{bmatrix} \sin\left(\omega_1 t_{i,j} + \phi_1\right) \\ \cos\left(\omega_1 t_{i,j} + \phi_1\right) \\ \vdots \\ \sin\left(\omega_{K/2} t_{i,j} + \phi_{K/2}\right) \\ \cos\left(\omega_{K/2} t_{i,j} + \phi_{K/2}\right) \end{bmatrix}, \omega_k = \frac{2\pi}{T_k}, \tag{C.1}$$

where the TE vector has K dimensions with $K/2$ different periods $\{T_k\}_{k=1}^{K/2}$ and phases $\{\phi_k\}_{k=1}^{K/2}$. For the sake of a fair comparison, we initialize the TE by using the same setting's rule of periods and phases used for the Fourier decomposition shown in section §3.4.3.1 (TimeFiLM). Note that the dimensions of the TE (K) follows the proportion $K = 2M$, where M is the number of harmonics used in the Fourier decomposition.

Table C.5 shows the late-classification performance results obtained for both, the TimeModRNN model and the CatTimeAttn model. In addition, table C.6 shows the early-classification performance results.

C.6.2.1. TimeModRNN Model Discussion

From the reported experiments, we observe that the TimeModRNN models achieved higher performances that the RNN baselines for both, the late-classification and early-classification. This suggests that the use of the time modulation (TimeFiLM) results in a better representation of the irregular cadence for the encoder, enhancing the overall performance w.r.t. the use of the time difference information. As previously discussed in section §C.4, this phenomenon could be related with the use of a dual representation of the time information: time modulation in the encoder and time difference in the decoder.

In fact, we hypothesize that the reconstruction scenario for the RNN baselines is peculiar as the encoder directly includes the time difference information Δt in the input vector, which is the same information used also for the decoder as the input vector (or highly related in the serial encoder case). This redundant information used for both, the encoder and decoder, could facilitate the situation where the encoder is learning the mapping of pairs $\{\Delta t, x\}$,

Table C.5: Late-classification performances for the BRF baseline, RNN baselines, and attention-based models (TimeModAttn) using 100-day multi-band light-curves. Several extra settings are shown, e.g., TimeModRNN, CatTimeAttn. Both, the serial (S-{}) and parallel (P-{}) encoders are reported along with several pre-training and data-augmentation schemes (mean±std from 5-fold cross-validation).

Model	b-Precision	b-Recall	b-F_1score	b-AUCROC	b-AUCPR
BRF (fmode=all; training-set=[r])	.527±.030	.687±.052	.525±.039	.866±.020	.602±.051
Serial Models					
S-RNN+Δt (cell=GRU)	.545±.034	.706±.070	.556±.045	.879±.034	.610±.066
S-RNN+Δt (cell=LSTM)	.550±.031	.711±.070	.558±.040	.887±.033	.621±.070
S-TimeModRNN+Δt (cell=GRU)	.581±.033	.737±.061	.596±.045	.898±.026	.651±.058
S-TimeModRNN+Δt (cell=LSTM)	.581±.024	.749±.047	.597±.037	.900±.024	.643±.057
S-TimeCatAttn (TE=24; H=8)	.577±.021	.745±.046	.581±.035	.905±.023	.647±.064
S-TimeModAttn (M=12; H=8; ε_t=6/24)	.588±.023	.759±.040	.596±.033	.910±.020	.671±.056
Parallel Models					
P-RNN+Δt (cell=GRU)	.547±.030	.697±.070	.552±.041	.879±.031	.610±.055
P-RNN+Δt (cell=LSTM)	.541±.022	.704±.061	.540±.032	.876±.029	.606±.051
P-TimeModRNN+Δt (cell=GRU)	.578±.033	.734±.063	.589±.049	.894±.028	.650±.067
P-TimeModRNN+Δt (cell=LSTM)	.580±.026	.741±.058	.592±.042	.897±.025	.646±.056
P-TimeCatAttn (TE=24; H=4)	.567±.025	.738±.054	.575±.043	.898±.024	.657±.059
P-TimeModAttn (M=12; H=4; ε_t=6/24)	.580±.020	.753±.044	.594±.035	.911±.017	.689±.047

Table C.6: Early-classification performances for the RNN baselines and attention-based models (TimeModAttn). The moving threshold-day Curve Average (mtdCA) is used (‡). Several extra settings are shown, e.g., TimeModRNN, CatTimeAttn. Both, the serial (S-{}) and parallel (P-{}) encoders are reported along with several pre-training and data-augmentation schemes (mean±std from 5-fold cross-validation).

Model	b-Precision‡	b-Recall‡	b-F_1score‡	b-AUCROC‡	b-AUCPR‡
Serial Models					
S-RNN+Δt (cell=GRU)	.481±.030	.577±.044	.454±.031	.792±.024	.520±.039
S-RNN+Δt (cell=LSTM)	.480±.023	.590±.036	.457±.027	.804±.026	.527±.043
S-TimeModRNN+Δt (cell=GRU)	.515±.027	.595±.036	.492±.031	.813±.021	.551±.043
S-TimeModRNN+Δt (cell=LSTM)	.512±.023	.601±.025	.491±.026	.815±.023	.555±.046
S-TimeCatAttn (TE=24; H=8)	.513±.018	.618±.025	.484±.022	.834±.016	.562±.038
S-TimeModAttn (M=12; H=8; ε_t=6/24)	.522±.022	.630±.026	.495±.020	.841±.016	.580±.040
Parallel Models					
P-RNN+Δt (cell=GRU)	.485±.023	.580±.041	.462±.028	.795±.026	.524±.040
P-RNN+Δt (cell=LSTM)	.476±.018	.586±.036	.451±.024	.795±.025	.516±.034
P-TimeModRNN+Δt (cell=GRU)	.509±.026	.597±.035	.490±.029	.810±.021	.550±.047
P-TimeModRNN+Δt (cell=LSTM)	.513±.021	.601±.033	.490±.025	.814±.022	.550±.043
P-TimeCatAttn (TE=24; H=4)	.501±.019	.604±.034	.482±.026	.822±.019	.561±.038
P-TimeModAttn (M=12; H=4; ε_t=6/24)	.514±.018	.621±.027	.499±.019	.841±.015	.587±.029

where x is the observation level, to a representation that can be directly decoded by using

the same values Δt handled by the decoder. therefore, the model could use the values Δt as pseudo sequential index information instead of the desired meaningful irregular cadence information to compute the relevant time dependencies, learning a function $f(\Delta t, x) = z$ with the encoder and, then, learning the inverse function f^{-1} with the decoder to recover the original observation level x by using both values: Δt and z.

Given the described phenomenon and recalling that the most relevant term in the pre-processing optimization is the reconstruction term, our conjecture is that the representation vector generated with the encoder could include automatic features only relevant for the reconstruction task, entangling the representation for the classification task. This could finally lead to a sub-optimal performance in the classification for architectures based on RNN autoencoder models that use Δt as the irregular cadence information.

Finally, note that the performance results, observed for the TimeModRNN model, are still lower than the results obtained for the TimeModAttn model. As discussed in section §4.1.5.2, this effect could be explained because, in contrast to the attention-based models, the RNN models lack the capability to directly access any observation of the light-curve (except the current observation). This seems to be especially important when processing SN light-curves, where the very first observations could be highly informative for this type of transient event. Note that the RNN baselines may have difficulties to properly capture the information from early observations as the maximum path length could be too large and composed of uninformative observations from the SN-extinction region, especially in long-duration SN light-curves.

C.6.2.2. CatTimeAttn Model Discussion

From the reported experiments, we observe that the CatTimeAttn models achieved lower performances that the TimeModAttn model for both, the late-classification and early-classification. Note that, in the late-classification scenario, the CatTimeAttn can be highly competitive w.r.t. the TimeModAttn model. In contrast, in the early-classification scenario, the performance difference is larger, suggesting that the TimeModAttn model is a suitable alternative for the early-classification scenario.

We hypothesize that the simple concatenation of the input vector and the TE vector could be detrimental because the time information is over-represented in the new resulting input vector of the encoder. If almost all the components of the input vector are related with time information (and just few of them with the brightness and/or bands information) then the model should use extra capacity just to discriminate which components of the input are associated with the, arguably, most important information: the observation-flux (brightness)[12]. Additionally, a risk of overfitting may exist related with this over-representation of the time information. In fact, we hypothesize that this detrimental phenomenon could be worsened if a higher number of TE dimensions is used.

Alternatively, the simple concatenation for the input vector may produce a non-normalized vector, which could be detrimental for the model optimization using GD-based algorithms.

[12]Note that the results in section §C.6.1 suggest this conjecture.

For example, the scale range of the values related with the observation-flux components, in the new input vector, might be lower (attenuated) than the values related with the time information (TE vector), biasing the optimization process towards the time information. Note that the two aforementioned effects may not apply for the proposed time modulation as the observation-flux is well-represented when it is modulated using the time information. In any case, more experiments should be performed following this line of research.

It is worth mentioning that the proposed time modulation (TimeFiLM) allows us to access an extra layer of interpretability for the SN event, exploring the time regions with higher or lower variability (section §4.1.5.3). In any case, the interpretability of the TE approach could be further explored with other type of experiments. For example, the Time2Vec (Kazemi et al., 2019) TE encoding could be used when processing light-curves of periodic stars, searching for correlations between the final learned period values and the intrinsic periods observed in the light-curve dataset.

C.6.3. Number of MHSelfAttn's Layers in the Encoder

In this section we study the effect of adding more MHSelfAttn's layers in the encoder of the TimeModAttn model. The main motivation for this analysis is to check for possible variations in the results of the proposed interpretability experiments.

For this, we increase the number of MHSelfAttn's layers in the encoder from $N_L = 1$ to $N_L = 2$. Note that, in the case of a single layer ($N_L = 1$), the query vector from the last MHSelfAttn's layer (the single MHSelfAttn's layer in the encoder) can exclusively access to information from the last observation of the light-curves. In contrast, in the case of $N_L = 2$, the query vector from the last MHSelfAttn's layer can access to information obtained from the complete light-curves because the first MHSelfAttn's layer can sequentially pre-process the information by using all the observations of the light-curves. Hence, the following questions arise: ¿Can the information handled by the last MHSelfAttn's layer query vector influence the general behavior of the attention mechanism? In particular ¿Is there a change in the overall and previous reported attention scores and temporal modulation behavior w.r.t. the case of a single layer?

When using $N_L = 2$, Fig. C.15 shows the statistics for the high attention scores extracted from the last layer of the encoder. Fig. C.16 shows the temporal modulation variability. In general, the results obtained from these interpretability experiments remain unchanged w.r.t. the results obtained when using $N_L = 1$, suggesting that the use of more MHSelfAttn's layers does not heavily affect the general behavior of the attention scores or the temporal modulation. Arguably, when using $N_L = 2$, a moderately less sparse high attention score distribution can be observed, where the distribution is slightly even more collapsed to the early observation-days.

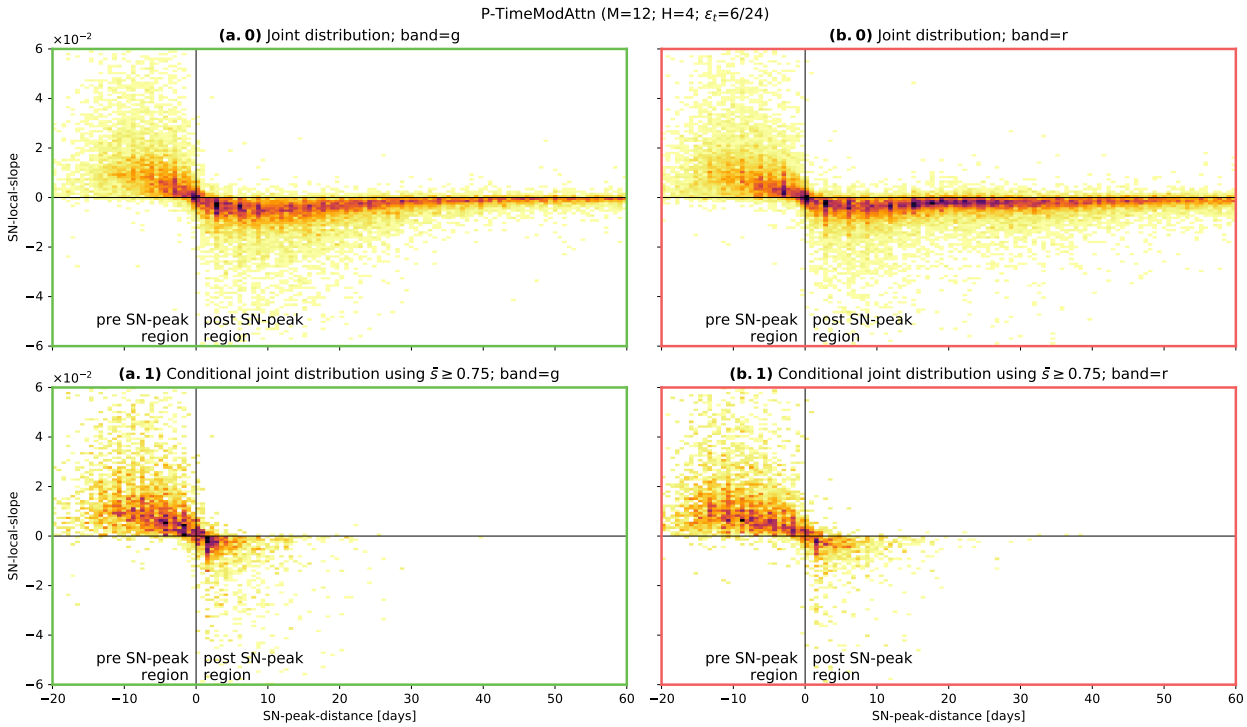

Figure C.15: Attention-based statistics for the P-TimeModAttn model after pre-training using $N_L = 2$. In the top row (plots (a.0) and (b.0) corresponding to the g and r band, respectively), the joint distribution is shown for both local SN-features, the SN-local-slope and the SN-peak-distance. In the bottom row (plots (a.1) and (b.1)), the conditional joint distribution is shown using a threshold for normalized attention scores of $\bar{s}_{\text{th}} = .75$. This allows highlighting the regions with higher attention scores. Visual guides, for zero SN-local-slope and zero SN-peak-distance, are shown as black lines. Green borders (plots (a.0) and (a.1)) correspond to the band g. Red borders (plots (b.0) and (b.1)) correspond to the band r.

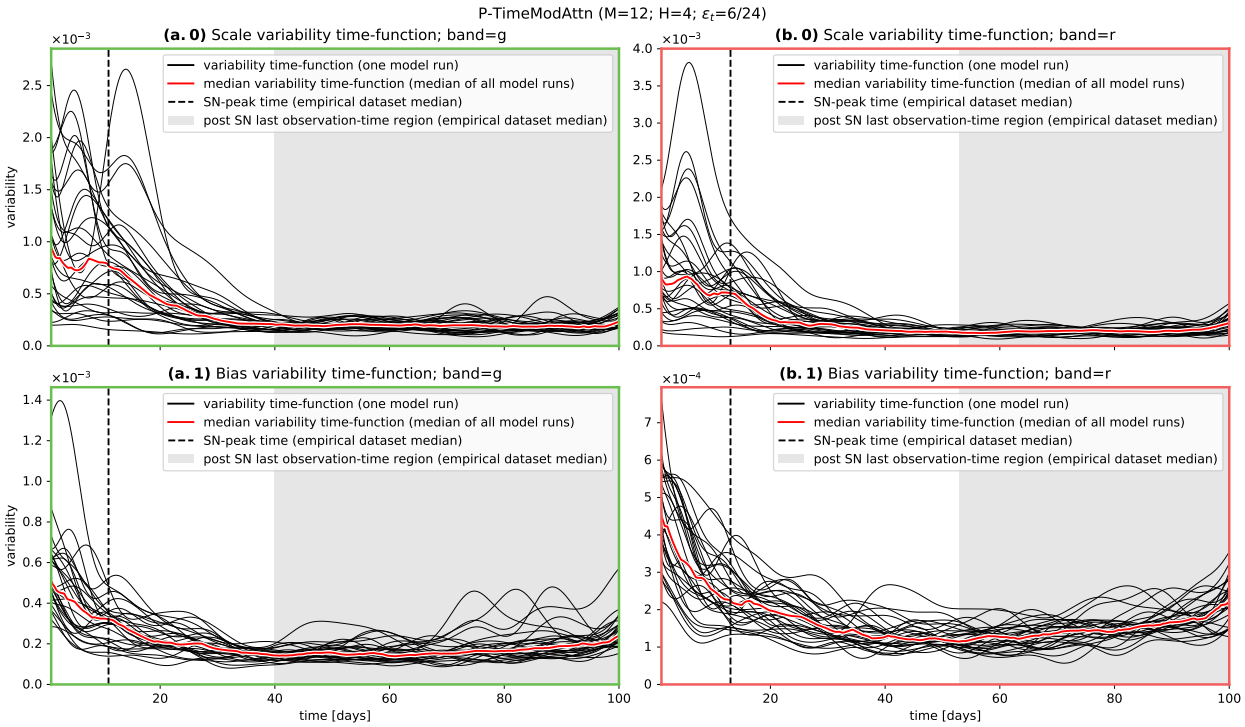

Figure C.16: Variability functions for the scale time-function (plots (a.0) and (b.0)) and bias time-function (plots (a.1) and (b.1)) computed from the temporal modulations learned by the P-TimeModAttn model after the pre-training process for a time range $[0, 100]$ [days] (using $N_L = 2$). A high variability can be observed before the SN-peak time. Each model iteration (total of N_{runs}) is represented with a black curve. The red curves are computed as the median curve using all the N_{runs} model iterations. The SN-peak time is computed as the median empirical maximum brightness time from the original dataset $\mathcal{D}$. The gray region starts at the empirical median SN last observation-time computed from the dataset $\mathcal{D}$. Green borders (plots (a.0) and (a.1)) correspond to the band g. Red borders (plots (b.0) and (b.1)) correspond to the band r.

Annexed D

Alternative Abstracts

D.1. Extended 250 Words Abstract

The following abstract was used in the submitted paper[1]:

In astronomical surveys, such as the Zwicky Transient Facility (ZTF), supernovae (SNe) are relatively uncommon objects compared to other classes of variable events. Along with this scarcity, the processing of multi-band light-curves is a challenging task due to the highly irregular cadence, long time gaps, missing-values, low number of observations, etc. These issues are particularly detrimental for the analysis of transient events with SN-like light-curves. In this work, we offer three main contributions. First, based on temporal modulation and attention mechanisms, we propose a Deep Attention model called TimeModAttn to classify multi-band light-curves of different SN types, avoiding photometric or hand-crafted feature computations, missing-values assumptions, and explicit imputation and interpolation methods. Second, we propose a model for the synthetic generation of SN multi-band light-curves based on the Supernova Parametric Model (SPM). This allows us to increase the number of samples and the diversity of the cadence. The TimeModAttn model is first pre-trained using synthetic light-curves in a semi-supervised learning scheme. Then, a fine-tuning process is performed for domain adaptation. The proposed TimeModAttn model outperformed a Random Forest classifier, increasing the balanced-F_1score from $\approx$ .525 to $\approx$ .596. The TimeModAttn model also outperformed other Deep Learning models, based on Recurrent Neural Networks (RNNs), in two scenarios: late-classification and early-classification. Finally, we conducted interpretability experiments. High attention scores were obtained for observations earlier than and close to the SN brightness peaks. This phenomenon was also correlated with an early highly variability of the learned temporal modulation.

[1] `https://arxiv.org/abs/2201.08482v2`.

D.2. Extended 300 Words Abstract

In astronomical surveys, such as the Zwicky Transient Facility (ZTF), supernovae (SNe) are relatively uncommon objects compared to other classes of variable events. Along with this scarcity, the processing of multi-band light-curves is a challenging task due to the highly irregular cadence, long time gaps, missing-values, low number of observations, etc. These issues are particularly detrimental for the analysis of transient events with SN-like light-curves. In this work, we offer three main contributions. 1) Based on temporal modulation and attention mechanisms, we propose a Deep Attention model called TimeModAttn to classify multi-band light-curves of different SN types, avoiding photometric or hand-crafted feature computations, missing-values assumptions, and explicit imputation and interpolation methods. 2) We propose a model for the synthetic generation of SN multi-band light-curves based on the Supernova Parametric Model (SPM). This allows us to increase the number of samples and the diversity of the cadence. The TimeModAttn model is first pre-trained using synthetic light-curves in a semi-supervised learning scheme. Then, a fine-tuning process is performed for domain adaptation. The proposed TimeModAttn model outperformed other Deep Learning models, based on Recurrent Neural Networks (RNNs), in two scenarios: late-classification and early-classification. Additionally, the TimeModAttn model also outperformed a Random Forest (RF) classifier (trained with real data), increasing the balanced-F_1score from $\approx .525$ to $\approx .596$. When training with synthetic data, the TimeModAttn model still achieves a highly competitive performance w.r.t. to the RF classifier while still maintaining advantages, such as the unnecessary costly computation of features from the new generated synthetic SN light-curves. 3) We conducted interpretability experiments. High attention scores were obtained for observations earlier than and close to the SN brightness peaks. This phenomenon was also correlated with an early highly variability of the learned temporal modulation.