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

IMAGE SEQUENCE SIMULATION AND DEEP LEARNING FOR ASTRONOMICAL OBJECT CLASSIFICATION

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

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

RODRIGO ANTONIO CARRASCO DAVIS

PROFESOR GUÍA:
PABLO ESTÉVEZ VALENCIA

PROFESOR CO-GUÍA:
GUILLERMO CABRERA VIVES
PROFESOR CO-GUÍA 2:
FRANCISCO FÖRSTER BURÓN

MIEMBROS DE LA COMISIÓN:
SUSANA EYHERAMENDY DUERR
CLAUDIO PEREZ FLORES

SANTIAGO DE CHILE
2019

RESUMEN DE LA TESIS PARA OPTAR AL
GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA
Y AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: RODRIGO ANTONIO CARRASCO DAVIS
FECHA: 2019
PROF. GUÍA: PABLO ESTÉVEZ VALENCIA

IMAGE SEQUENCE SIMULATION AND DEEP LEARNING FOR ASTRONOMICAL OBJECT CLASSIFICATION

In this thesis, a new sequential classification model for astronomical objects based on a recurrent convolutional neural network (RCNN) which uses sequences of images as inputs is proposed. This approach avoids the computation of light curves or difference images. This is the first time that sequences of images are used directly for the classification of variable objects in astronomy. Another contribution of this work is the image simulation process. Synthetic image sequences that take into account the instrumental and observing conditions were simulated, obtaining a realistic, unevenly sampled, and variable noise set of movies for each astronomical object. The simulated dataset is used to train the RCNN classifier. This approach allows to generate datasets to train and test the RCNN model for different astronomical surveys and telescopes. Moreover, using a simulated dataset is faster and more adaptable to different surveys and classification tasks. The aim is to build a simulated dataset whose distribution is close enough to the real dataset, so that a fine tuning could match the distributions and solve the domain adaptation problem between the simulated dataset and real dataset. To test the RCNN classifier trained with the synthetic dataset, real-world data from the High cadence Transient Survey (HiTS) was used, obtaining an average recall of 85% among 5 classes, improved to 94% after performing fine tuning with 1000 iterations using 10 real samples per class. The results of the proposed RCNN model were compared with those of a light curve random forest classifier. The proposed RCNN with fine tuning has a similar performance on the HiTS dataset compared to the light curve random forest classifier, trained on an augmented training set with 100 copies of 10 real samples per class. The RCNN approach presents several advantages in an alert stream classification scenario, such as a reduction of the data pre-processing, faster online evaluation and easier performance improvement using a few real data samples. The results obtained encourage us to use of the proposed method for *astronomical alert brokers* that will process alert streams generated by new telescopes such as the Large Synoptic Survey Telescope. Ideas for a multi-band classifier and a better image simulator are proposed based on the difficulties faced in this work.

Resumen

En esta tesis, se propone un nuevo modelo de clasificación secuencial para objetos astronómicos basado en el modelo de red neuronal convolucional recurrente (RCNN) que utiliza secuencias de imágenes como entradas. Este enfoque evita el cálculo de curvas de luz o imágenes de diferencia. Esta es la primera vez que se usan secuencias de imágenes directamente para la clasificación de objetos variables en astronomía. Otra contribución de este trabajo es el proceso de simulación de imagen. Se simularon secuencias de imágenes sintéticas que toman en cuenta las condiciones instrumentales y de observación, obteniendo una serie de películas de ruido variable, realistas, muestreadas de manera irregular para cada objeto astronómico. El conjunto de datos simulado se utiliza para entrenar el clasificador RCNN. Este enfoque permite generar conjuntos de datos para entrenar y probar el modelo RCNN para diferentes estudios astronómicos y telescopios. Además, el uso de un conjunto de datos simulado es más rápido y más adaptable a diferentes *surveys* y tareas de clasificación. El objetivo es crear un conjunto de datos simulado cuya distribución sea lo suficientemente cercana al conjunto de datos real, de modo que un ajuste fino sobre el modelo propuesto pueda hacer coincidir las distribuciones y resolver el problema de adaptación del dominio entre el conjunto de datos simulado y el conjunto de datos real. Para probar el clasificador RCNN entrenado con el conjunto de datos sintéticos, se utilizaron datos reales de *High Cadence Transient Survey (HiTS)*, obteniendo un *recall* promedio del 85 % en 5 clases, mejorado a 94 % después de realizar un ajuste fino de 1000 iteraciones con 10 muestras reales por clase. Los resultados del modelo RCNN propuesto se compararon con los de un clasificador de bosque aleatorio o *random forest* de curvas de luz. El RCNN propuesto con ajuste fino tiene un rendimiento similar en el conjunto de datos HiTS en comparación con el clasificador de bosque aleatorio de curva de luz, entrenado en un conjunto de entrenamiento aumentado con 100 copias de 10 muestras reales por clase. El enfoque RCNN presenta varias ventajas en un escenario de clasificación de *streaming* de alertas astronómicas, como una reducción del preprocesamiento de datos, una evaluación más rápida y una mejora más sencilla del rendimiento utilizando unas pocas muestras de datos reales. Los resultados obtenidos fomentan el uso del método propuesto para los sistemas *astronomical alert brokers* que procesarán *streamings* de alertas generados por nuevos telescopios, como el *Large Synoptic Survey Telescope (LSST)*. Se proponen ideas para un clasificador multibanda y un mejor simulador de imágenes en función de las dificultades encontradas en este trabajo.

A mis padres. Muchas gracias.

Agradecimientos

Quiero agradecer a mis padres por todo el apoyo incondicional que me han dado, y en particular por enseñarme que uno tiene que hacer lo que le gusta y además disfrutar de aquello. Ambos acostumbran a decir que no me pueden ayudar mucho, pero la verdad es que sin su apoyo, en todo orden de cosas, yo no estaría escribiendo los agradecimientos para una tesis de magister, que por lo demás debería tener sus nombres en la portada. También quiero agradecer a mis hermanos, a mi abuela y al resto de mi familia por la fe que han tenido en mi.

Agradezco a la Coni, mi polola, quien me acompañó en el último tramo de mi título y le ha tocado ver de cerca lo que me costó terminar este trabajo. Gracias por compartir tu alegría, tu cariño y tu sabiduría de la vida conmigo. También agradezco a mis amigos, que por suerte son muchos, Jorge-kun, Charlie, Cesar, Nico, Lucho, Jose por la fe que me tienen, las risas, conversaciones filosóficas e incontables horas de estudio. A mis amigos del colegio, al Pancho, la Pame, Yves, Omar, Gonzalo y Felipe a quienes conozco por más de diez años, me conocen bien y siempre me desean lo mejor, muchas gracias.

Obviamente este trabajo no hubiera sido posible sin la ayuda de mi profesor guía Pablo Estévez, gracias por confiar en mi, por su guía y consejo. Igualmente agradezco a Francisco Förster y a Guillermo Cabrera, por compartir conmigo sus conocimientos y su buena onda. Gracias a la gente del laboratorio, a Ignacio Reyes, Jorge Vergara, Pablo Huijse, Rosario Molina, Esteban Reyes, el Nico VAE, Nico Sueño y German. No pude tener más suerte con las personas con las que trabajo, siempre dispuestos a ayudar.

Ciertamente que hay personas que no he mencionado, si alguna vez compartimos risas o palabras de aliento, entonces te agradezco a ti también. Mención honrosa a la Pelusa que le gusta sentarse al lado mio cuando me quedo estudiando hasta tarde.

Este trabajo ha sido parcialmente financiado por **CONICYT-PCHA-22180964** Beca de Magister Nacional 2018 y **FONDECYT-1171678**

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Astronomy is faced with the challenge of increasingly large streams of data produced by large survey telescopes. New telescopes, such as the Large Synoptic Survey Telescope [1] and the Zwicky Transient Facility [2] are designed to study variables and transients on wide areas of the sky. Variable stars, such as pulsating (e.g., RR Lyrae, Cepheids) or eclipsing stars; or transients, such as supernovae, are expected to be produced in large numbers. These objects have characteristic timescales from hours to months, and can be detected and characterized by repeatedly observing the same region of the sky. Obtaining these repeated images with large cameras will generate a very large volume of data. For example, it is estimated that the LSST will generate 30 TB of data per night to produce a complete image of the southern sky every 3 days.

Some research areas in astronomy require the classification of a large number of objects: e.g, supernovae are used to estimate cosmological distances for studies about the expansion of the universe [3, 4] and variable stars such as RR Lyrae or Cepheids are needed to map the structure of the Milky Way as they serve as *cosmic distance ladders* [5, 6]. In order to classify different astronomical objects in these large data streams it is necessary to apply fast and accurate classification methods capable of managing large amounts of data in real-time. This problem will be addressed by systems called *astronomical alert brokers* e.g., ALeRCE, LASAIR, ANTARES [7], which are capable of receiving, processing, classifying and reporting relevant information about the alert streams generated by large survey telescopes in real time.

Traditional methods to classify variable astronomical objects are based on pre-processing a sequence of images (calibration) followed by feature extraction (measurement). One way to extract features from a sequence of images is doing photometry [8], which is the calculation of the total amount of light arriving from the source to the camera as a function of time, generating a time series called a *light curve*. In principle, a point–like source's light curve should contain all the relevant information to classify the source, but when the detection is spurious the information contained in the images becomes more relevant for the classification. Additionally, extragalactic sources such as supernovae tend to be near extended sources, i.e. galaxies, whereas galactic variable stars tend to be relatively isolated, information which is also contained in the pixels.

Obtaining the light curve reliably requires performing difference imaging first for certain sources (e.g., when the object occurs in a bright galaxy), which is the process of aligning, convolving and differencing pairs of images to show only those pixels which have changed from frame to frame [9]. Computing the difference image presents some problems, most of the time it is necessary to reduce the quality of one of the two images to subtract them correctly and it is also very sensitive to alignment errors between the frames.

Once the full light curves are computed, additional features can be extracted by manual design or automatic learning from the data. In the case of manual feature extraction, the scientist must design attributes that are expressive enough to contain relevant information for the classification, which usually requires a lot of effort and time [10, 11, 12, 13, 14, 15]. Learning features directly from data is one way to avoid manual design and can be very useful to find informative attributes for classification [16, 17, 18, 19, 20, 21, 22, 23]. However, even though representative features were obtained, if the data from the pre-processing step is not informative enough or contains errors from the procedure it would be difficult to obtain a good classification. Furthermore, in the case of an alert stream, for a new incoming sample of a light curve, it is desirable to update the feature value using the last point instead of using the entire light curve to avoid unnecessary computation and data retrieving [24].

Deep learning techniques are examples of data-driven solutions extracting features automatically that have proven to be successful in classification problems. Convolutional neural networks [25] have been applied to spatially correlated data such as images [26, 27] and temporal correlated data such as audio [28, 29] among others. Recurrent neural networks, e.g., those containing Long Short Term Memory units [30, 31, 32], have been applied to many natural language processing problems [33] like translation [34] and speech recognition [35].

Recently, deep learning has been successfully applied to astronomical problems using convolutional neural networks, for example, for real/bogus separation [16, 17], photometry computation [36], calculation of an image comparable to the difference image [18], gravitational wave detection [19] and exoplanet detection [20]. Recurrent neural networks have been used for light curve classification [21, 22, 23, 24].

Recurrent convolutional neural networks are a special type of neural network where convolutional layers are combined with recurrent layers. Usually, a first stage of convolutional layers extract features from the raw data and generate high-level representations in deeper layers, then a second stage of recurrent layers uses the features yielded by the convolutional layers to learn time dependencies. Examples of applications are action recognition in videos [37, 38, 39] and speech recognition [40].

The first contribution of this work is the proposal of a model to classify variable astronomical objects based on a recurrent convolutional neural network (RCNN), which uses sequences of images directly as inputs. In this way, it is possible to estimate the class probability of a specific astronomical object by using the alert stream directly. With this approach, there is no need for recomputing features when a new observation arrives, only feed the recurrent model with the incoming image. The information about previous images of the source is encoded in the network state. The computational cost of the proposed model scales linearly with the number of images within a sequence, encouraging us to use it when facing large data stream. Furthermore, it is ensure that all the information available on the images is fed to

the classifier without inducing errors by computing the light curve (e.g., in spurious sources) or the difference images (e.g., in badly convolved images). The proposed model consists of convolutional layers aimed at learning spatial correlations automatically from the images at each epoch, followed by a recurrent layer aimed at learning time dependencies between frames of an image sequence. To the best of our knowledge, this is the first time that image sequences are used directly to classify astronomical objects.

The second contribution of this work is the image simulation process that generates synthetic image sequences that take into account the instrumental and observing conditions, obtaining a realistic, unevenly sampled, and variable noise set of movies for each astronomical object. The simulated dataset is used to train the RCNN classifier. This procedure is faster and more adaptable to different surveys and classification tasks, as compared to collecting real labeled data which is time-consuming and fixed to a specific survey. Furthermore, by randomizing simulations correctly it could be possible to generate a virtually infinite number of labeled samples, avoiding the problem of manually labeling a large number of real objects. Simulating a dataset allows, for example, creating data samples for telescopes that are still under construction such as the LSST. Simulation parameters can be tuned according to specific classification tasks and scientific objectives.

Since the simulated data distribution may differ from those of real image sequences, a transfer learning technique called fine tuning [41, 42] is used to adapt the RCNN model trained over simulated images to solve the classification task on real images. By just using a few real labeled image sequences, the RCNN model performance improved substantially on the real dataset.

The structure of this thesis is the following: in Chapter 2, important concepts for developing this thesis are visited, astronomical concepts and machine learning among others. In Chapter 3 the methodology used in this work is presented. In particular, the process of simulating synthetic images is described in Section 3.1 and the proposed RCNN image classifier model is explained in Section 3.2, along with the random forest light curve classifier used for comparison purposes. In Chapter 4, the results are presented in Section 4.1 and an analysis of the results is presented in Section 4.2 along with classification examples in Section 4.3. In Section 4.4, assumptions, difficulties and proposed improvements are discussed. In Chapter 5, important conclusions, implications and future steps are commented.

1.1 Hypothesis

- It is possible to use the sequence of astronomical images directly as input to the RCNN classifier to discriminate between astronomical objects. Using raw images produce equal or better accuracy than using only the light curve since the image provides additional information.
- Training a classifier with realistic simulated astronomical objects produce good models in terms of accuracy when is applied to real data, which can be fine tuned later on.

1.2 General Objectives

In order to avoid the aforementioned pre-processing, the general goal, is to develop a method of classification of astronomical objects based on convolutional neural networks and recurrent networks, which uses the sequence of images of an object directly as input to the classifier.

1.3 Specific Objectives

- Design a simulator of astronomical images of transient objects considering observation conditions and telescope parameters. The simulator must be configurable for different surveys.
- Design a neural network based classifier, with convolution layers and recurrence layers, using the sequence of images simulated as input.
- Compare the proposed model with a model that uses light curves and Random Forest classifier
- Train the image sequence classifier using synthetic images and then apply it to real images.
- Fine tune the model obtained with a few real samples.

Chapter 2

Background

In this chapter, the important concepts for the development of this thesis are explained, including related work and examples. In particular, basic concepts of astronomy are described, the process of obtaining astronomical images, also astronomical objects of study such as supernovae, variable stars, among others. Then, concepts associated with deep learning, neural networks, convolutional networks, training methods, are reviewed.

2.1 Astronomy

Astronomy is the study of celestial objects and the composition of the universe on a small and large scale. Astronomers apply mathematical, physical and chemical knowledge to explain the existence and evolution of various objects in the universe. There are different observational methods to study objects. Depending on the observational method, different aspects of an astronomical object can be measured. Most telescopes focus on measuring electromagnetic waves emitted by astronomical objects, in different wavelength ranges such as ultraviolet, optical or infrared. Other observational methods are gravitational wave measurement [43] or neutrino detection [44, 45]. This thesis is focused on optical astronomy (visible and near infrared range), but many of these concepts and methods may apply to other kinds of signals.

2.1.1 Astronomical Surveys and Big Data

A survey corresponds to the execution of an observation plan using a particular instrument and the collection of the resulting images. In summary, an observation plan defines what regions of the sky will be captured, light frequencies and times to observe. In the case of a survey, the observation plan is not focused on observing a particular object, but instead takes data of regions of the sky and then analyzes it depending on the research task. For example, in the HiTS Survey [9] they observed the same region of the sky 5 times per night every two or three nights in order to find supernovae at the very beginning of the explosion and perform fast follow up (take the spectra of the source) and find evidence of the Shock

Breakout [46, 47]. They also used g band to take most of the images since it increases the detection efficiency. Examples of other surveys are MACHO [48], ASAS [49] and EROS [50], among others.

The Zwicky Transient Facility (ZTF [1]) and the Large Synoptic Survey Telescope (LSST [2]) are surveys covering large regions of the sky at a relatively high cadence. Figure 2.1 shows the field of view (area of the sky observed in a single exposure by the telescope) vs the light collecting area (or telescope diameter), where each circle in the plot is a different telescope. The product between both quantities is shown as the area of the circle for each telescope and is defined as **etendue** which corresponds to a measure of the volume of the universe that the telescope is able to observe in a single exposure. Note the large volume of the universe that the LSST is capable of capturing in comparison to the rest of the telescopes.

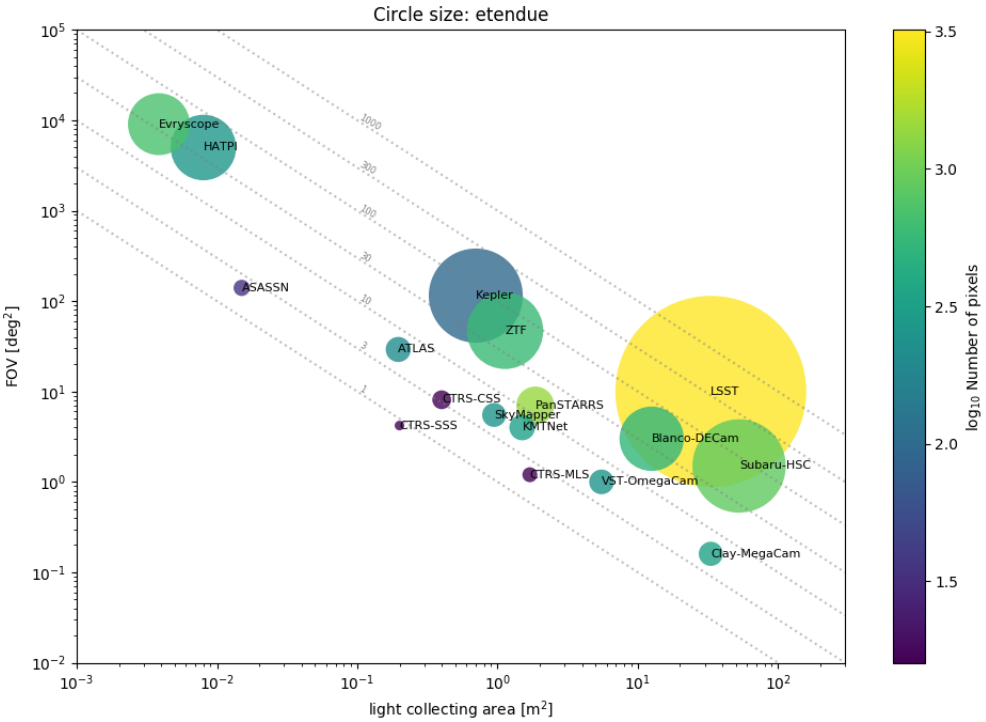


Figure 2.1: Etendue for different telescopes. The etendue is a measurement of the volume of the universe that a telescope is able to observe in a single exposure. It is the product between the field of view in [deg^2] and the light collecting area in [m^2], represented as the area the circle for each telescope.

One of the main objectives of the new telescopes is to find transients in large quantities. Transients are astronomical events of finite duration, such as supernovae or gamma rays burst. In particular, supernovae can be used as standard candles, useful for measuring long distances in the universe. Also, other astronomical objects exist that serve to measure distances, variable stars for example, but they are used to measure distances within the Milky Way and nearby galaxies [6, 5]. On the other hand, supernovae can be detected at much greater distances and they are used for cosmological studies such as the expansion of the universe [3, 4]. In order to measure a transient adequately to calculate its distance from the earth, it is necessary to carry out "follow up" observations in order to measure the spectra

of the transient. The decision to execute follow up on a transient must be fast and with a low rate of false positives since the transient may be of short duration and it is not desirable to allocate observation time to false transients, which could be caused by camera errors or image pre-processing errors.

Telescopes with large etendue will be able to observe a large number of transient candidates, called "alerts" which are triggered when there is a considerable variation in the brightness between 2 images (explained later in Section 2.1.6). For example, in the case of LSST, it will produce 10 million alerts per night at a rate of 10,000 alerts every 37 seconds approximately. To have an idea, each step of processing the data should last a maximum of $37/10{,}000 = 3.7$ milliseconds to avoid the accumulation of pending alerts to be processed in the queue (assuming single thread of processing). As mentioned, it is necessary to perform this processing quickly in order to trigger follow up observations with other telescope facilities on interesting alerts. A precursor work of this kind of processing is the HiTS survey [9], designed to find transients on time scales from hours to days and focused on early stages of the supernova explosion. This problem will be addressed by systems called *astronomical alert brokers*, such as ALeRCE (Automatic Learning for the Rapid Classification of Events), LASAIR, ANTARES (The Arizona-NOAO Temporal Analysis and Response to Events System) [7], which are capable of receiving, processing, classifying and reporting relevant information about the alert streams generated by large survey telescopes in real time.

2.1.2 Basic Concepts

In optical astronomy, the brightness in a specific frequency band of an astronomical source is characterized by the magnitude of the source, which corresponds to a logarithmic scale of the energy flux rate produced by the source. Two kinds of magnitudes are important for this work.

Apparent magnitude: Measure of the brightness of a star perceived on earth. The apparent magnitude can be defined as a function of the photons rate coming from the source registered by the detector:

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

$$= -2.5 \log_{10}(F_s) + zp \tag{2.2}$$

where m is the magnitude value, F_s is the flux measured from the source and F_{ref} is a reference flux. Notice that brighter objects observed from the earth have lower magnitudes. The F_{ref} term in equation 2.1 can be removed from the logarithm and define it as zp called **zero point**. The zero point enclose information such as absorption and color diffraction corrections produced by the atmosphere. The value of the zero point is usually chosen to match the apparent magnitude of a known star measured using a different instrument. More details about the flux measuring process are explained in Section 2.1.6.

Absolute magnitude: Correspond to the measured flux of a source as it was placed at a distance of 10 parsecs. From the inverse square law of light, the absolute magnitude M

can be computed using the apparent magnitude m and the luminosity distance D (Stars and their spectra [51]):

$$M = m + 5 - 5\log_{10}(D). \tag{2.3}$$

When the source is a nearby object, the luminosity distance D, which corresponds to the distance traveled by the light from the source, is a good approximation for the distance in space. When the source is distant enough to consider cosmological effects, the luminosity distance is related to the transverse comoving distance D_M [52] by $D = (1+z)D_M$ where z is called redshift, which is explained later.

The absolute magnitude is an important quantity since relates the distance to a source with the luminosity measured by the instrument. In some cases, astronomers can calculate the absolute magnitude of a source from other measurements, for example, using a relation of the period with the luminosity of RR Lyrae or Cepheids stars which are periodic stars. Then, using equation 2.3 astronomers can infer the distance to the source and use it as a standard candle.

Light Curve: Usually, many images of the same astronomical object are generated in a survey, these images are irregularly taken in time, having variable time gaps between each image that have a duration from a couple of seconds to months, these gaps are due mainly to observational constraints like seasonal and day/night availability of the sources. By computing the magnitude for each image, we can build a time series called "light curve", that corresponds to the brightness of the source as a function of time. The light curve is one of the ways to discriminate among different types of astronomical objects and it is used for classification tasks [10, 13, 53] and for testing physical models [47]. Further explanation of light curves and its computation is given in Section 2.1.6. For time units, astronomers usually use Modified Julian Date (MJD), which is the number of solar days since midnight of November 17, 1858, Universal Time.

Redshift: According to the work about the expansion of the universe, meritorious of a Nobel Prize in 2011 [4], the universe is expanding at an accelerated rate. The speed at which galaxies are moving apart from each other due to the expansion follows the Hubble's Law $v = H_0 D$, where v is called recessional velocity (velocity of objects moving away from earth), H_0 is called the Hubble's constant and D is the distance to the source. Then, further astronomical objects are moving faster compared to near ones. If the source has a very high recessional velocity (which means it is very far), then a relativistic effect called "redshift" must be considered for analysis. The redshift is a Doppler Effect for light. If a source that emits light is moving away from the earth at high speed, every wavelength of the light signal is elongated, the wavelength elongation can also be produced by a gravitational field. The term "red shift" is based on the human perception of light as colors, where longer wavelengths are redder than shorter ones. The opposite effect is "blueshift", which occurs when the source of light is moving towards the earth. The redshift z is related to the emitted wavelength λ_{em} and observed wavelength λ_{ob}, the speed of light c and with the recessional velocity (considering a Minkowsky Space) by

$$1 + z = \sqrt{\frac{1+\frac{v}{c}}{1-\frac{v}{c}}} = \frac{\lambda_{\text{ob}}}{\lambda_{\text{em}}} \tag{2.4}$$

Figure 2.2 shows an example of redshifted spectra of a Type IIP supernova. The attenuation of the observed spectra is because the source is further away due to cosmological expansion with higher redshift. By using Hubble H_0 constant and the recessional velocity v given by the redshift z, the distance to the source D can be computed and also the flux attenuation which is given by the inverse square law for light $F_{ob} = F_{em}/D^2$ with F_{ob} the flux observed and F_{em} the flux emitted.

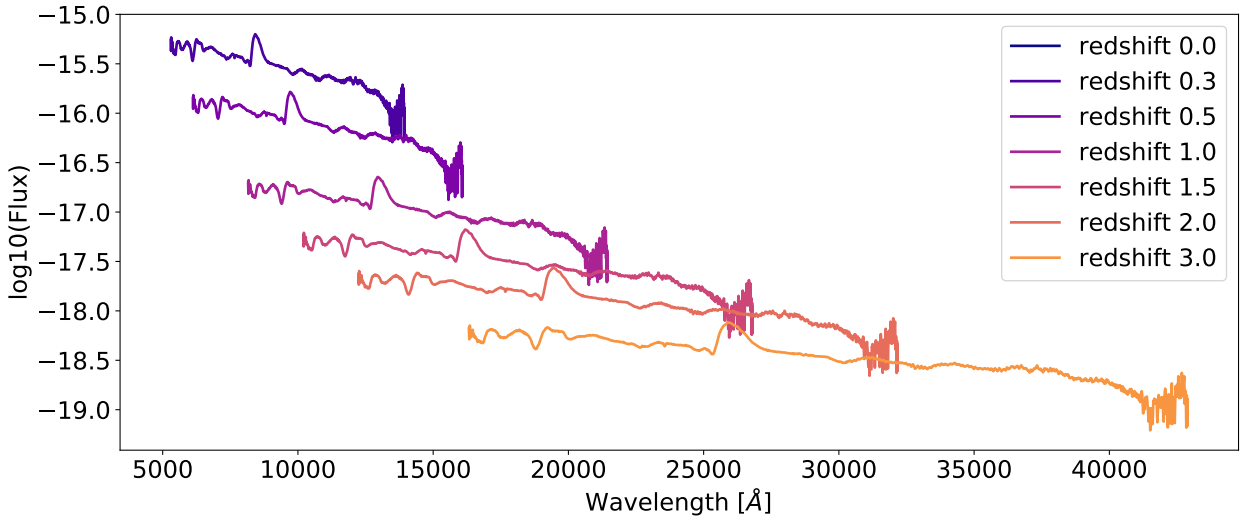


Figure 2.2: Supernovae spectra for different redshifts [54]. Each emitted wavelength λ_{em} is mapped to an observed wavelength λ_{ob} by equation 2.4. The distance is estimated from Hubble's law and the given redshift, then the inverse square law for light is applied to the flux.

2.1.3 Astronomical Objects

An astronomical object is a physical body or natural event that exists or occurs in the universe. Examples of astronomical objects are stars, planets, galaxies, black holes, supernovae, among others. In this section, the astronomical objects used in this work are explained, in terms of physical behavior and light curve characterization. The objects were grouped based on their light curve variability, these groups are Transients, Periodic and Non Variable objects.

Transients

Astronomical transients are events with limited duration. Some examples of transients are supernovae, tidal disruptions events, gamma-ray bursts, etc. Observing these objects helps to understand their underlying physical mechanism and some of them can be used to indirectly study other objects that have generated or influenced the transient event [55, 56]. The transients related to this work are Type Ia and Type II supernovae.

- **Type Ia Supernova**: This kind of supernovae may occur on binary systems where one of the stars consumes material from the companion, in a process called accretion. If the mass of the fed star reaches the Chandrasekhar mass of 1.38 solar masses, then a nuclear explosive reaction is triggered producing the supernova Type Ia explosion. [57, 58]. An example of a light curve of a Type Ia supernova is depicted in Figure 2.3.
- **Type II Supernova**: Occur in stars with a mass between 8 and 40 solar masses. During the life of a star, the energy produced by the fusion of elements such as hydrogen and helium prevents the star from collapsing under its own gravity, generating an equilibrium between those two forces. At the core of the stars, heavier elements like carbon, oxygen, silicon, among others, begin to be produced as a result of the fusion. When this process reaches iron and nickel, the synthesis of these elements no longer produce energy that can compensate the gravitational pressure, so the core is compacted and supported by the electron degeneracy pressure (two fermions cannot be in the same quantum state, this satisfies the Pauli exclusion principle). Once the inner core reaches the Chandrasekhar mass, the degeneracy pressure is not enough to prevent the core collapse produced by its own gravity. The core of the star is compressed even more in a matter of seconds, the formation of neutrons and neutrinos takes place and the inward collapse bounce outward producing a shock wave which is the supernova explosion. The energy released by the explosion ionize the hydrogen in the outer layers increasing the opacity of the surrounding material. Then, when hydrogen recombines due to cooling the opacity slowly decrease releasing the energy. This effect can be noticed in the light curve, where a plateau is formed right after the explosion due to recombination. An example of a light curve of a Type II supernova is depicted in Figure 2.3.

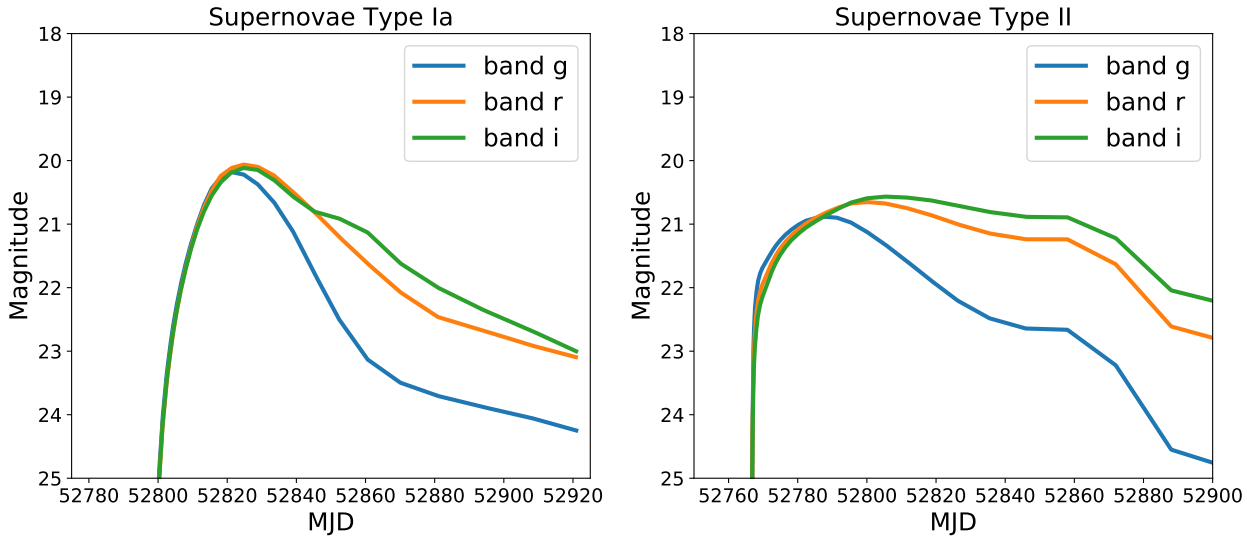


Figure 2.3: Type Ia and Type II supernovae simulated light curves. Type Ia supernova light curve is based on Hsiao models [58], type II supernova is based on Moriya models [59].

Periodic Objects

Some astronomical objects change their luminosity in time due to their physical mechanism, as in the case of transient objects. A subset of variable objects are periodic objects, where

a luminosity pattern is repeated through time. Examples of periodic objects are Cepheids, RR Lyrae, eclipsing binaries, delta scuti, long period variables, among others. In this thesis, the following periodic objects were used:

- **RR Lyrae**: Variable stars with periods between 0.2 to 1 day, and masses between 0.6 to 0.8 solar masses. RR Lyrae are very important since they only appear in very old stellar population (older than 10 Gigayears) [60]. By using the period-luminosity relation, RR Lyrae can be used as standard candles to calculate distances to these old stellar systems. Figure 2.4 shows a light curve example of an RR Lyra in different bands.
- **Cepheids**: Variable stars with periods between 1 to 100 days, and masses between 2 to 20 solar masses. Cepheids can be found further than RR Lyrae since the former are more luminous, and they are used to measure distances to nearby galaxies such as the Magellanic Cloud and the Andromeda Galaxy. These variable stars are found in stellar systems with recent star formation, for example, Cepheids are found within the young stellar population disk in the Milky Way [60]. Figure 2.4 shows a light curve example of a Cepheid in different bands.
- **Eclipsing Binaries**: In this case, the variation in luminosity of the source is not caused by an intrinsic variation of the brightness of a star. Eclipsing binaries occur in binary systems when two stars are orbiting around each other. When the orbit plane of the binary system is aligned with the line of sight, then, the luminosity of the system, observed from earth, decays when one star blocks the light of their companion. Binary systems are important since the orbit can be used to estimate the mass, along with other parameters such as the density of the stars, their luminosity and distance. Figure 2.4 shows a light curve example of an eclipsing binary in different bands.

Figure 2.4 shows examples of different variable stars. The upper plots are the time based light curves, i.e, the magnitude as a function of time. Because of the irregular sampling and noise, most of the time is hard to see a periodic structure in this representation. A better way to work with periodic signals is by transforming the time to phase space, which corresponds to mapping each time sample to a position in the phase of the periodic pattern. This transformation is also called "folding" the light curve and can be written mathematically as:

$$\theta_i = \lfloor t_i \rfloor_P \cdot \frac{1}{P} \tag{2.5}$$

where θ_i is the phase for sample i, t_i is the time in days, P is the period of the signal and $\lfloor \cdot \rfloor_P$ denotes the module P operator. Then $\theta_i \in [0, 1]$, and usually the phased light curves are plotted with two periods for visualization purposes. The lower plot in Figure 2.4 show the folded light curves for RR Lyra, Cepheid and eclipsing binary.

Non-variable Objects

This section refers to astronomical objects which their brightness does not change in a human time scale. For this thesis, two types of non-variable or constant objects are considered, non-variable stars and galaxies, which differs between them in their image shape.

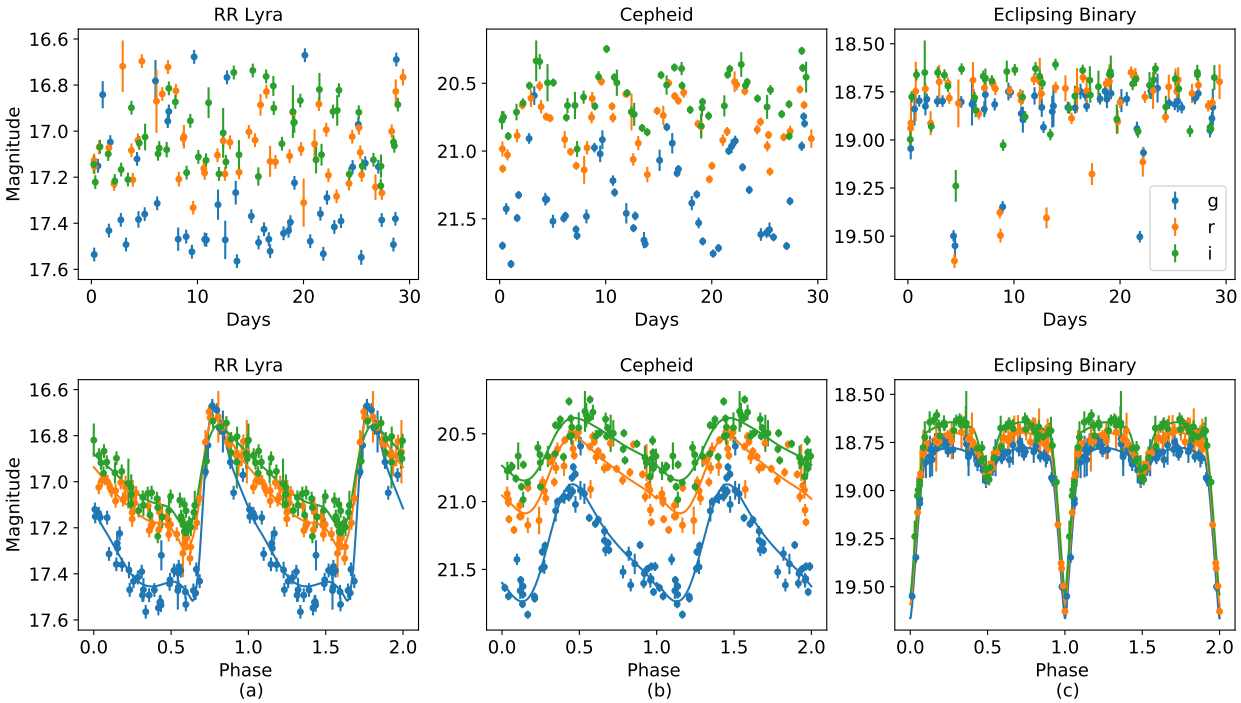


Figure 2.4: Examples of light curves for (a) RR Lyra, (b) Cepheid and (c) eclipsing binary. The upper plots show the magnitude vs time of simulated observation with irregular sampling and noise. The bottom plots show the magnitude vs phase after folding the original light curve.

- **Non-variable stars**: Luminous astronomical object. Different kinds of stars belong to this category, depending on their composition, localization, etc. For this work, we will characterize these objects as a point-like sources with constant brightness on each band.
- **Galaxies**: Correspond to structures compound by an enormous amount of stars, gas, dust and dark matter, gravitationally bound. Depending on the image resolution, galaxies are observed with an elongated structure, and they exist in different shapes such as elliptical, spirals and irregulars [61].

Further details about each object are given in Chapter 3, where some astronomical parameters must be considered in order to simulate their light curve and image. Furthermore, each type of object has properties in addition to their underlying physical mechanism, for example, their abundance in the universe, biases of finding certain objects depending on the detection system or the pointing of the survey.

2.1.4 Astronomical Image Formation

In optical telescopes, roughly speaking, the process of taking an astronomical image is the following:

- The telescope is pointed in the desired direction to observe the sky. In addition, a

bandpass filter for light is selected in order to register photons in a given range of frequencies. Examples of these bandpasses are shown in Figure 2.5, where the filters response of the Dark Energy Camera (DECam [62]) as a function of the wavelength is depicted. The amount of flux that is able to pass through the filter is proportional to the relative transmission.

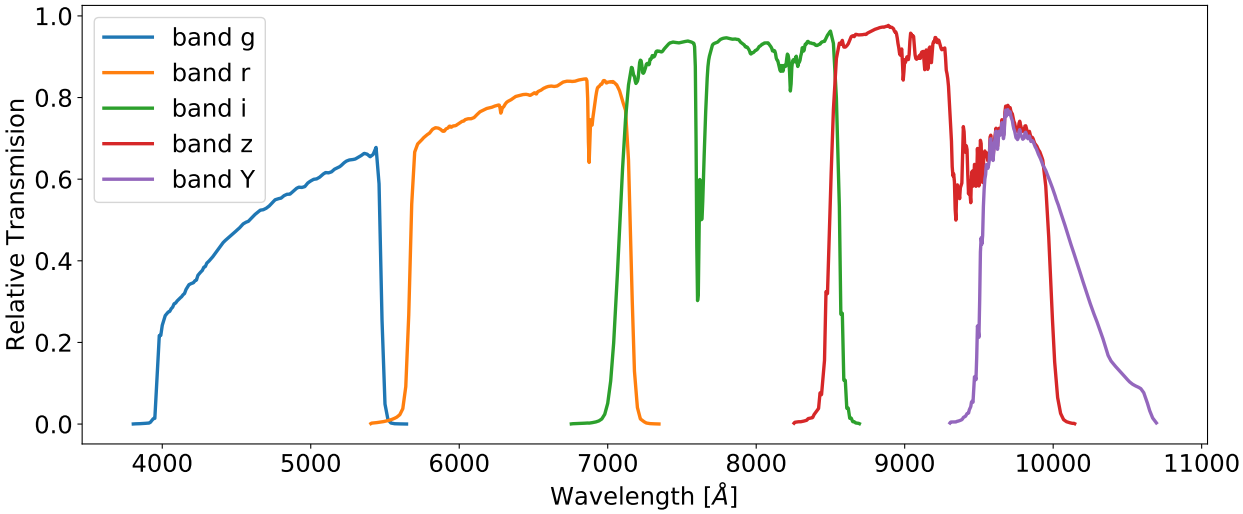


Figure 2.5: DECam Filter response

- The camera shutter is opened, the light coming from the source goes through the atmosphere, and the telescope structure (mirror, lens, filter, etc). The light gets distorted during this path. This distortion is characterized as a function called Point Spread Function (PSF) detailed later. The distorted version of the light arrives at the Charge-coupled device (CCD) camera and starts to fill each pixel with a charge package ([e-] units) proportional to the intensity of light in that region. The shutter is closed after a period of time called exposure time.
- Once the shutter is closed, the charges on each pixel are read and converted to a matrix of integer numbers in Analog to Digital Units (ADUs). Usually, the telescope moves to the next position (slew) while the CCD charges are been read.

An example of an astronomical image taken using DECam is shown in Figure 2.6.

Camera Specifications

Usually, charge-coupled device (CCD) cameras are used to take the images in optical telescopes. In each pixel of the image, a conversion is made of the number of photons detected in the pixel to electron charges using a semiconductor material. The camera pixels are exposed to light during the exposure time, which is controlled by the shutter. The relevant parameters to understand for this work are:

- **Camera gain**: The gain G of the camera corresponds to the conversion between the number of electron charges $C[\mathrm{e}^-]$ in a pixel and the final measurement of the pixel in units of "counts" or ADU (Analog-to-digital units) that follows $\mathrm{ADUs} = C \cdot G^{-1}$, so

the gain has units of [e^-/ADU]. The images of the CCD cameras are delivered for pre-processing in units of counts or ADUs.

- **Pixel Saturation**: Each pixel of the camera has a maximum capacity of charges that can store, the maximum number of charges that a pixel can contain is called the saturation value of the pixel. Once the level of saturation is reached, the pixel "overflows" of charges and begins to fill the surrounding pixels with the excess of electron charges, this phenomenon is called blooming effect [63]. The saturation value is in units of [e^-].
- **Pixel scale**: The region of the sky that is being observed for each pixel. This parameter is not only related to the CCD, but also to the entire optical system prior to the arrival of the photon to the CCD and it could vary within the same CCD. It is in units of [arcsec / pixel].
- **Dark current**: Electrons charges added to the pixel by a thermal effect. It is generated at every moment even after reading and resetting pixels before exposure. It could be avoided by cooling the CCD.
- **Read noise**: Once the exposure is done and the shutter is closed, the electron charges are read and the reading process has an associated noise, modeled as a Gaussian noise characterized by its standard deviation in units of charges [e^-].
- **Filter or Band**: Usually, optical telescopes observe light around a wavelength called a band or filter. For example, in the case of The Dark Energy Camera (DECam [64]) used in HiTS, has filters called g, r, i and z around a wavelength of 400-550, 560-710, 700-850 and 830-100 nanometers respectively. See Figure 2.5.

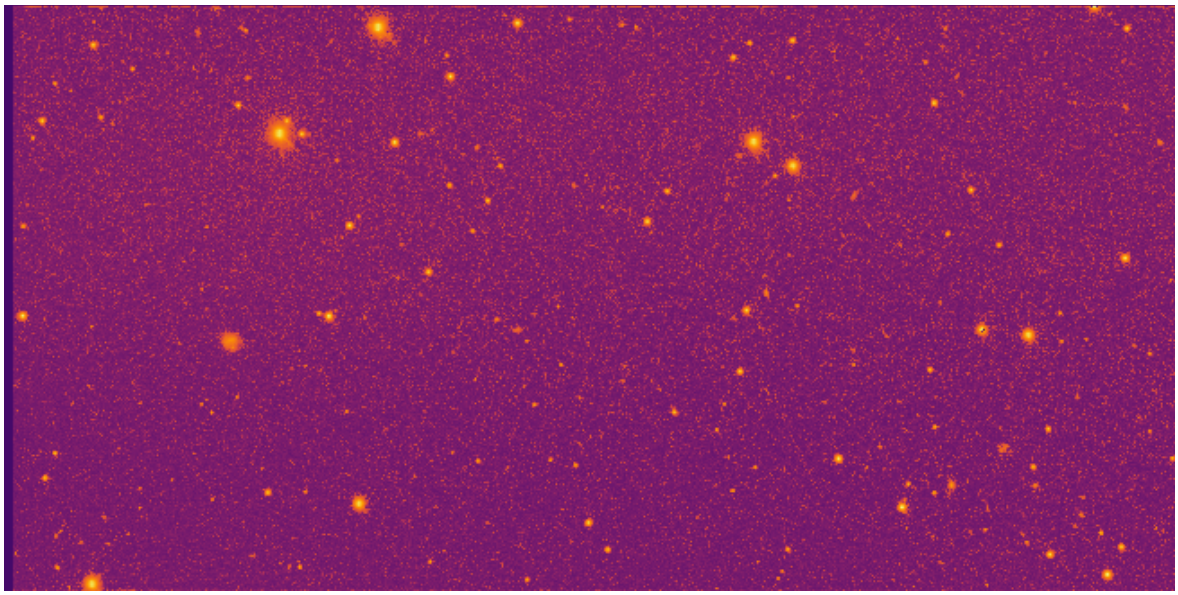


Figure 2.6: HiTS Survey image example. This image is one CCD out of 62 CCD array of DECam, with a resolution of 2048 by 4096 pixels, pre-processed for visualization purposes.

Observation Conditions

The observation conditions are the variables related to the atmospheric conditions when the astronomical image is taken. It is strictly related with the observation plan which defines which region of the sky the telescope will point at, at what time and in which band. Depending on the time and the region of the sky observed, the important parameters related to the image quality are the following:

- **Sky brightness**: It is the light emitted by the atmosphere by scattering and diffusion of light coming from different sources. The sky brightness varies depending on several sources, the Sun and Moon position when the image is taken and light pollution from near town or cities. The units of the sky brightness are [mag/arcsec^2] and can be converted to [ADU/pixel] depending on the need.
- **Point Spread Function (PSF)**: It is the function that describes how everything between the light source and the CCD camera spread the photons before arriving to the CCD. This can also be understood as the response of the atmosphere and the telescope to a point source modeled as a delta function. One of the important features used to describe the PSF is the full width at half maximum (FWHM), and it can be computed fitting a Gaussian function to an estimated PSF and finding the points where the Gaussian is half of its maximum value, then, the FWHM is the distance between those points. It is related width the standard deviation σ of the Gaussian by the expression $\text{FWHM} = 2\sqrt{2\ln 2}\sigma$. For a given exposure time, the astronomer usually registers the FWHM of the PSF at the zenith as shown in Figure 2.7. The FWHM is called **seeing** when is measured at the zenith and it partially characterizes the quality of the atmospheric conditions for astronomical observation. Larger seeing means worse observation conditions since the light from the source will be more spread. In Figure 2.7 we show an example of how the point spread function affects the image for different FWHM.
- **Zero Point**: Constant that shifts the conversion between magnitude and flux shown in equation 2.1. Computed to match the apparent magnitude of stars with a given reference magnitude. Ideally, one zero point should be calculated for each exposure, but sometimes it is approximated as a constant value for all exposures, this is not always a good approximation since the zero point also contains important information related to losses in flux due to atmosphere dispersion and differential color refraction.
- **Airmass**: A measure of the amount of atmosphere between the telescope and the source. It is estimated by computing the zenith angle shown in Figure 2.7 denoted by θ, then taking the airmass as $\sec(\theta)$. Observing with an angle far from zenith will produce large airmass estimation, observing at zenith has an airmass of 1. Higher airmass increase the magnitude resulting in a fainter source, this makes sense since a larger amount of atmosphere will scatter more light before arriving at the camera.
- **Limiting Magnitude**: The maximum relative magnitude necessary for an object in order to produce a signal within the image that is 5 standard deviation above the noise. Objects with a magnitude lower than the limiting magnitude will produce a considerable signal and will be considered as a detection.

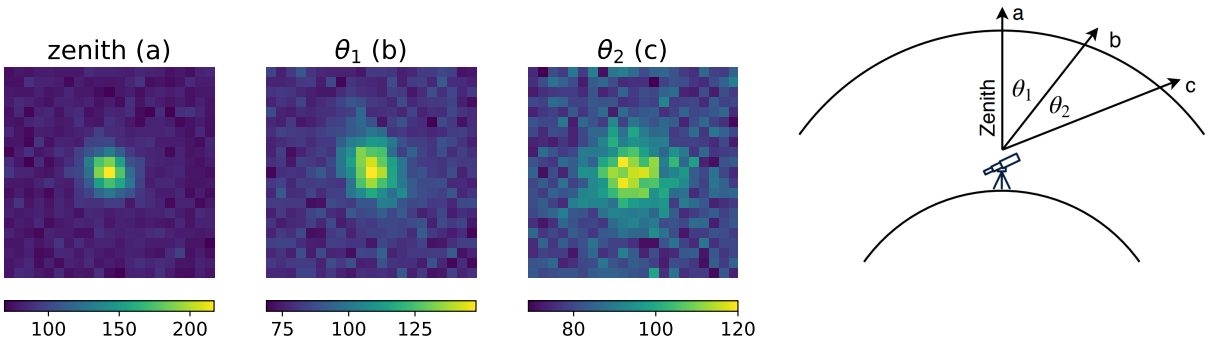


Figure 2.7: Zenith angle and distortion depending on the angle of observation. For the same source, observations closer to the zenith will have better quality in most of the cases since the light goes through less amount of atmosphere as compared to observing in different angles θ_1 and θ_2, for example. Here, from direction a to c, the airmass increases and also the FWHM of the PSF. The spreading of the light is larger and the flux loss due to scattering in the atmosphere is higher.

Noise sources

Astronomical images have different sources of noise, some of them are related to observation conditions, others with the quality of the camera. In this section, we explain each source of noise, its modeling and how affects the signal to noise ratio of the image. For a more detailed explanation see `http://www.ucolick.org/~bolte/AY257/`. For this section, we will use r as the radius of aperture in pixels, that is the area of the image where we are measuring the signal, t_{exp} as the exposure time, $n_p = \pi r^2$ is the number of pixels in the aperture. All these noise effects apply to electron charges $C[\text{e}^-]$, so to return to ADU units we divide by the gain G.

- **Source noise**: It is the intrinsic noise of the source, which is the astronomical object that we are observing. The electron charges arrived to the CCD C in $[\text{e}^-]$ units coming from the source distributes as Poisson, so $E(C) = Var(C)$ where $E(C)$ is the expectation value and $Var(C)$ is the variance. Considering the registered flux of the source $F = C/t_{\text{exp}}$, then the standard deviation of the source is $\sigma_s = \sqrt{F \cdot t_{\text{exp}}}$.
- **Sky noise**: As described in Section 2.1.4, the sky has its own brightness produced by scattering of light coming from different sources such as the moon or nearby cities. It could be considered as a counting noise, so it is often approximated as an independent Poisson distribution per pixel. Actually, there is a correlation of sky noise between close pixels within the CCD camera due to the point spread function. For further explanation see Section 2.1.6. Denoting the sky flux per pixel as F_{sky} in $[\text{e}^-/\text{pixel}/s]$, then the sky noise within the aperture area could be taken as $\sigma_{\text{sky}} = \sqrt{n_p \cdot F_{\text{sky}} \cdot t_{\text{exp}}}$.
- **Read noise**: According to the previous section, read noise is modeled as a Gaussian distribution and characterized by its standard deviation σ_{readout} in $[\text{e}^-]$ units and zero mean. Then, the noise within the aperture area is $\sigma_r = \sqrt{n_p \cdot \sigma^2_{\text{readout}}}$.
- **Dark current**: Usually produced by random generation of holes an electrons in a semi-conductor material. This source of noise could be modeled as a positive mean Gaussian distribution, but it has a heavy tail towards higher noise values [65]. Dark

current σ_{dark} has [e$^-$/pixel/s] units, then the noise produced by dark current within the aperture area is $\sigma_{\text{d}} = \sqrt{\sigma^2_{\text{dark}} \cdot n_p \cdot t_{\text{exp}}}$

- **Extra sources**: Astronomical images may contain other light sources besides the target astronomical object. For instance, supernovae explosion tend to occur near a host galaxy, which increases the background brightness and thus the noise in the image. A simple way to approximate the noise due to extra source in the image is by considering the light of the extra source that falls into the aperture area, then add their variance considering Poisson noise. If $s(x_i, y_j)$ is the flux rate of the source in pixel coordinates x_i, y_j, then the noise of the extra source within the aperture area A is

$$\sigma_{\text{extra}} = \sqrt{\sum_{i,j} s(x_i, y_j) \cdot t_{\text{exp}}} \tag{2.6}$$

  where for every (i, j), $(x_i, y_j) \in A$. In the latter expression, pixel correlation is not considered.

Using all previously discussed noise sources, the signal to noise ratio of the source image is defined as:

$$S/N = \frac{F \cdot t_{\text{exp}}}{\sqrt{\sigma_s^2 + \sigma^2_{\text{sky}} + \sigma_r^2 + \sigma^2_{\text{d}} + \sigma^2_{extra}}} \tag{2.7}$$

2.1.5 Difference Image

The basic idea behind the computation of the difference image is that if two images taken at different times but pointing to the same region of the sky are subtracted, everything that varied between the two images will remain, and everything unchanged will be canceled. This procedure is very suitable to find transients like supernovae and asteroids. Usually, for a sequence of images of a source, a **reference image** is chosen, then every other new image of the object, the **science images**, are compared against the reference image.

The procedure to compute the difference image can be summarized as the following:

1. Align the two images, i.e., move the center of each source at the same image coordinate frame. This part is very important because errors in alignment could generate dipoles. An example of a dipole due to alignment error is depicted in Figure 2.8
2. Subtract the sky from both images, so the resulting average background of the image is statistically zero.
3. Estimate the PSF for both images and find a convolution kernel that brings the shape of the highest quality PSF (the one with lower FWHM) to the shape of the lowest quality PSF. In the example shown in Figure 2.8, the reference image has a larger FWHM as compared to the science image. This process is called "PSF Matching" since the kernel is used to change the PSF of one of the images to the PSF of the other image.
4. Convolve the kernel with the highest quality PSF image (science image in this example), resulting in a new science image, as if it had been seen with the PSF of the reference image.

5. Subtract both images, which are the reference image with background subtracted, and the science image with background subtracted and convolved with the kernel found in step 2.

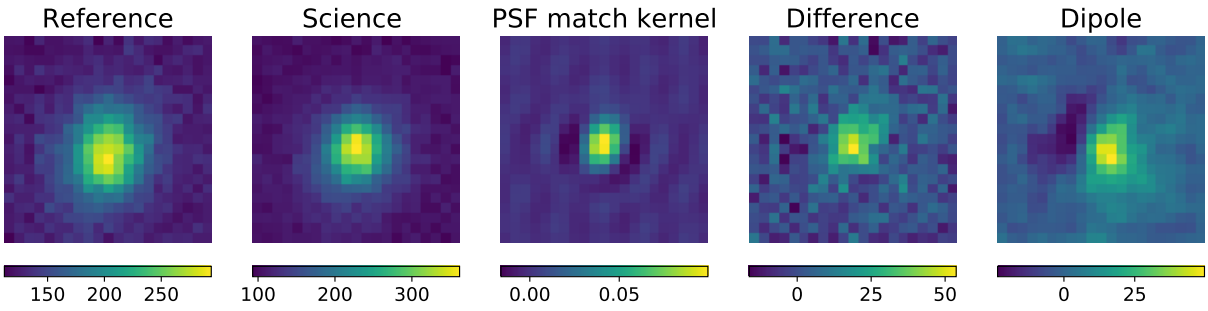


Figure 2.8: Difference image computation example. In this case, the reference image has a larger FWHM than the science image, so the PSF match kernel maps the PSF from science image to the PSF from reference image through convolution. The resulting difference image is the product form the procedure explained. The dipole shown here was simulated by generating an alignment error of 0.5 pixels.

2.1.6 Light Curve Calculation

As mentioned in the previous section, computing the light curve of an astronomical object is useful to characterize it. In this work, Naylor's optimal photometry [8] is used to estimate light curve from a series of stamps, which are a reduced version of the original image with the source centered.

For each of the available stamps of an object, which could be the science or the difference image, the procedure to estimate the flux of a source from the image is the following (for further details, see Naylor's original paper [8]):

1. Estimate the PSF $P_{i,j}$ from the image, where (i, j) is the position in pixels. Align the center of the target source with the center of the PSF.
2. Compute the variance per pixel

$$V_{i,j} = V_s + \frac{D_{i,j} - S_{i,j}}{\sqrt{G}}, \tag{2.8}$$

   where $D_{i,j}$ is the original value of the image in the pixel (i, j) in ADU units, $S_{i,j}$ is the estimated background level (sky plus known extra sources), G is the camera gain, and V_s is the variance of the background. The variance of the background can be estimated directly from the image, or using assumptions about noise distribution as explained in previous section.
3. Compute the counts C_{ADU} from the source in ADU units as

$$C_{\text{ADU}} = \sum_{i,j} W_{i,j}(D_{i,j} - S_{i,j}), \;\; W_{i,j} = \frac{P_{i,j}/V_{i,j}}{\sum_{k,l} P_{k,l}^2/V_{k,l}}, \tag{2.9}$$

where $W_{i,j}$ is called weight mask, which gives more importance to pixels where the PSF has a higher value.

4. The error of the flux estimation C_{err} is

$$C_{\text{err}} = \sum_{i,j} W_{i,j}^2 V_{i,j} \tag{2.10}$$

5. Move from counts to flux $F = C_{\text{ADU}} \cdot G / t_{\text{exp}}$, then from flux to magnitude using equation 2.1.

The estimation of the variation of brightness of a source is a basic step to determine if the object is of interest or not. For example, if the flux F of the source is estimated to be n times larger than the estimation error F_{err} (usually $n = 5$, this is **five sigma detection**), then the source is considered to be an astronomical object. Furthermore, if the sigma detection is from the flux computed using the difference image, this means that an object increases its flux in time, and it is labeled as an "alert" since could be an interesting transient object to perform follow up, like a supernova explosion.

The light curve could be computed from the science image or the difference image. When the science image is used, extra sources, different from the target of photometry, may appear and they should be removed from the source. This could be complicated in cases where a known catalog of sources is not available since it is necessary to determine which part of the flux is coming from the target object or from an extra source. Figure 2.9 shows an example of a light curve extracted from a series of science image stamps. Using the difference image to compute the light curve presents the advantage that extra sources disappear in the resulting difference image if they have a constant brightness, however, dipoles due to alignment errors could trigger an alert of a transient.

2.1.7 Traditional Classification Methods

As we mentioned in Section 2.1.1, some research areas in astronomy require the classification of a large number of objects: e.g., supernovae are used to estimate cosmological distances for studies about the expansion of the universe [3, 4], and variable stars such as RR Lyrae or Cepheids are needed to map the structure of the Milky Way and serve as *cosmic distance ladders* [6, 5].

Traditional methods to classify variable astronomical objects are based on light curve computation. Then, features are extracted from the light curve, for example, the Lomb Scargle periodogram [66, 67], which is a Fourier transform for irregularly sampled signals, also statistics such as mean, variance, skewness and kurtosis, and model fitting to use the parameters of the model as features [13]. All these features were manually designed to discriminate between different type of astronomical objects by looking at the light curve, some of the features are known to be important for this task such as the period to discriminate among periodic stars. Examples of this methodology are [13, 53, 11, 12, 14, 15].

Sometimes, the manual design of expressive features for the classification task could be hard to accomplish. This requires a notion about what kind of features are more suitable to

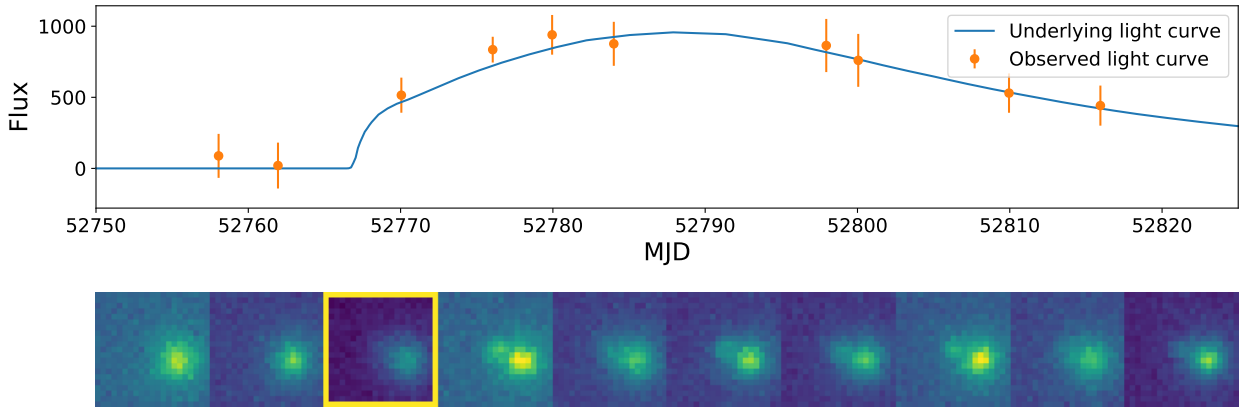


Figure 2.9: Light curve obtained from a series of stamps. Applying Naylor's photometry to each of images taken by a telescope forms a light curve shown in orange. Each point of the light curve is a noisy version of one point sampled from the underlying physical model. In this example, the alert is triggered in the third observation (frame 3 of stamps), since the flux is at least 3 times the background noise. Here, the science image is used to compute the light curve, so the host galaxy must be subtracted from the image (term $S_{i,j}$ in equation 2.9).

discriminate between classes, which is not always clear. Furthermore, to quantify the gain of adding a certain feature could require expensive computation, such as feature selection algorithms [68, 69], or training a classifier to compute feature relevance [70, 71, 72]. One way to avoid manual feature design is learning features directly from the data (either the light curve or the difference image). Representative features are learned during the training of the classification model, avoiding manual feature computation. This approach presents some disadvantages, since learned features could be hard to interpret and also could be biased in some undesirable way as a product of an artifact of the data. By manually designing features both problems can be controlled. Example of features learned from data for classification problems are [16, 17, 23, 22, 18, 21, 24].

2.2 Deep Learning

In this section, **Deep Learning** and related concepts are explained, starting with the basics of artificial neural networks. Then, convolutional neural networks and recurrent neural networks are visited. Next, important considerations during neural networks training are discussed, in order to present some techniques to improve the training process. Finally, deep learning methods applied to astronomical problems are reviewed.

2.2.1 Artificial Neural Networks

Artificial neural networks (ANN) are a special kind of machine learning method, which are inspired on the biological functioning of the brain. The building block for ANNs is the artificial neuron, also called **perceptron**, usually described as:

$$y = f\left(\sum_i w_i x_i + b\right), \tag{2.11}$$

where w_i are called the synaptic weights of the neuron that ponderates the input x_i, then the product is mixed with other inputs through a sum plus a bias term b, the combination of inputs goes through an activation function $f(\cdot)$ giving an output value y. A representation of a single artificial neuron is depicted in Figure 2.10a.

To build an ANN, a set of artificial neurons are grouped by layers, then each layer is stacked, so the input of the next layer is the output of the previous one. A single ANN neural network layer can be expressed with matrix notation as the following:

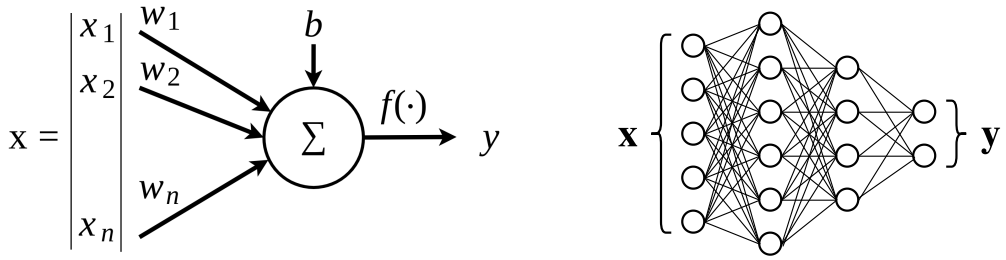
$$\mathbf{y} = f\left(W^T\mathbf{x} + \mathbf{b}\right), \tag{2.12}$$

where $\mathbf{x} \in \mathbb{R}^n$ with n the input dimension, $\mathbf{y} \in \mathbb{R}^m$ with m the number of neurons within the layer which also defines the output dimension. Here the weights of each neuron in the layer correspond to the columns in $W \in \mathbb{R}^{n\times m}$ and the bias of each neuron is $\mathbf{b} \in \mathbb{R}^m$. The activation function $f(\cdot)$ is applied element-wise to the vector. Typical choices for f are the logistic sigmoid, hyperbolic tangent or rectifying linear unit ReLU [73], shown in Figure 2.11 respectively. The layer in equation 2.12 is called **fully connected layer** since every output neuron is connected through a weight to all the components of the input $\mathbf{x}$.

An ANN is built by stacking layers. The inner layers, which are not the input or output are called hidden-layers. This kind of structure is known as multi-layer perceptron (MLP). For example, the output of a 2-layer perceptron can be expressed with a matrix notation as the following:

$$\mathbf{y}_{\text{out}} = f_2\left(W_2^T f_1\left(W_1^T\mathbf{x} + \mathbf{b}_1\right) + \mathbf{b}_2\right). \tag{2.13}$$

If $\mathbf{y}_{\text{out}} \in \mathbb{R}^m$ and $\mathbf{x} \in \mathbb{R}^n$, then it has been proven that the MLP in equation 2.13 is an **universal approximator** for any continuous function $g : \mathbf{x} \in \mathbb{R}^n \to \mathbf{y} \in \mathbb{R}^m$ in a compact subset of $\mathbb{R}^m$, and under certain assumptions on the activation function [74, 75]. Because of this property, ANNs are used for classification or regression tasks where g maps, for



(a) Single neuron. (b) Multi-layer neural network.

Figure 2.10: Graphical representation of (a) a single artificial neuron or perceptron and (b) a MLP network.

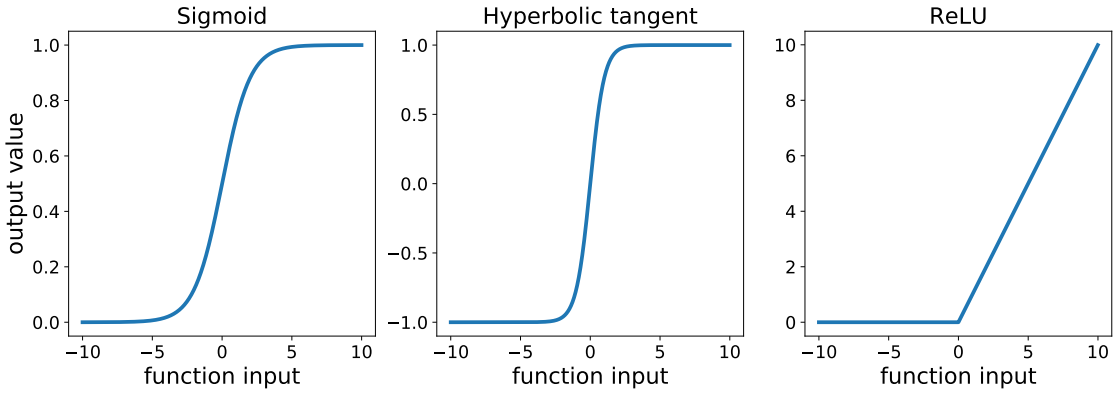


Figure 2.11: Typical activation functions.

example, a handwritten digit image to a label that represents the number in the image [76]. A representation of a multi-layer perceptron is depicted in Figure 2.10b.

The universal approximation theorem states that a value of W and b exists that approximate g, but it doesn't specify which are the specific values of W, b and number of hidden layers. In order to find W and b for each layer that map a target function $g : \mathbf{x} \in \mathbb{R}^n \rightarrow \mathbf{y} \in \mathbb{R}^m$, a dataset of examples of input-output of g, $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$ are required. Denoting the function of the MLP as $h : \mathbf{x} \rightarrow \hat{\mathbf{y}}$, it is desirable that $\forall i,\ h(\mathbf{x}_i) = \hat{\mathbf{y}}_i$ is close to $\mathbf{y}_i$. A loss function is defined to evaluate how close is $\mathbf{y}$ to $\hat{\mathbf{y}}$. For classification problems with M classes, $\mathbf{y}, \hat{\mathbf{y}} \in \mathbb{R}^M$ are the probabilities of $\mathbf{x}$ belonging to a certain class c. Usually, $\mathbf{y}$ has a one-hot encoding format, where $y_c = 1$ if the sample belongs to class c and 0 elsewhere. For classification problems, the most common choice to compare $\mathbf{y}$ and $\hat{\mathbf{y}}$ is the cross entropy $H(\mathbf{y}, \hat{\mathbf{y}})$, leading to the average cross entropy L_{cross} along all the examples as loss function, as follows:

$$L_{cross} = \frac{1}{N} \sum_{i=1}^{N} H(\mathbf{y}_i, \hat{\mathbf{y}}_i), \qquad H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{c=1}^{M} y_c \log(\hat{y}_c). \tag{2.14}$$

For regression problems, the most common choice is the mean square error (MSE):

$$L_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{y}_i - \hat{\mathbf{y}}_i)^2. \tag{2.15}$$

The smaller the loss function, the closer are $\mathbf{y}$ and $\hat{\mathbf{y}}$. To find the parameters of the MLP

that better maps the samples, the loss function must be minimized. For a MLP with more than one layer, the loss function is non-convex in the network parameters, this means that the function could have more than one minimum (or maximum) value. The loss is minimized by iteratively computing the gradient $\nabla_W L$ using the back-propagation algorithm [77] (using the chain rule for neural networks), then using the gradient, it is possible to find a new network W^{t+1} that leads to a lower loss value by computing:

$$W^{t+1} = W^t - \alpha \nabla_W L \tag{2.16}$$

where t is the iteration index, and α is called the learning rate. This method to minimize the loss function is called gradient descent. When the dataset is too large, not all the examples within the dataset can be used to compute the gradient mainly due to GPU/RAM memory or computation time, so each iteration is done by using a mini-batch of the data, with fewer samples that estimate the gradient, this leads to the stochastic gradient descent method. The process of finding the weights and biases of a network iterating over the samples, and computing the gradient to correct the weights as explained here is called **training**.

As we mentioned before, since the loss is a non-convex function, this optimization could lead to a local-minimum solution. To speed up the procedure, there are other versions of the update rule in equation 2.16, for example, stochastic gradient descent with momentum [78]. There also variations that have an adaptive learning rate, such as Adagrad [79], Adadelta [80], ADAM [81], AMSGrad [82], among others.

One of the problems that an MLP has during the training process is the **vanishing gradient**. When more than two or three layers are stacked and trained, the gradients for the input layers (the ones further from the output, i.e., the error between $\mathbf{y}$ and $\hat{\mathbf{y}}$) gets smaller. Consider the graph shown in Figure 2.12 that represents a single layer of a ANN, where y_{l-1} is the output of the previous layer $l-1$, W_l, b_l are the weights and biases for layer l, $z_l = W_l \cdot y_{l-1} + b_l$ are the weighted input sum before the activation function $f_l(\cdot)$ and $y_l = f_l(z_l)$ is the layer output. For simplicity, consider every quantity as a scalar (this does not harm the conclusion), by taking the derivative of the output y_l with respect to the weights of k layers before W_{l-k}, the following is obtained:

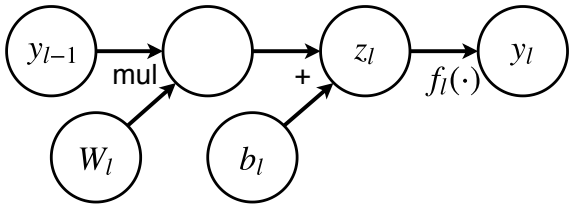


Figure 2.12: Graph representation of a single ANN layer, where **mul** denotes matrix multiplication

$$\frac{\partial y_l}{\partial W_{l-k}} = \frac{\partial y_l}{\partial y_{l-1}} \frac{\partial y_{l-1}}{\partial y_{l-2}} \cdots \frac{\partial y_{l-k}}{\partial W_{l-k}} \tag{2.17}$$

$$= (f_l{}'(z_l) \cdot W_l)(f_{l-1}{}'(z_{l-1}) \cdot W_{l-1}) \cdots (f_{l-k}{}'(z_{l-k}) \cdot y_{l-k-1}) \tag{2.18}$$

$$= \underbrace{\left[\prod_{i=l-k}^{l} f_i{}'(z_i)\right]}_{f^*} \underbrace{\left[\prod_{i=l-k+1}^{l} W_i\right]}_{W*} y_{l-k-1}, \tag{2.19}$$

where $f_l{}'(z_l)$ denotes the derivative of the activation function of the layer l. If $f_l(\cdot)$ for any layer is either a sigmoid function or an hyperbolic tangent, then its derivative takes values $f_l{}'(\cdot) \in (0, 1)$ and the first term f^* in equation 2.19 gets smaller while k increases, this means that the derivative $\partial y_l / \partial W_{l-1}$ gets smaller when more k layers are stacked in the architecture. This is one of the reasons to prefer ReLU functions rather than sigmoid or hyperbolic tangent, the last two present saturation regions where the input $|x|$ is large enough, while ReLU only saturates for $x < 0$. Furthermore, if the value of the weights distributes as a Gaussian $\mathcal{N}(0, 1)$, then the second term W^* that multiplies the values of the weights W_l of each layer will be small too. Small values of the gradient does not allow to train the layers that are too far from the output, this is the vanishing gradient problem.

In 2006, Geoffrey Hinton showed that it is possible to train deep architectures with more than 3 layers by initializing in a "good starting point" for the optimization by using Restricted Boltzmann Machine [83, 84]. The problem of the vanishing gradient was not attenuated but bypassed, by the fact that the intermediate representation of hidden layers was already good because of the smart initialization. This was the proof that deep architectures can be trained effectively and that they perform better than shallow ones, starting the field known as **Deep Learning**. Since this breakthrough, deep architectures in terms of the number of layers and time dependencies, are being applied in different fields producing state of the art results. Deep learning methods are particularly useful when a large amount of data is available. In addition, when good features to extract from the data and solve the problem are not clear, deep architectures learn the features directly from the data. A variety of architectures have been developed that solve the vanishing gradient in different ways, also adapted for different input data formats, training performance in term of accuracy or computation time, the goal of the task such as classification, clustering or data generation. Some of the architectures and their advantages are described in the following sections.

2.2.2 Convolutional Neural Networks

For a certain type of data, where features are spatially or temporally correlated, there is a special kind of neural networks called Convolutional Neural Networks (CNN). They are based on the biological visual system of animals [85, 86] and they have been applied to spatially correlated data such as images [26, 27] and temporal correlated data such as audio [28, 29] among others.

The convolutional layers within the CNN apply convolutional operations to their inputs. The most popular one is the convolution over images done by adaptive filters, which are adjusted through the training process. The convolution $\mathbf{y}$ of an image $\mathbf{x}$ with a filter W plus

a bias b is expressed as:

$$y_{i,j,k} = \sum_{m,n,p} x_{i-m,j-n,p} \cdot W_{m,n,p,k} + b_k, \tag{2.20}$$

where $\mathbf{x}$ and $\mathbf{y}$ are 3D tensors, with values x and y respectively, i, j are the image coordinates, the indexes m, n run over the filter and the index p runs over the depth or the number of input channels, and k is the number of output channels. The filters move over the image by skipping a defined number of pixels in the spatial dimension, this is called stride. The filters W have four dimensions, height, width, number of input channels and number of output channels. W is usually chosen to be much smaller in the height and width dimension than the image, 3×3 or 5×5 for example. On the other hand, the output channel is commonly much larger than the input channel. For example, for an RGB image, the number of input channels is 3, architectures with 32, 64 or even more output channels per layer are frequent [26, 27]. Right after the convolution operation, a non-linear activation function is used, the same way as in the fully connected layer. An example of the convolution operation is shown in Figure 2.13, where a single channel is used for simplicity, but the convolution may also occur in the channel dimension.

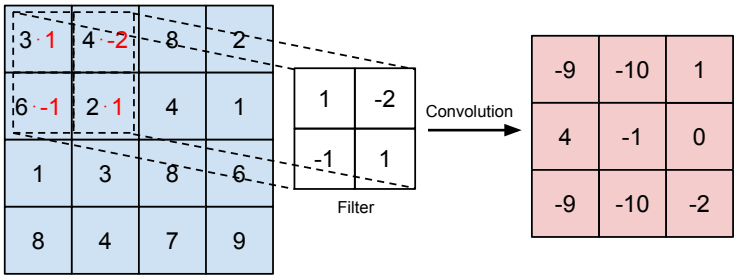


Figure 2.13: Convolution operation example. Using stride 1 and a single channel for both, the input image and the filter.

Convolutional layers have the interesting property of invariance to a translation of features within the image, because the same weights of the filter are applied to different groups of pixels, this is called **parameter sharing**. For example, if the filter gets activated when finding a corner in a certain direction within the image, then the same corner found in another part of the image will produce the same activation of the filter. Furthermore, the fact that the filter is much smaller in the first two dimensions means that it is "seeing" a small portion of the image detecting local features, then the next layer sees a bigger portion of the image and is able to combine features detected from the previous convolutional layer. In practice, it has been observed that layers closer to the input detect lines and textures, then the next layer uses the previous layer activations to detect corners or curved lines, the complexity of the detected features increases as more layers are stacked in the network [87].

Another type of operation used in CNNs are the **pooling layers** [88]. These layers are applied to reduce the dimensionality of the representation through the network and act as a regularizer. In this work, max pooling operations were used, which return the maximum value within a sub-matrix of the image, $Y_{i,j} = \max\left(X_{i:(i+n),j:(j+m)}\right)$ where i, j are the image coordinates and n, m are the coordinates of the sub-matrix. An example of the pooling operation is shown in Figure 2.14.

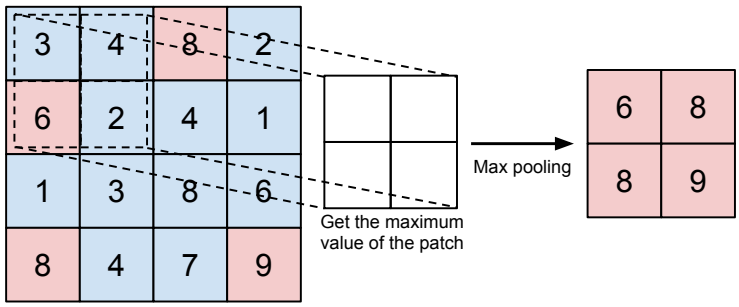


Figure 2.14: Max pooling operation example. In this case, the stride is 2, so the patch moves every two pixels of the input image and selects the largest value within the patch.

Different kinds of architectures have been designed. For example, a special kind of CNN called **Deep Residual Neural Network** or ResNet [89], which bypasses the input $\mathbf{x}$ to the output of the convolutional layer $\mathbf{y}$, this is:

$$\mathbf{y} = \mathbf{x} + \text{Conv}(\mathbf{x}) \tag{2.21}$$

where $\text{Conv}(\cdot)$ denotes the regular convolutional layer. By doing this, the error gradient can go through the bypass directly, allowing architecture training with hundreds of layers without suffering from the vanishing gradient problem. Another example of architecture is the **Densely Connected Convolutional Network** or DenseNet [90], which connects each convolutional layer to every other layer, mitigating the vanishing gradient problem.

2.2.3 Recurrent Neural Networks

Just as CNNs are specialized to process spatially correlated data, Recurrent Neural Networks or RNNs are used to process sequential data or time series. There are many types of RNNs [91] but most of them have a feedback connection to the input from previous time steps or a state (or both), where a state is an arbitrary representation of a memory of previous inputs. This recurrent structure can be expressed as $\mathbf{y}_t, \mathbf{s}_t = h(\mathbf{x}, \mathbf{s}_{t-1})$, where $\mathbf{y}_t$ is the model output at time t, $\mathbf{x}_t$ is the input, $\mathbf{s}_t$ is the state (which could also include the output y_t) and $h(\cdot)$ is the forward function of the RNN. An example of a simple RNN graph is shown in Figure 2.15. There are two representation, the forward graph and the unfolded graph, the latter shows the time dependencies explicitly.

In order to compute the gradients and train this kind of networks, back-propagation is performed through time and the network forward operation. In principle, the back-propagation through time (BPTT) could be computed from the last point in the time series $\mathbf{y}_N$ to the first one $\mathbf{y}_1$. This is not always possible due to memory and computational time issues. In practice the back-propagation in time is truncated to a time shorter than the total length of the time series. For example, for the unfolded network shown in Figure 2.15 the gradient could be propagated until k time steps in the past from t.

For long time dependencies, the BPTT has the following problem. Consider the recurrent neural network layer shown in Figure 2.16 and expressed in equation 2.22. If the dependency

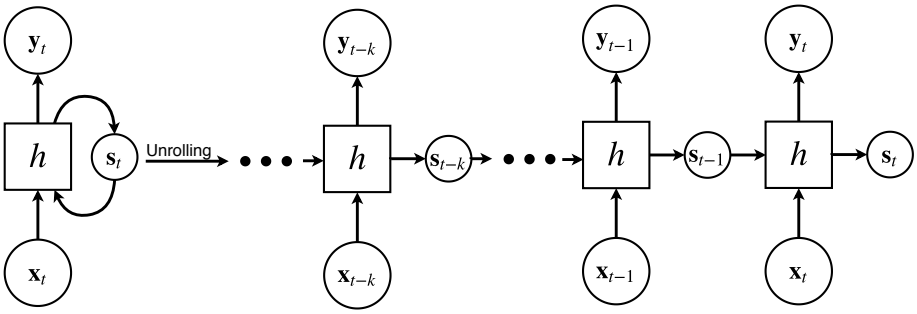


Figure 2.15: Recurrent neural network representation. The graph on the left represents the recurrent structure of $\mathbf{y}_t, \mathbf{s}_t = h(\mathbf{x}_t, \mathbf{s}_{t-1})$, which is unfolded on the right side of the figure to depict the time dependencies of previous inputs and states explicitly.

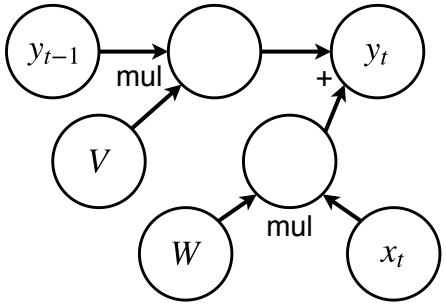


Figure 2.16: Recursive layer graph, the output $\mathbf{y}_t$ depends on the current input $\mathbf{x}_t$ and the previous output $\mathbf{y}_{t-1}$.

of $\mathbf{y}_t$ is explicitly unrolled until k previous time steps, then equation 2.26 is obtained,

$$\begin{aligned}
\mathbf{y}_t =& W^T\mathbf{x}_t + V^T\mathbf{y}_{t-1} && (2.22)\\
=& W^T\mathbf{x}_t + V^T\left(W^T\mathbf{x}_{t-1} + V^T\mathbf{y}_{t-2}\right) && (2.23)\\
=& W^T\mathbf{x}_t + V^TW^T\mathbf{x}_{t-1} + \left(V^T\right)^2\mathbf{y}_{t-2} && (2.24)\\
=& W^T\mathbf{x}_t + V^TW^T\mathbf{x}_{t-1} + \left(V^T\right)^2\left(W^T\mathbf{x}_{t-2} + V^T\mathbf{y}_{t-3}\right) && (2.25)\\
&\vdots\\
\mathbf{y}_t =& \sum_{i=0}^{k}\left(V^T\right)^i W^T\mathbf{x}_{t-i} + \left(V^T\right)^{k+1}\mathbf{y}_{t-k}, && (2.26)
\end{aligned}$$

Let V admits an eigenvalue decomposition $V^T = B^TAB$ and replacing in equation 2.26, we get:

$$\mathbf{y}_t = \sum_{i=0}^{k} B^TA^iBW^T\mathbf{x}_{t-i} + B^TA^{k+1}B\mathbf{y}_{t-k}, \quad (2.27)$$

with eigenvalues λ_i and eigenvectors $\mathbf{a}_i$ (taken from The Deep Learning Book, Chapter 10 [92]). Then, the gradient with respect to any parameter of the network for longer dependencies than k will be multiplied by A^{k+1}. If any of the values $\lambda_i < 1$, then the vanishing gradient problem will arise for directions aligned with $\mathbf{a}_i$ as k increases. If $\lambda_i > 1$, then the gradient

will increase by a factor of λ_i^{k+1} in the corresponding direction $\mathbf{a}_i$, this is called **exploding gradients**.

In this work, we use a special kind of recurrent model called **Long-Short Term Memory** or **LSTM** [30, 31, 32]. The main characteristic of LSTMs is the use of gates that control the content of the state in order to learn longer time dependencies than conventional recurrent networks. The LSTM cell (or layer) is described by the equations 2.28 to 2.32. For a given input vector $\mathbf{x}_t$ and previous outputs $\mathbf{h}_{t-1}$, the LSTM controls what is written and deleted from the cell state c_t using three gates: the forget gate f_t in equation 2.28 removes part of the state using the information from current input and previous output, the input gate i_t in equation 2.29 updates the state, and the output gate o_t in equation 2.30 combines input, state, and previous output to give a new output $\mathbf{h}_t$. Each gate uses weights W_f, W_i, W_o, W_c that multiplies the current input, weights U_f, U_i, U_o, W_c that multiplies the previous outputs, biases b_f, b_i, b_o, b_c and sigmoid function σ. The gate mechanism of LSTM allows the cell state c_t in equation 2.31 ($*$ denotes the element-wise product) to keep track of long-term dependencies within a sequence, this can be easily achieved by saturating the sigmoid function of the forget gate to 1 in the right components. A graphical representation of an LSTM cell is depicted in Figure 2.17. Some variants of LSTM are, for example, the **Gated Recurrent Unit** [93] and the **Phased LSTM** [94].

$$f_t = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + b_f) \quad \text{Forget gate} \quad (2.28)$$

$$i_t = \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + b_i) \quad \text{Input gate} \quad (2.29)$$

$$o_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + b_o) \quad \text{Output gate} \quad (2.30)$$

$$c_t = f_t * c_{t-1} + i_t * \tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1} + b_c) \quad \text{Cell state update} \quad (2.31)$$

$$\mathbf{h}_t = o_t * \tanh(c_t) \quad \text{Output vector} \quad (2.32)$$

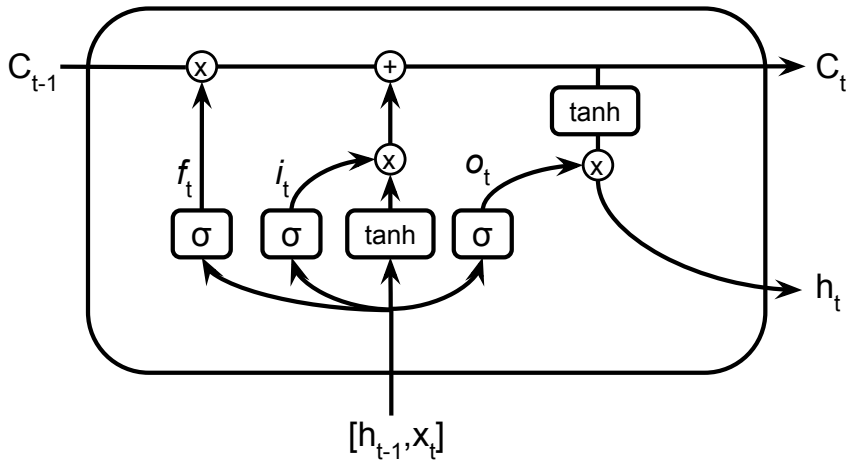


Figure 2.17: LSTM graphical representation. Boxes with activation functions σ or tanh denotes a fully conected layer as shown in equations 2.28 to 2.31.

2.2.4 Deep Learning Training

Usually, deep neural network architectures present a large number of free parameters that can be tuned to map any function. The larger the number of free parameters θ, the larger the space of functions that is able to map, also called the model **capacity**. If the capacity of a model $\hat{\mathbf{y}} = h_\theta(\mathbf{x})$ is high enough, through training described in Section 2.2.1, it is possible to achieve an arbitrary small value of the loss function to any sufficiently small set $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$ that are examples of a function $g : \mathbf{x} \in \mathbb{R}^n \rightarrow \mathbf{y} \in \mathbb{R}^m$. This is not always desirable since the estimated map h_θ could be adjusted to only map the available data, instead of learning the underlying structure given by g, so when new data is fed to the model for inference $\hat{\mathbf{y}} = h_\theta(\mathbf{x}_{\text{new}})$, $\hat{\mathbf{y}}$ it will be very far from the real value $\mathbf{y}_{\text{new}}$, since the model is only able to map the presented data during training. This phenomenon is called **overfitting**, and the capacity of a model to describe correctly new data not presented during the training is called **generalization**. The inverse effect is called **Underfitting** and it occurs when the capacity of the model is not large enough to fit the training data.

In order to monitor the training process and overfitting, the available data is usually split into three subsets, the training set used to find the parameters of the model, the validation set used to monitor the generalization capacity of the model, and the test set which is used only to evaluate the model. Decisions about the model design and training procedure could be led by its performance on the validation set, this is an undirected flow of information from the validation set to the final model. The test set is the part of the data that the model never "sees" in any way to choose the parameters.

The example shown in Figure 2.18 consists of simulated data sampled from a parabola, the data was divided into training and test set. Then, the training data was used to fit a polynomial model $y = \sum_{i=0}^{N} \alpha_i x^i$. In this cases, models with $N = 1, 2, 5, 10$ were used. The model with $N = 1$ is not able to correctly fit the training data since the capacity of this model is to low causing underfitting. For $N = 2$ the model approximate reasonably well both, the training data and validation data. For $N = 5$, the model map perfectly some of the points in the training data, for $N = 10$ the model maps every point in the training data, but the latter two models perform poorly in the validation data since they are overffited to the training data.

One way to prevent overfitting is by reducing the capacity of the model until a good compromise between the performance in training and validation is reached. The process of reducing the capacity of a model to prevent overfitting is called **regularization**. There are many methods to regularize a model, for example:

- Reducing the number of free parameters of the model.
- Reducing the search space of possible solutions by restricting the values of possible parameters. One way of doing this is by adding a penalization term on the loss function, such as $\|\theta\|_2$ or $\|\theta\|_1$, these are called **Ridge** [95] and **Lasso** [96] regularization respectively. For example, in the case of $\|\theta\|_2$, the larger the norm of the parameters, the larger the loss function value, then points of θ that are closer to the origin are preferred during the optimization, excluding θ with large norm from the search space. The extra term added to the loss function is usually $\lambda\|\theta\|_2$, where λ is a constant that

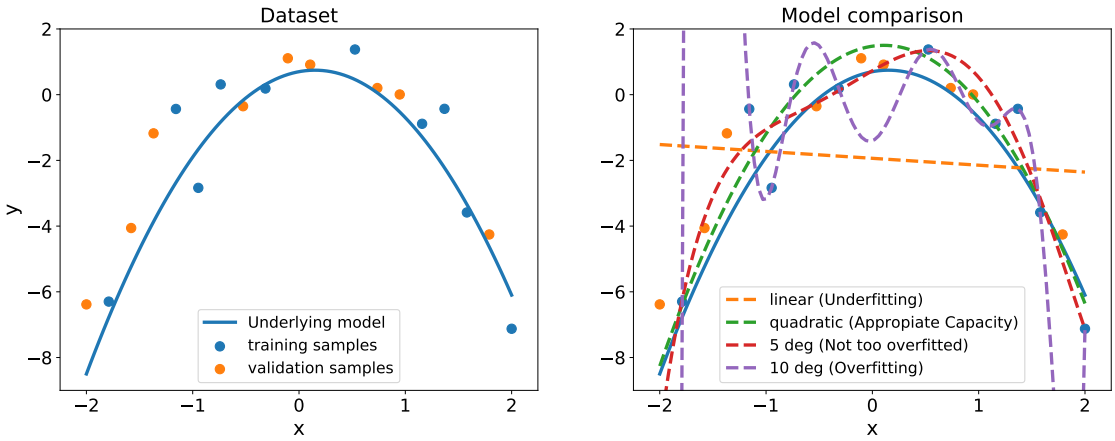


Figure 2.18: Polynomial models with different capacity fitted to (a) simulated data. The capacity of (b) each model is regulated by the degree of its polynomial function, for degrees with $N = 1$ the model is underfitted, $N = 2$ approximates the correct solution and $N = 5, 10$ are overfitted due to the excess of model capacity.

controls the weight of the penalization in the loss function.

- Stopping training iterations before the model starts to overfit to the training data. This is called **early stopping**. For iterative methods, it is convenient to monitor the loss function for training and validation set while the model is being trained. If a further adjustment to the training data is increasing the loss function for the validation data, then the training is stopped. This method can be understood as reducing the number of functions that the model can map by limiting the exploration time.
- For ANNs, turning off some neurons during training acts as a regularized. This method is known as **Dropout** [97]. The method consists of keeping a neuron with a probability p on each iteration, the rest are turned off. By doing this, for each iteration, a thinned network is sampled and trained, sharing variables between each sampled network. Then, during inference, all the neurons are turned on and the expected output is computed by multiplying each weight by the probability of being active during training p. This method produces some sort of mixed network using the thinned ones sampled during training, in practice, this reduces the overfitting, acting as a regularization.

Another aspect of neural network training is the **data normalization**. The most common way to normalize the input data is by doing $\mathbf{x}' = \frac{\mathbf{x}-\mu}{\sigma}$, where $\mathbf{x}$ is the original input data, μ is the mean of the available data, σ is the standard deviation and $\mathbf{x}'$ is the normalized data. In general, normalizing the data helps the training process since the model does not need to learn the scale of the data, but the relative difference between data examples. Another way to deal with the data scale is by using **Batch Normalization** [98] which standardizes the values of the internal representation by shifting and scaling them. Batch normalization is sometimes used at the input of the network in order to normalize the input data, and also it is used between layers to normalize internal representations. Some of the effects of batch normalization are faster training speed and better regularization. In this work, a variant of batch normalization was used called **Batch renormalization** [99] that helps to match the normalization process for training and inference while preserving advantages from regular

batch normalization. The batch renormalization operation during training is:

$$\mathbf{y} = \left(\frac{\mathbf{x} - \mu_{\text{batch}}}{\sigma_{\text{batch}}} \cdot r + \text{d} \right) \gamma + \beta, \tag{2.33}$$

where $\mathbf{x}$ is the layer input, $\mathbf{y}$ is the normalized version of the input, μ_{batch} is the mean and σ_{batch} is the standard deviation for the current batch, and γ and β are trainable parameters. The renormalization extension corresponds to $r = \text{clip}_{[1/r_{\max}, r_{\max}]} \left(\frac{\sigma_{\text{batch}}}{\sigma_{\text{d}}} \right)$ and $\text{d} = \text{clip}_{[-\text{d}_{\max}, \text{d}_{\max}]} \left(\frac{\mu_{\text{batch}} - \mu_{\text{d}}}{\sigma_{\text{d}}} \right)$, where μ_{d} and σ_{d} are the mean and standard deviation of the different inputs during training computed using a moving average $\mu_{\text{d}} = \alpha \mu_{\text{d}} + (1 - \alpha) \mu_{\text{batch}}$ with α a momentum parameter, r and d are taken as constant when computing the gradients and setting $r = 1$ and $\text{d} = 0$ we recover the original batch normalization. In the case of convolutional layer outputs, this operation is applied to each channel independently.

2.2.5 Deep Learning in Astronomy

Traditional methods based on features for astronomical object classification has been visited in Section 2.1.7. Deep learning methods learn features directly from the data, avoiding the manual design. These methods have been applied to astronomical problems using convolutional neural networks, for example, for real/bogus separation [16, 17], photometry computation [36], calculation of an image comparable to the difference image [18], gravitational wave detection [19] and exoplanet detection [20]. Recurrent neural networks have been used for light curve classification in [21, 22, 23, 24].

In the cases where stamps are used as input data, they have the advantage of giving all the information available from the image to the model. These methods usually analyze a group of stamps, science, template and difference image, which are samples of two points in time, not considering long-term dependencies between stamps. For some classification problems e.g., detecting periodic sources long-term dependencies are very important. Furthermore, the difference image described in Section 2.1.5 could present some problems such as dipoles.

Other deep learning models have been applied to light curves which were described in Section 2.1.6. Some of these methods use the light curve directly as inputs without computing designed features. Even though these models consider long-term dependencies by using many measurements from the same source, the light curve does not describe the local information within the image stamp.

The proposed methodology described in Chapter 3 mixes both approaches. It learns the features from a sequence of stamps without computing the difference image, in order to keep local information within the science image, but also learns the long-term dependencies that describe better some astronomical objects.

Chapter 3

Methodology

In this work, we use the sequence of science images directly as input to the classifier. The proposed methodology bypasses possible errors and loss of information produced by light curve and difference image computation.

One of the difficulties of the new approach is the availability of a sufficiently large training dataset. Given a certain survey, it is not always possible to have a large amount of the entire sequence of images for each type of objects, especially for rare objects like supernovae. Astronomical surveys usually have biases in terms of the number of specific objects found depending on scientific goals and observation limitations. Moreover, the properties of the images highly depend on the survey. For example, the cadence, sky brightness or zero points, are different among different surveys. This issue difficults the use of a model learned in one survey to another. This problem is known as **Transfer Learning**. One way to elude or at least facilitate the solution to the aforementioned problems, the number of real samples available and the transfer learning problem between surveys, is by simulating images for each of the classes of astronomical objects considering realistic weather conditions and instrument specification of a specific survey. In Section 3.1, the image simulation process we used to build a training set is detailed. Simulating the dataset presents some advantages at the cost of imposing additional assumptions.

Having the necessary information, it is possible to simulate images for any survey in the optical range that uses CCD cameras. The information needed for the simulation are **observation conditions** and **camera parameters** visited in Section 2.1.4, then the image properties of a survey can be reproduced. This has an implicit assumption, which is that the physical process of the real image formation is sufficiently well known. The image formation process is constituted of many effects but only the main ones are modeled in this work. Due to discrepancies between the simulated dataset distribution and real images distribution a **Domain Adaptation** problem arise. The model trained to solve the classification task in the simulated dataset is adjusted to the domain of the simulations. To solve the same task for the real dataset, the difference between the domain of simulated data and real data has to be considered, since the simulation does not take into account every physical effect, nor has exactly the same distribution compared to real data. Weather conditions forecasting errors must be considered too, since the conditions used for simulating the data are probably not

exactly the same as the real data.

If a new class of astronomical object is needed in the dataset, the only necessary information is a model of the light curve in the corresponding bands, along with a distribution of parameters to sample from and get different instances of the same astronomical object. When choosing the parameters distribution of the light curve model, the survey must be considered to avoid undesirable effects when simulating. For example, the magnitude distribution of the object is related with the maximum depth that the survey is able to see, characterized as the limiting magnitude. If a considerable part of the magnitude sampled from the distribution lies above the limiting magnitude, then the simulated image sequences would not have any significant signal from the source, becoming useless or even harmful for training a classifier. In the case of periodic stars, it could happen that the time span and cadence of the survey is not able to observe a complete period of the object or either some of the periodic components such as eclipses in the cases of eclipsing binaries. In the last example, if the eclipse of an eclipsing binary is not observed because of the time span or cadence, then the resulting light curve will be indistinguishable from constant stars. As these examples show, there could be other important effects related to the prior distribution of object parameters.

Once the image sequence dataset is simulated, a Recurrent Convolutional Neural Network (RCNN) is proposed as a classifier, explained in detail in Section 3.2.2. The proposed model is trained on the simulated dataset, then by using a few real samples, the model is fine-tuned [41, 42] this allows adapting the model from the simulated domain to the real domain. In addition, for comparison purposes a Random Forest model is trained using the recovered light curve from stamps, as explained in Section 3.2.3.

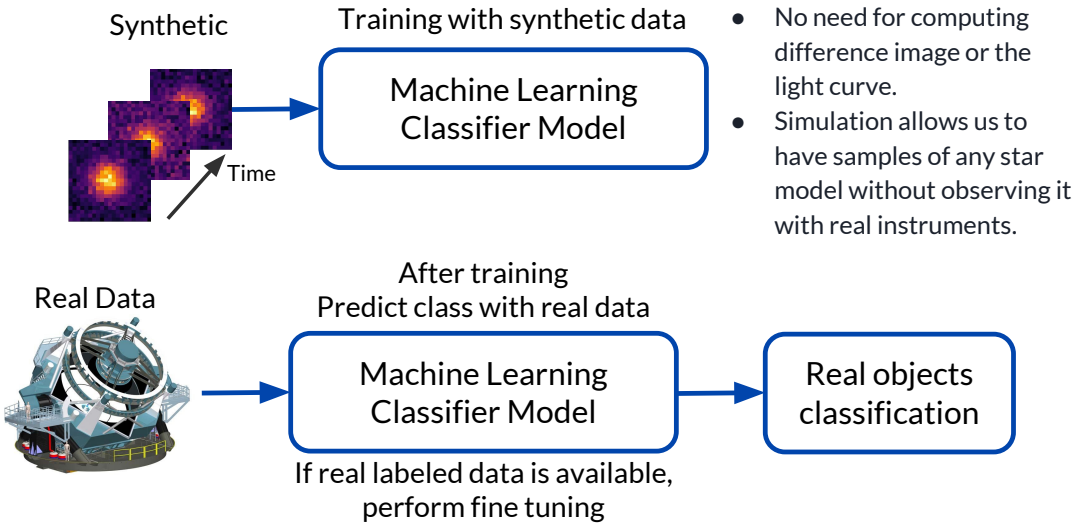


Figure 3.1: Main steps of the proposed methodology. First, simulate data using realistic consideration about observation conditions and camera parameters. Train the RCNN model to solve the classification problem in the simulated dataset. If real data is available, fine tune the model using a few real samples and classify the rest of real samples.

Summarizing, in a real case scenario, the proposed methodology for classification of astronomical objects coming from an alert stream is the following: Collect survey information

and simulate the images, train the RCNN model to solve the classification problem on the trained dataset, then when real data is available, fine tune the model with real examples to adjust the model from simulated domain to real domain. The methodology is summarized in Figure 3.1.

3.1 Data Simulation

The simulated dataset of labeled image sequences was created using the following procedure. First, all the information required to mimic realistic observing conditions which corresponds to instrument specifications, observation dates, exposure times and atmospheric conditions was gathered. These parameters are specified in Section 3.1.1. In this work, observing conditions for the HiTS survey 2015 on band g [9, 47, 53] were simulated. Next, light curves were simulated based on physical and empirical models resulting in a continuous function of the magnitude as a function of time, then light curves are built by sampling this function using the observation dates. The instrument specifications, exposure times and atmospheric conditions are used to generate an image for each point of a light curve, and finally noise is added to each image. In this way, the result of the procedure is an irregularly sampled movie of 21x21 pixels for each astronomical object.

3.1.1 Synthetic data simulation parameters

The simulation process is adjustable to different observing conditions, so the proposed model can be trained and applied to different instruments. To apply the RCNN classifier to a different survey, it is required to gather representative parameters, simulate images and train a new model on the simulated dataset. Ideally the distribution of the simulated data should be equal to the distribution of the real data. In practice, this is not always possible, so a domain adaptation problem arises where the features and tasks are the same but the distributions are different. The aim is to build a simulated dataset whose distribution is close enough to the real one, so that during a fine tuning phase with a few real samples could match the distributions of the simulated and real training set used to train the model and solve the domain adaptation problem.

Table 3.1: Image Simulation Parameters: Camera parameters are constant for a given instrument, but exposure parameters vary in time.

Camera parameters	
Gain [e-/ADU]	Read Noise [e-]
Saturation [ADU]	Pixel Scale [arcsec/pixel]
Exposure parameters	
Date [MJD]	Seeing [pixels]
Airmass	Sky brightness [ADU]
Zero Point [mag]	Filter [g, r, i or z]
Exposure Time [sec]	Limiting magnitude [mag]

The observing condition parameters are composed of camera and exposure parameters. A camera has a unique list of parameters describing the conversion from photons to digital units. An exposure has a unique list of parameters which describe the time and duration of the exposure, as well as the relevant atmospheric conditions during exposure. These parameters are summarized in Table 3.1. The parameter values used in this work were taken from the HiTS survey, which consists of 50 fields observed by DECam [62] with 62 CCD cameras per field, and between 25 and 30 observations per field. In this work, empirical observing conditions were sampled from real observations from HiTS. The typical observing conditions are described in [9].

The simulated images are produced assuming a given point spread function (PSF), which is sampled from a collection of empirical PSFs, an efficiency of conversion from physical units to analog digital units given by the camera and exposure parameters, and the sky level given in the exposure parameters. More details are given in Section 3.1.3.

3.1.2 Light curve simulation

Seven classes of astronomical objects were simulated (see Table 3.2): two non–variable (non–variable stars and galaxies) and five variable or transient (RR Lyrae, Cepheids, eclipsing binaries, supernovae and asteroids). Variable sources are simulated in two steps: 1) sampling from either a physical model or empirical data, and 2) adjusting their brightness by sampling from certain magnitude distributions as explained below. In order to sample each type of light curve for a given observation date, different interpolation methods were used as shown in Table 3.2.

[1] `https://www.lsst.org/scientists/simulations/catsim`

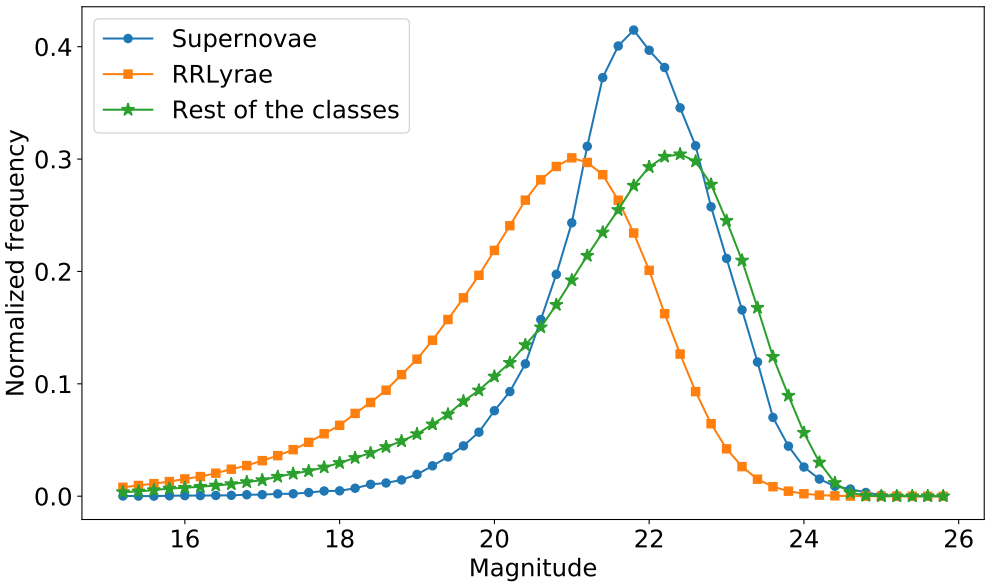


Figure 3.2: Magnitude density distribution of the simulated data.

Table 3.2: Class description, astronomical sources simulated in this work.

Astronomical Object	**Generation model**
Supernovae	Simulations based on physical models of SNe II from [59] and SNe Ia spectrophotometric templates from [58]
RR Lyrae	483 light curve templates, sampling a random phase and average magnitude [100]
Cepheids	600 real cepheids light curve [101] fitted with a Gaussian process for interpolation [102]
Eclipsing Binaries	375 Eclipsing binaries templates from CatSim[1], part of the LSST simulation tools
Non-Variable objets	Constant brightness value for each time of observation
Galaxies	Exponential and De Vaucouleur's luminosity profile using parameters from SDSS galaxy catalog [103]
Asteroids	Simulated as a bright source in just a single time of observation

The process starts by sampling a light curve either from a physical model or from empirical data. Supernova redshifts and light curves are obtained from simulations which take into account cosmology and supernova rates, the telescope parameters, and physical models for SNe II from [59] and spectrophotometric templates for SNe Ia from [58]. For both type of supernovae, the explosion occurs within the time span of the simulated survey, and at least 1 point of the light curve must lie above the limiting magnitude in order to have a detection. RRlyrae light curves were sampled from Gatspy package templates [100]. Cepheids were sampled using real light curves from M33 CFHT variability survey [101], by folding the real light curves using the reported period, and performing a Gaussian Process regression [102] using a periodic kernel for interpolation of the survey cadence. Eclipsing binaries were sampled using templates from the LSST Catalog Simulation database (`https://www.lsst.org/scientists/simulations/catsim`, CatSim). In the case of eclipsing binaries, it was imposed that the eclipse of the light curve must be detected with 5 sigma above noise between the constant regime and the eclipse. Light curves for non–variable objects were simply simulated as a constant light curve, and asteroids as a single peak. Galaxy simulations are explained in Section 3.1.3.

Light curves were sampled using the empirical exposure parameters from HiTS and scaling them to follow a magnitude distribution that reproduces the HiTS observations. Magnitudes for non–variable sources, eclipsing binaries, Cepheids, and asteroids are sampled from the green curve (rest of the classes) in Figure 3.2. This distribution was obtained by fitting an exponential function to the distribution of stars in HiTS. A constraint was added to smooth the decay at large magnitudes in order to follow the supernovae magnitude distribution. This was performed by multiplying the exponential density distribution by a cutoff function $f(m, m_{\text{cutoff}}) = 1 - \text{erf}(m - m_{\text{cutoff}})/2$, where erf is the error function and m_{cutoff} is the value where $f(m, m_{\text{cutoff}}) = 0.5$. The supernovae magnitude distribution decay at large magnitudes can be mimicked by using $m_{\text{cutoff}} = 22.8$ for all the classes, except for RR Lyrae and galaxies. For RR Lyrae, $m_{\text{cutoff}} = 21.5$ was chosen to make the distribution with magnitude boundaries based on [104].

At this point, simulated light curves have no noise. Errors of flux in real light curves are estimated using statistical assumptions about measurement noise within the image, so the noisy versions of simulated light curves are recovered from noisy simulated images in later steps. Fig. 3.3 shows examples of light curves for each category (except for the galaxy class).

3.1.3 Image simulation using light curves

Having sampled simulated light curves with the corresponding cadence, the zero point values $Z_p(t)$ were used to convert each point of the light curve from magnitudes to ADU using:

$$m(t) = Z_p(t) - 2.5 \log \left(\frac{\text{ADUs}(t)}{T(t)} \right), \tag{3.1}$$

which is the same expression as equation 2.1, where in this case t is the observation time and $T(t)$ is the exposure time for an image at time t. Usually, there are other terms in this conversion associated with airmass and color, but these Z_p's were computed using PanSTARRS1 [105] to fit the resulting magnitude of known sources. For each light curve, a random CCD array was chosen and its exposure parameters at different epochs were used. Then, for each point in ADU units of the light curve, a PSF $p_t(x, y)$ is used to generate a source image, where x, y are pixel coordinates and $\sum_{x,y} p_t(x, y) = 1$. The source image (see the example shown in Figure 3.6) was generated by creating an empty image of 21×21 pixels and adding $\text{ADUs}(t) \cdot p_t(x - x_0, y - y_0)$ where x_0, y_0 is the center of the source in the image, sampled from a uniform distribution within the single image central pixel to simulate random centering errors. The PSF p_t is estimated by averaging real source images from the HiTS survey and computing its FWHM by fitting a 2D-Gaussian function as an estimation of size. The PSF estimations have different sizes, but for each observation time t, a single p_t was matched

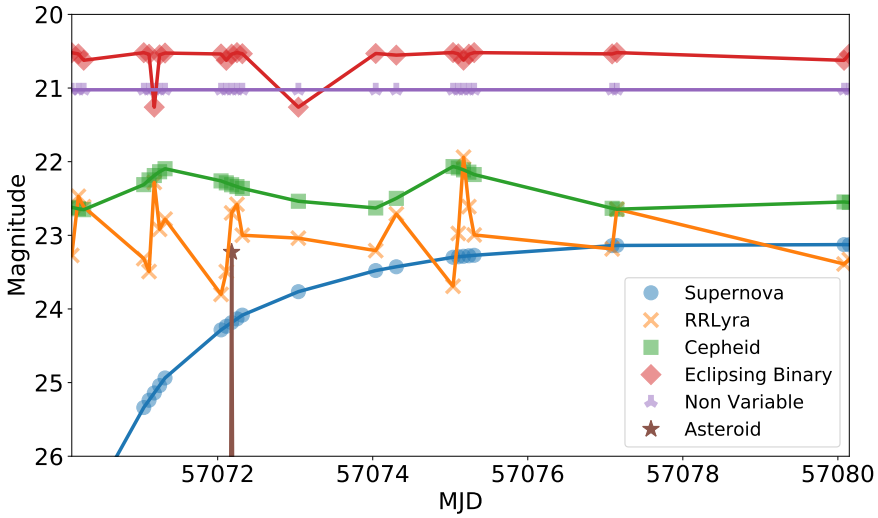


Figure 3.3: Light curve examples for six classes of astronomical objects in the dataset. Galaxies are simulated by using the method described in Section 3.1.3.

with the current FWHM(t). Dates with FWHMs larger than 2" were not used. A random rotation and mirroring is applied to make the classifier invariant to rotations of the PSF.

Some sources may have a host galaxy, which were simulated by using Sérsic profiles [106], described by

$$I(r) = I_0 \exp\left\{-b_n\left[\left(\frac{r}{R_e}\right)^{1/n} - 1\right]\right\}, \tag{3.2}$$

where $I(r)$ is the intensity as a function of the radius r, I_0 is the intesity at the effective radius R_e, n is called the Sérsic index and defines the shape of the profile, and b_n is a constant defined so that R_e is the half-light radius. [107]. For $n = 4$, $b_4 = 7.669$, the function is called De Vaucouleurs profile used to model the bulge of the galaxy. For $n = 1$, $b_1 = 1.678$ the function is called exponential profile used to model the disk of the galaxy. The complete profile was obtained by a linear combination of both profiles described. The parameters for exponential and De Vaucouleurs profiles were obtained from the Sloan Digital Sky Survey [103], including the following (if many bands are used, these quantities are per band): radii, ellipticities, proportion between the two profiles, and the luminosity of the galaxy in magnitudes. Examples of simulated profiles for different bands are shown in Figure 3.4. For each of the supernovae, the redshift was matched with the corresponding one of the sampled galaxy, in order to be consistent with shapes and the magnitude of the galaxy due to its distance. In order to simulate a SN in a host galaxy, the position of the star was sampled from a distribution following the exponential profile in the g band, as shown in Figure 3.4. The magnitudes were converted to ADUs and distributed using the exponential and De Vaucouleurs profiles. These profiles have a spike at the center, concentrating most of the flux in the central pixel. In order to avoid this issue, 20 uniform random positions in the range of the central pixel were sampled, computed the bulge profile and averaged them to distribute the flux across the image. We finally convolved this image with $p_t(x, y)$ generating a galaxy image $\mathrm{IM}_{\mathrm{gal}}$. Examples of the resulting galaxy image per band (without noise) is depicted in Figure 3.5.

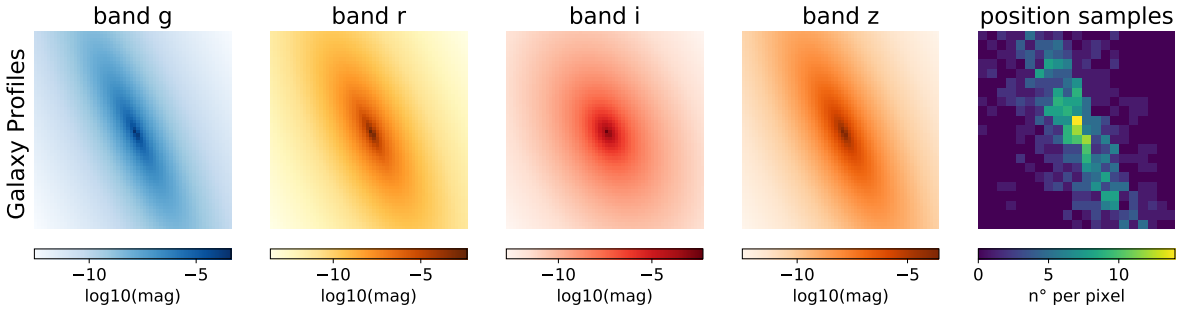


Figure 3.4: Simulated galaxy profile for bands g, r, i and z. The magnitude per band was spread within the image according to the profile (plotted in log scale for visualization purposes). The last image at the right shows an histogram of possible positions for the transient within the galaxy, which follows the disk profile in g band.

The last step for image simulation is to produce a joint image by adding up the PSF-like image, the galaxy, and the sky brightness Sky(t) for time t. Then, we convert ADU pixels to electrons e− multiplying by the corresponding Gain of the camera, in order to apply independent Poisson noise to each pixel and Gaussian readout noise. Finally, the

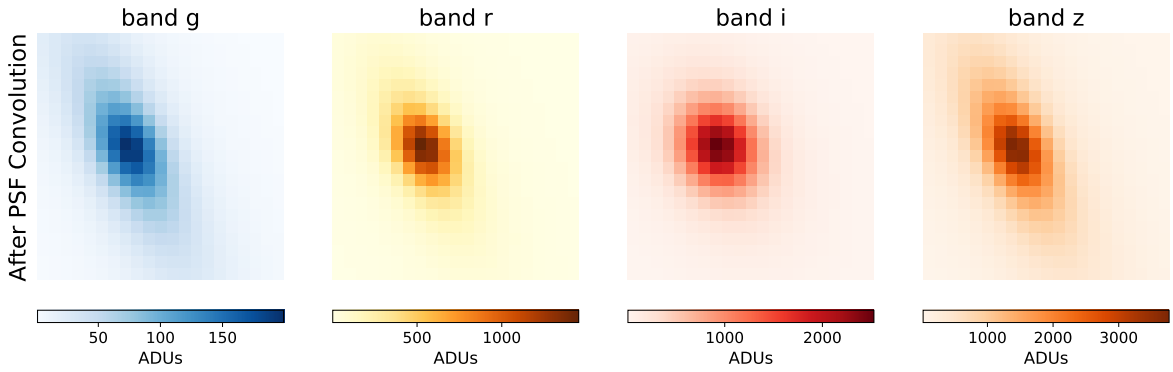


Figure 3.5: Simulated galaxy images. After the random averaging of the bulge to distribute the luminosity correctly, the magnitude per pixel was converted to ADU units using equation 3.1. Then convolved with the corresponding PSF p_t. The location of the source is sampled from the disk of the galaxy, then the galaxy image is displaced in order to put the location of the source in the center of the image.

image is converted back to ADUs. The resulting image sequence for each object is unevenly sampled. Also, the noise on each stamp is variable and consistent with a realistic observation since it depends on the observation conditions such as the PSF size, sky brightness, zero point estimation, presence of a galaxy, along with the CCD readout noise. An example of a simulated SN with a host galaxy image is shown in Figure 3.6.

Host galaxies were added on to 50% of the supernovae objects and 5% of the rest of the classes. This proportions are due to the fact that the supernovae positions tend to follow the host galaxy light as opposed to the other variable classes used in this work.As mentioned before, the galaxy class is a simulated image of a host galaxy with varying exposure parameters. Extra structures like galaxies were added to prove that the proposed model is able to learn extra information aside from the source for the classification task.

In Figs. 3.7 to 3.9, examples of simulated images compared to the real ones are shown. Given a real non–variable source with a known magnitude, the same observation conditions where the real source was observed were used to simulate a non–variable source. Sample time goes from left to right, top to bottom. Since the estimated point spread function used for simulations is computed by averaging single PSFs, simulated images have bigger PSFs than real ones because of the blurring effect when averaging. As can be observed, the pixel distribution on the simulated images is very close to the real ones, including the brightness of the source and background levels.

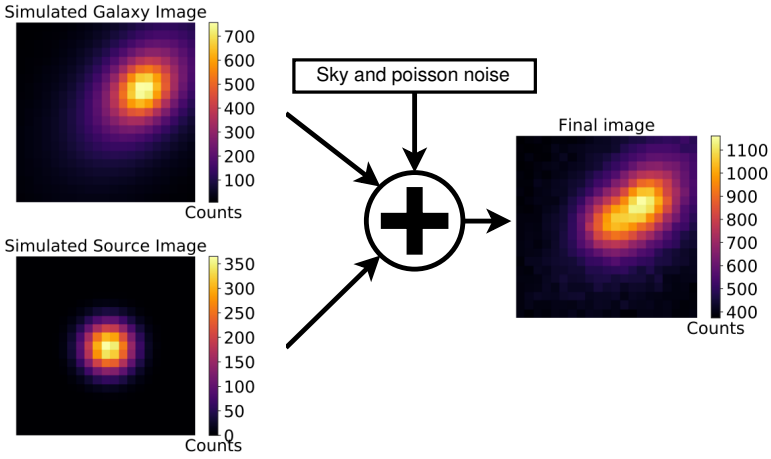


Figure 3.6: Summary of the image simulation process. The light coming from a source is spread in the Source Image. A simulated host galaxy is added to the source image. The sky brightness is added as a constant value to all the pixels and Poisson noise is sampled with variance equal to the number of photons in each pixel along with CCD readout noise.

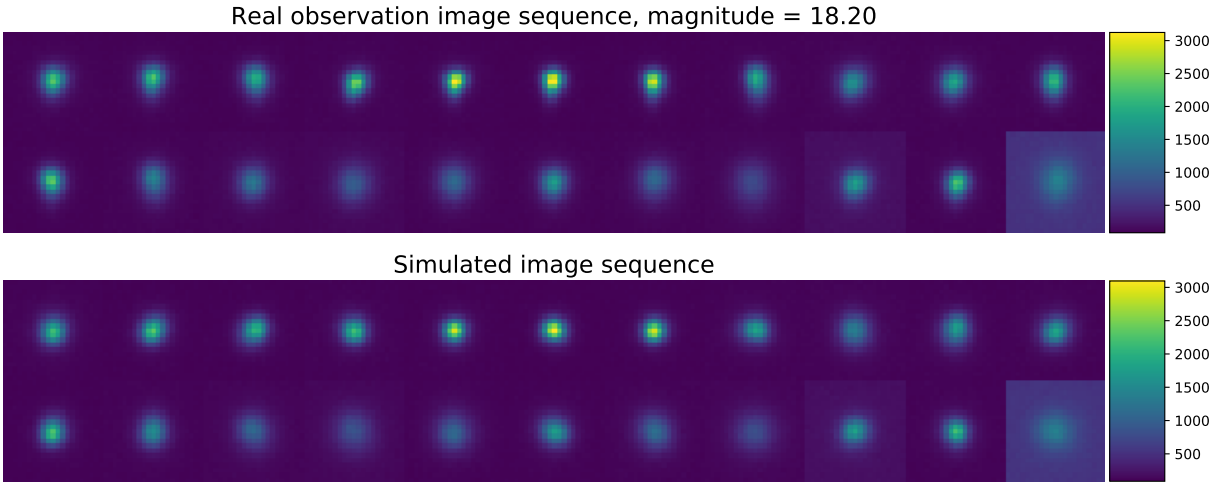


Figure 3.7: Image simulation example 1

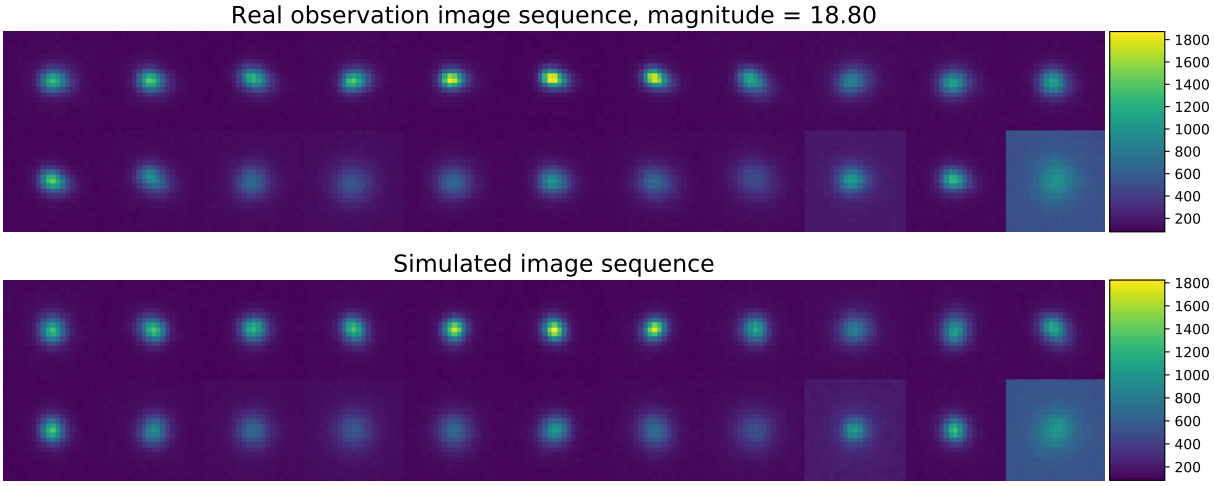


Figure 3.8: Image simulation example 2

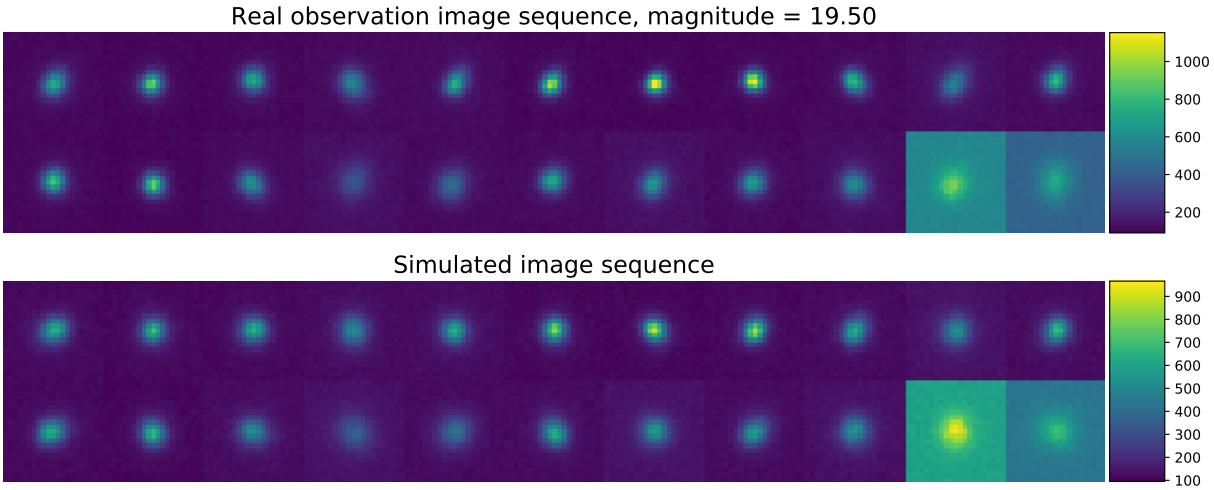


Figure 3.9: Image simulation example 3

3.2 Classification Models

In Section 3.2.1 the input of the proposed RCNN model based on the science image is explained. Then, the proposed model architecture is shown in Section 3.2.2, along with the light curve classifier based on features from light curves in Section 3.2.3. Finally, in Section 3.2.4 the training processes for both classifiers are detailed.

3.2.1 Sequence input to the model

Before inputting an image to the classifier a pre-processing step is performed, that consists in subtracting the sky in counts to the image as a constant value. Then, each pixel is multiplied by a factor γ that ensures the same number of counts for a given magnitude. This factor γ can be derived by equating the ADU units for the same magnitude m which is the inverse function in equation 3.1, but with different zero point as follows:

$$10^{\frac{Z_p^{\text{ref}}-m}{2.5}} \cdot T^{\text{ref}} = 10^{\frac{Z_p(t)-m}{2.5}} \cdot T(t) \cdot \gamma \tag{3.3}$$

$$\gamma = 10^{\frac{Z_p^{\text{ref}}-Z_p(t)}{2.5}}, \tag{3.4}$$

where Z_p^{ref} and T^{ref} are the zero point and the exposure time at the reference time which is taken as the first exposure for each field, $Z_p(t)$ and $T(t)$ are the zero point and the exposure time at the current exposure. Using the same exposure time for the entire survey, the resulting zero point correction factor γ is the one shown in equation 3.4. Then, the pre-processing operation is:

$$\text{IM}_{\text{proc}}(x, y) = (\text{IM}_{\text{orig}}(x, y) - \text{sky}) \cdot \gamma, \tag{3.5}$$

where $\text{IM}_{\text{proc}}(x, y)$ is the resulting preprocessed image, $\text{IM}_{\text{orig}}(x, y)$ is the original image. $\text{IM}_{\text{proc}}(x, y)$ was used to build the inputs to the model.

Objects where the source is detectable in at least one of the images within a sequence i.e., at least one point of the original light curve in magnitude must be above the limiting magnitude, are candidates to supernovae or asteroids, the first point when this happens is the "first alert" triggered by the rise of flux in time.

Because we are interested in classifying supernovae in the early stages of their explosion, once an alert is triggered (supernova explosion or asteroid appearance) at time t_i, the five images previous to the alert are queried and form a stack of n_w consecutive images using t_{i-5} as the first image of the stack. In this work $n_w = 3$ is used, so if the alert occurs at time t_i, then the input images for the first stack are at time $(t_{i-5}, t_{i-4}, t_{i-3})$, then next time step input will be at $(t_{i-4}, t_{i-3}, t_{i-2})$, then $(t_{i-3}, t_{i-2}, t_{i-1})$ and so on. Therefore the input to the model is a stack of n_w consecutive images forming an input tensor of shape $(21 \times 21 \times n_w)$. Figure 3.10 shows a scheme that represent the inputs for the proposed classifier.

N_{d} is defined as the number of available dates, which corresponds to the number of points between five images before the first detection and last exposure on the respective field. N_{d} depends on the first detection date for supernovae or asteroids. The same N_{d} dates were

used to build the input sequence of the remainder of classes (RR Lyrae, Cepheids, eclipsing binaries, non variables and galaxies), in order to have variable size and observation conditions for every class in the dataset. The maximum number of available dates was truncated to $N_{\mathrm{d}} = 22$ for every object to evaluate the model.

A $n_w > 1$ number of stacked images at the input was used, and not just a single image because convolutional layers can learn short timescale dependencies between these n_w consecutive images, letting the recurrent layer learn about longer timescale dependencies. The difference in sampling time between the first image of the sequence and the rest of the images was fed to the model to consider the irregular sampling for each image.

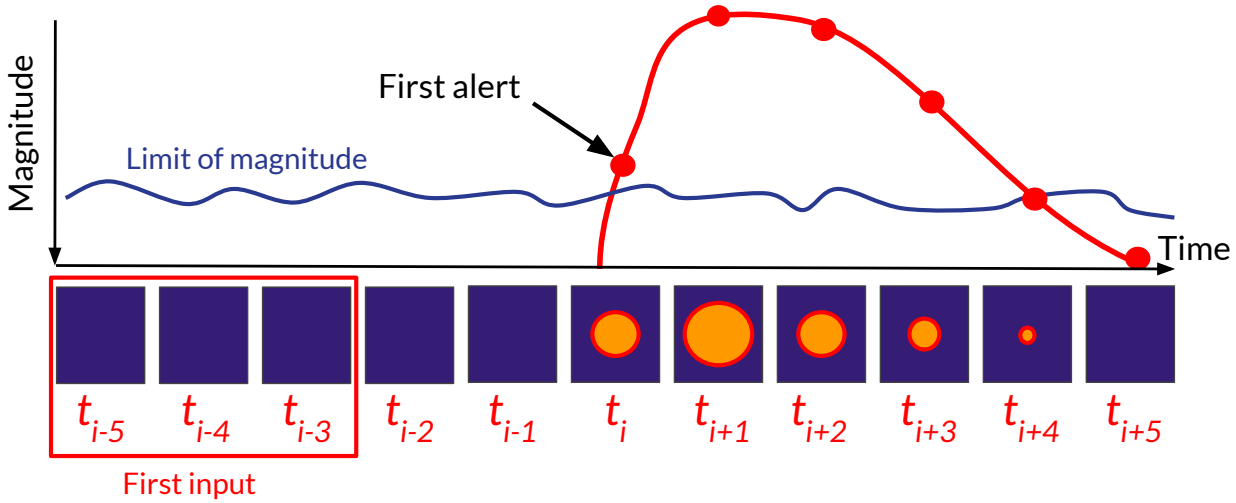


Figure 3.10: Input sequence for RCNN classifier. The red curve is a supernova light curve representation, where red circles are the points observed. The first point observed above the limiting magnitude is the first alert or detection, with time defined as t_i. The last 5 images are retrieved and starting from t_{i-5} the first input tensor is built (red box at left). The next input for each time step is created by moving the red box one image per time step until no image is available. Because of this construction, the first alert will be always at the fourth of n_w images input, and the sixth image of the sequence would be fed to the model.

3.2.2 Image sequence classifier architecture

A new model is proposed to classify astronomical objects based on a recurrent convolutional neural network (RCNN), which uses sequence of images as input. Convolutional layers are able to automatically learn the spatial correlation between pixels in the input image and extract high-level features, which are used by the recurrent layer to learn time dependencies among images sampled at irregular times. The RCNN model uses high-level representations of the image obtained by convolutional layers as inputs to the recurrent layer. In this way, we can add information to the memory of the classifier while the images are received by changing one image at a time. The LSTM units in the recurrent layer contain memory cells that store learned knowledge from past input images.

As mentioned above, the input is a 3D tensor of size $(21, 21, n_w)$, made by n_w consecutive

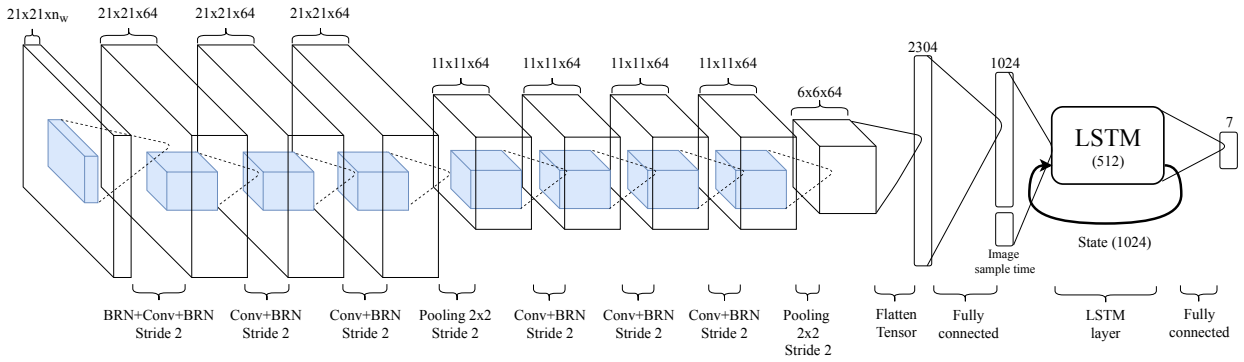


Figure 3.11: RCNN architecture. An input tensor with shape $(21, 21, n_w)$ is shown at the left of the image. Every layer is described at the bottom and the shape of the data is indicated at the top.

21×21 images. First, a batch renormalization layer is applied, followed by a convolutional layer to increase the number of channels from n_w to 64 and an extra batch renormalization layer. 64 filters and a stride of 1 were used on each convolutional layer with ReLU as activation function. Also at the output of each convolutional layer a batch renormalization layer is implemented for each channel, then a pool layer after the first three convolutional layer + batch renormalization, followed by three convolutional layers + batch renormalization and a final pool layer.

The output of the pool layer is flattened to a vector of size $6 \times 6 \times 64 = 2304$, which is the input to the first fully connected layer with 1024 hidden units. The time difference between the date of observation of the n_w images and the first image of the entire sequence (n_w size vector) is added to the input of the LSTM layer which has 512 units. The initial state of the LSTM is an array filled with zeros and the state is updated for every input tensor with n_w stacked images. Finally, the LSTM output is passed through a fully connected layer with softmax activation functions. The details of each layer are shown in Table 3.3. Fig. 3.11 shows an illustration of the RCNN architecture.

3.2.3 Light curve random forest classifier

For comparison purposes, a light curve classifier using feature extraction and a random forest (RF) classifier [70] was designed. The light curves in ADUs were extracted from the same set of simulated stamp images given to the RCNN using optimal photometry [8]. All the features available in FATS [13] were computed for each light curve, except for the features associated with color, *Color, Eta_color, Q31_color, StetsonJ, StetsonL* since the HiTS survey uses mostly the g band. In order to evaluate the accuracy of the light curve RF classifier on the real dataset as a function of time, FATS features were recomputed for each new point added to the light curve. FATS includes the estimation of the period of the light curve and other features from Lomb Scargle Periodogram [66], which are known to be important for variable star classification. The light curves are almost perfectly derived from the images, since the same information used to simulate them is used in the light curve extraction such as the PSF, presence of a galaxy and the background value to extract the light curve correctly. In

Table 3.3: Recurrent Convolutional Neural Network architecture.

Layer	**Layer Parameters**	**Output Dim**
Input Layer	$21 \times 21 \times n_w^{\mathrm{a}}$	$21 \times 21 \times n_w$
BRN	n_w (mean and std)	$21 \times 21 \times n_w$
Conv + BRN	$3 \times 3 \times 64$, 64	$21 \times 21 \times 64$
Conv + BRN	$3 \times 3 \times 64$, 64	$21 \times 21 \times 64$
Conv + BRN	$3 \times 3 \times 64$, 64	$21 \times 21 \times 64$
Max pooling	2×2, stride 2	$11 \times 11 \times 64$
Conv + BRN	$3 \times 3 \times 64$, 64	$11 \times 11 \times 64$
Conv + BRN	$3 \times 3 \times 64$, 64	$11 \times 11 \times 64$
Conv + BRN	$3 \times 3 \times 64$, 64	$11 \times 11 \times 64$
Max pooling	2×2, stride 2	$6 \times 6 \times 64$
Fully connected (with dropout)	2304×1024	1024
LSTM	$1024 + \Delta t$ of samples 512 units	512
Output softmax	512×7	7 (n° classes)

[a] n_w is the number of images stacked in the input tensor, BRN stands for Batch Renormalization

practice, these parameters should be estimated from the images. Finally, a RF classifier is trained to discriminate among the seven classes. Feature based random forest classifiers are commonly used in astronomy [108, 109, 9, 53].

3.2.4 Training Process

Recurrent neural networks are trained using variants of gradient descent, by using backpropagation through time. Since features are extracted from the images using convolutional layers and are fed to the recurrent layer, the gradients through time are also used to adjust the parameters of the convolutional layers. For the proposed RCNN model, cross-entropy was used as loss function to compare the outputs of the model with the labels. The total loss of a single example is loss $= \sum_t^{N_{\mathrm{d}}} \mathrm{loss}(t)$ where loss(t) is the cross-entropy at time step t. The training algorithm is AMSGrad [82] which is an adaptive learning rate algorithm. The batch size was 256 sequences of simulated images and the training was done by running 30,000 iterations presenting a single batch per iteration to the image sequence classifier, using $5 \cdot 10^{-4}$ as learning rate. The final model was chosen by selecting the one that had the lower loss in the validation set during the training process. Graphic processor units (GPUs) were used to train the models. Each model takes approximately 8 hours to complete 30,000 iterations, equivalent to 9.5 epochs, in a GeForce GTX 1080 Ti. For the fine-tuning phase, after training RCNN with the simulated dataset, 1000 extra iterations with 10 image sequences per class randomly selected from the real dataset were used. The code was implemented on Tensorflow 1.7 [110]

For the light curve classifier, a RF was trained using the light curves extracted from the simulated images, on which 58 FATS features were computed, also running a grid search for the best max depth of each decision tree and the number of them. The best results in the validation set of the simulated dataset were obtained with 25 estimators and a depth of 15. In order to make a fair comparison with the fine tuned version of the RCNN, the RF

model was trained by adding 10 samples of light curves per class from the real dataset to the simulated training set, resulting in an augmented training set. Each real sample added to the augmented training set was copied 100 times on the training set, which is comparable to the number of times that the RCNN observed every additional real example during the 1000 extra iterations.

3.3 Methodology summary

Figure 3.12 shows a step by step summary of the methodology proposed in this work. Starting from observation conditions and camera parameters of a survey, light curves are simulated, and then images are simulated. Simulated images are used as input directly for the proposed RCNN classifier, which is trained on the simulated dataset, and then fine tuned with real data. Noisy versions of the light curve are recovered from the images, then FATS features are computed and used to train a random forest classifier. Finally, real samples are added to the training set of the random forest to simulate a fine tuning of the light curve model.

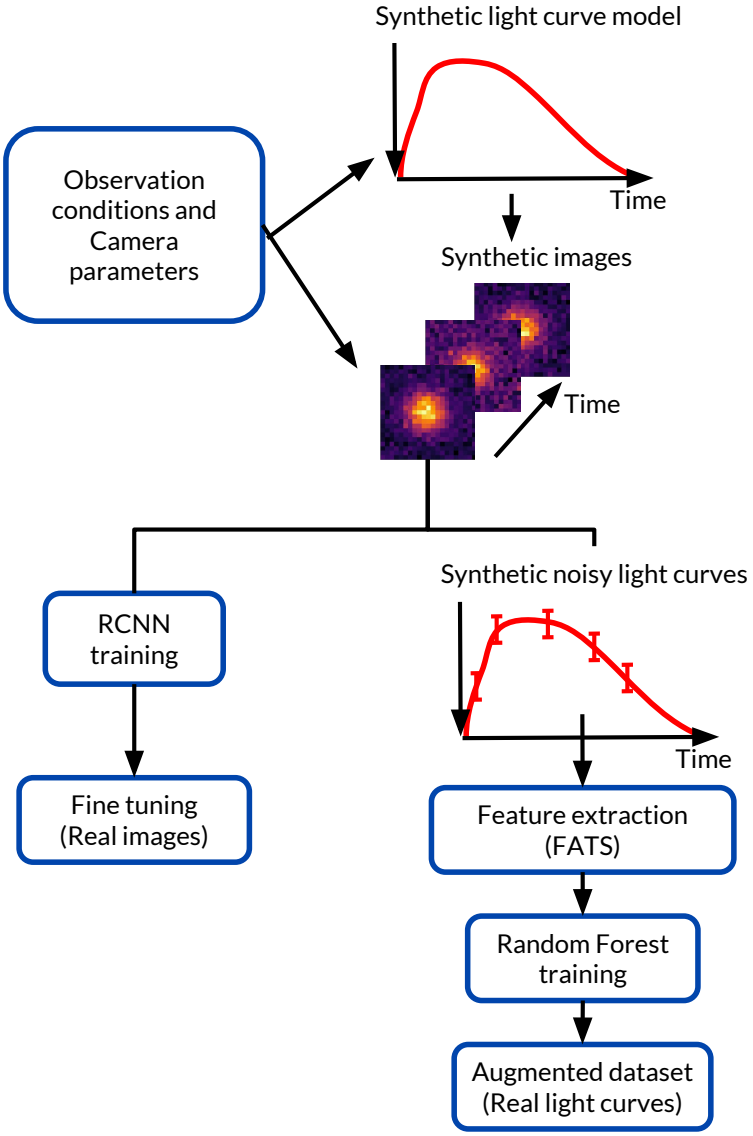


Figure 3.12: Flow chart of the methodology. The first step is the light curve simulation, then image simulation using the light curves. The second step is the RCNN training using simulated images and fine tuned with real data. Third, light curves are extracted from simulated images, the features are computed to train a random forest classifier to finally add real examples to the RF training set.

Chapter 4

Results and Analysis

In this chapter, the results of experiments proposed in chapter 3 are detailed. In Section 4.1 the main results for the proposed RCNN classifier are shown and compared against the random forest light curve classifier. Both classifiers were tested on simulated and real data. Next, the results are discussed in Section 4.2, identifying the most important advantages of the proposed model, also showing particular evaluation cases in Section 4.3. Finally, assumptions and difficulties about the methodology are visited, proposing future experiments to find improvements for some of them in Section 4.4.

4.1 Results

After completing the training process on the simulated dataset, the image RCNN and the light curve classifier were evaluated both on the simulated dataset and on the real image dataset. The RCNN was evaluated both with and without fine-tuning. The light curve RF classifier performance is evaluated both using the simulated training set and the augmented training set as explained in Section 3.2.4. In the case of real supernovae and asteroids, the first detection is defined as the first point where the number of counts on the estimated light curve is five times higher than the counts error, then the input to the models is built in the way described in Section 3.2.1. For the rest of the classes, the sequence of real data starts at the first exposure. Figure 4.1 shows a comparison between the image sequence classifier (RCNN) and the light curve RF classifier, in terms of the accuracy evolution for simulated data as a function of the number of samples, images in case of the RCNN and light curve points in case of the light curve RF classifier. Standard deviations were estimated by training 5 different models. Figure 4.2 shows the average recall over all classes when the models are applied to real images, weighting every class equally. The curves in Figure 4.2 show the average recall for real data using the models trained over the simulated dataset, as well as the RCNN with fine tuning and the RF classifier trained with the augmented training set, where the test set does not include samples used to fine tune the models. The errors in Figure 4.2 were computed by taking 5 trials of 10 samples per class randomly selected to fine tune the model. Figure 4.3 shows the accuracy as a function of the magnitude of the simulated

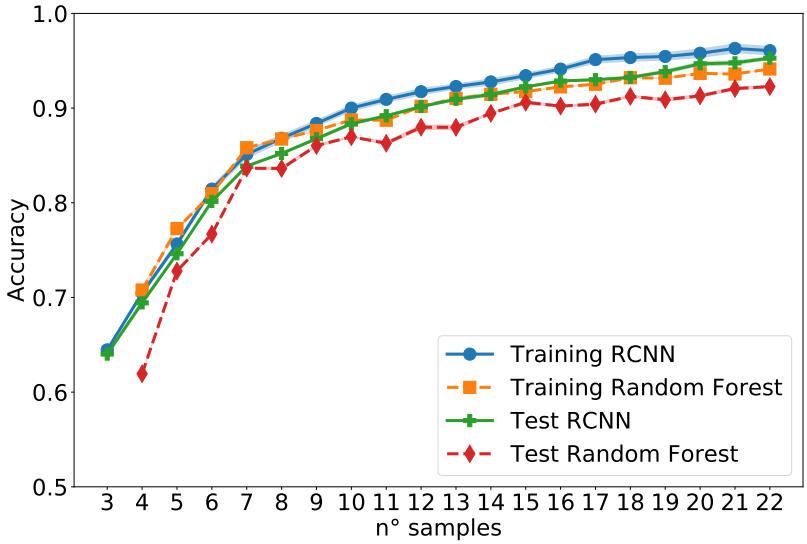


Figure 4.1: Accuracy on the simulated dataset as a function of the number of samples of the sequence available to the classifiers. The accuracy obtained at the last point (after 22 samples) for each curve is: Training RCNN: 0.961 ± 0.005, Training Random Forest: 0.941 ± 0.001, Test RCNN: 0.953 ± 0.002 and Test Random Forest: 0.923 ± 0.002.

objects for both models, and the recall for real objects after fine tuning separated for each class, with standard deviations estimated using 5 different fine tuned models.

Figures 4.4a and 4.4b show confusion matrices for the RCNN and RF classifiers, respectively, on the simulated dataset after using all the points on each sequence to classify. Figures 4.5a and 4.5b show confusion matrices on the real dataset for both models when trained on simulated data only. Figures 4.6a and 4.6b show confusion matrices for both models, after fine tuning the RCNN model and training the light curve RF classifier with the augmented training set.

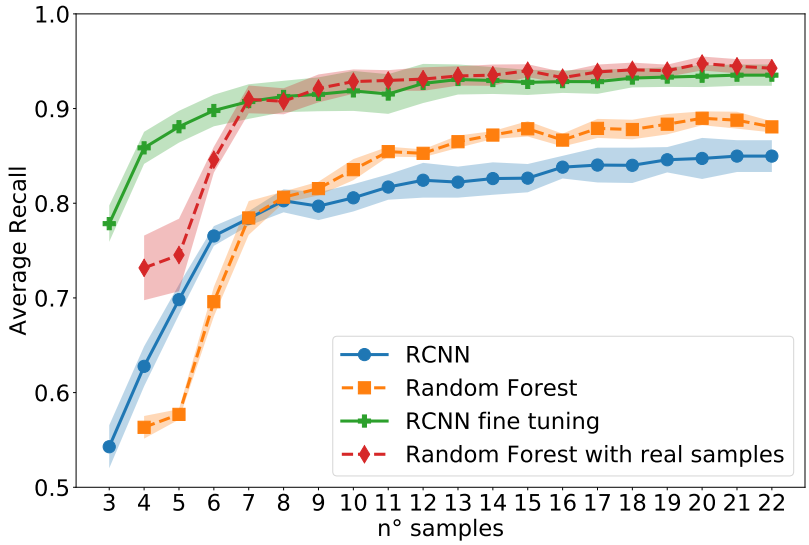


Figure 4.2: Average recall on the real dataset without and with fine tuning as a function of the number of samples of the sequence available to the classifiers. The average recall obtained at the last point (after 22 samples) for each curve is: RCNN: 0.85 $\pm$ 0.02, Random Forest: 0.88 $\pm$ 0.01, RCNN with fine tuning: 0.94 $\pm$ 0.01 and Random Forest trained with augmented training set: 0.94 $\pm$ 0.01. For the last two cases, the test set does not include samples used to fine tune the model.

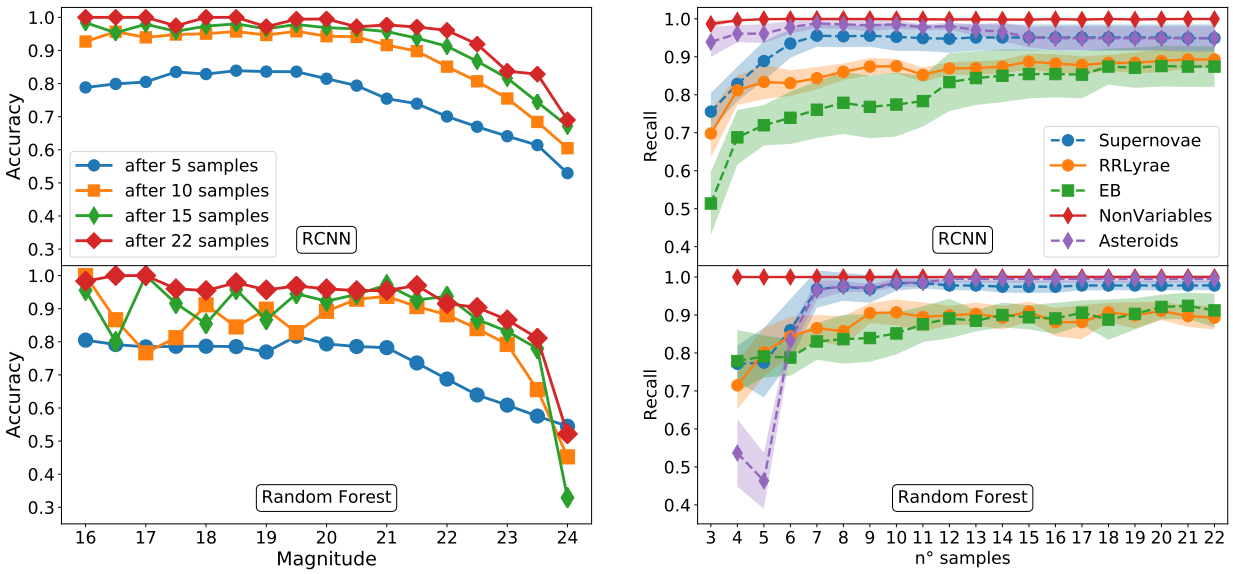
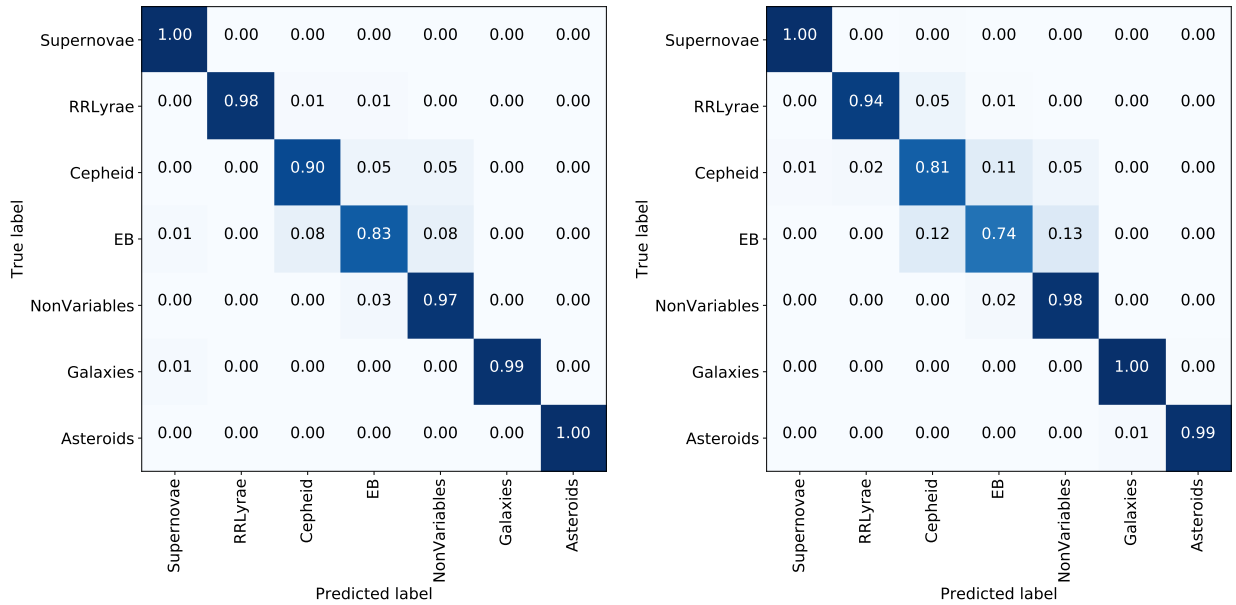


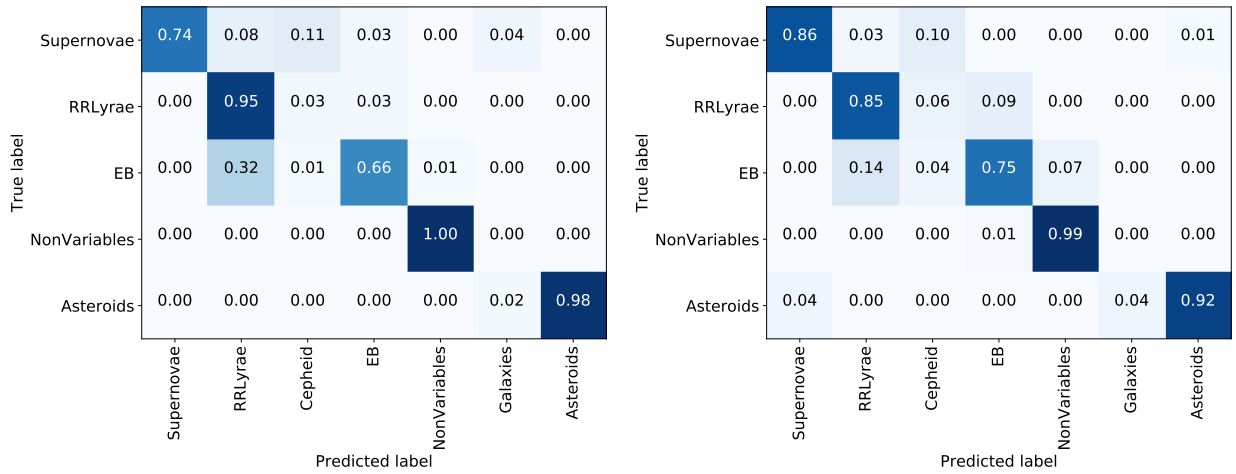
Figure 4.3: Results comparison between the image sequence RCNN classifier (top row) and light curve RF classifier (bottom row). The left plot shows the accuracy on the simulated dataset as a function of the object magnitude and the number of images. The right plot shows the recall for each class available on the real dataset as a function of the number of samples presented to the classifier.



(a) Proposed RCNN classifier

(b) RF light curve classifier.

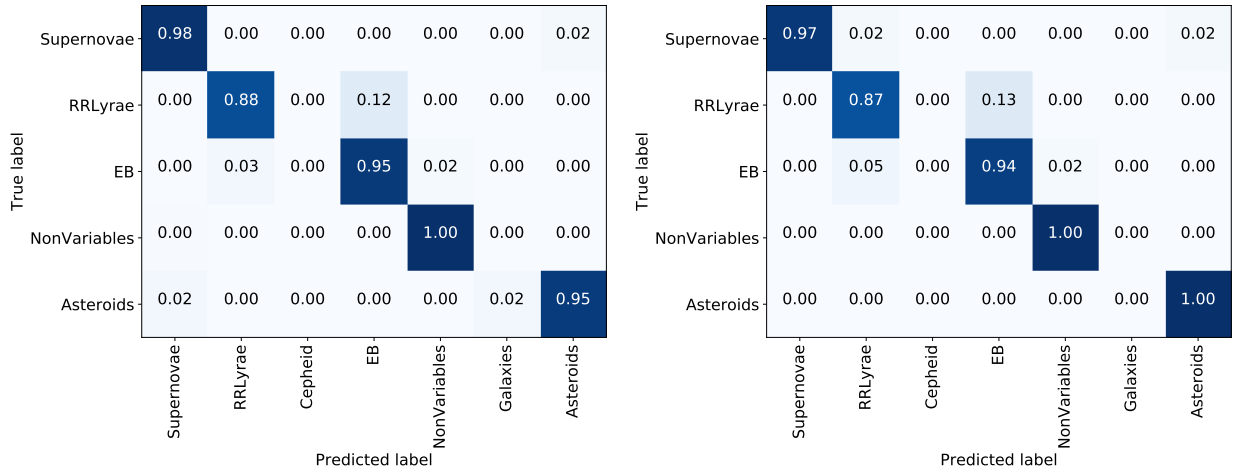
Figure 4.4: (a) Confusion matrix on simulated test set obtained with the image sequence RCNN classifier using all samples available on each sequence to feed the neural network. (b) Confusion matrix on simulated test set obtained with the light curve RF classifier using all the points available on each light curve.



(a) Proposed RCNN classifier

(b) RF light curve classifier.

Figure 4.5: (a) Confusion matrix on the HiTS real dataset obtained with the image sequence RCNN classifier without fine tuning. (b) Confusion matrix on the HiTS real dataset obtained with the light curve RF classifier trained on simulated data.



(a) Proposed RCNN classifier

(b) RF light curve classifier

Figure 4.6: (a) Confusion matrix on the HiTS real dataset obtained with the image sequence RCNN classifier with fine tuning. Real samples used to fine tune the model are not in this evaluation set. (b) Confusion matrix on the HiTS real dataset obtained with the light curve RF classifier trained with the augmented training set. Real samples used to augment the training set are not in this evaluation set.

4.2 Discussion

As can be observed in Figure 4.1, the RCNN achieved a higher accuracy on the simulated dataset as compared to the light curve RF classifier. The proposed RCNN model is particularly better on variable star classification (Cepheids, RR Lyrae and eclipsing binaries) as can be observed in the confusion matrices shown in Figures 4.4a and 4.4b. This result suggests that the image sequence RCNN classifier is able to retrieve the necessary information directly from images to solve the classification task. Since there is not much bias on each class apart from the light curve shape, my conjecture is that the model must learn to perform some form of photometry and extract the flux generated by the object on the image. On the other hand, Figure 4.2, and confusion matrices in Figures 4.5a and 4.5b show that the light curve RF classifier outperforms the RCNN classifier on the real dataset without fine tuning. This might be explained by a kind of "Domain Overfitting". If the RCNN were more overfitted to the domain of the simulated images than the light curve RF classifier, then its generalization capacity to the domain of real images would be diminished. In fact, the light curve RF classifier achieved a lower accuracy on the simulated dataset, likely producing less overfitting to it, but obtained a higher accuracy on the real dataset. There could be other factors as the models are not directly comparable.

The domain overfitting of the RCNN and RF can be fixed by adjusting the models using a few real labeled data. Figure 4.2, and confusion matrices in Figures 4.6a and 4.6b show that both models perform substantially better when a few real samples are used, achieving results comparable to those obtained with the simulated dataset. Notice in Figure 4.2 the improvement in accuracy at low number of samples, which is the moment previous to the first detection in the cases of asteroids and supernovae. From this, it can be inferred that the RCNN model is able to use additional information from the stamps than just the flux of the source, such as the presence of a host galaxy improving the initial guess between asteroid and supernovae for moments before the first detection This effect can be seen on the right plot of Figure 4.3 where asteroids are highly miss-classified by the light curve RF classifier until their first detection (sample number 6). In Section 4.3 some examples of the image sequence classifier applied to real supernovae are shown, where we can clearly observe this effect once again. This effect seems to be more important for fainter sources, which tend to be more distant and have smaller angular size galaxies, as can be observed on the left plot of Figure 4.3, where there is a clear improvement in accuracy for fainter sources when using the image sequence RCNN classifier in comparison to the light curve RF classifier.

It is worth to mention the fact that the simulations are good enough to classify correctly most of the real objects in the HiTS dataset without fine tuning. We can see in Figure 4.5a that real non–variable objects are perfectly classified using the image sequence classifier, as well as the majority of supernovae (74%), RR Lyrae (95%), eclipsing binaries (66%) and asteroids (98%), resulting in an average recall of 85% on the HiTS dataset. This means that the process of obtaining the right distribution that represents the physical effects of the light passing through the atmosphere, lenses and being captured by the CCD camera, by using the estimated exposure conditions, camera parameters and point spread function from empirical data is close enough to real conditions, making the simulation approach suitable to train a good model. The resulting model is used as a starting point to perform fine tuning with a

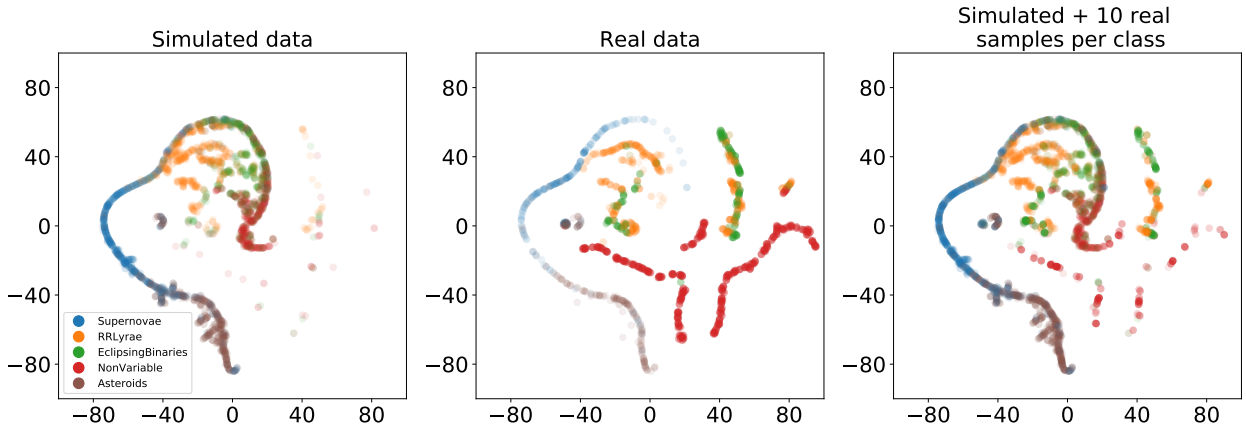


Figure 4.7: 2D projection of simulated and real light curves, represented by the 6 most important FATS features, according to random forests, using t-SNE. The six features are *MeanVariance, PercentDifferenceFluxPercentile, CAR_mean, PercentAmplitude, Mean, Max_slope*. 2000 light curves from the simulated dataset were used along with all the real light curves available, using a perplexity value of 40 and 1000 iterations.

few real samples, and get excellent results as shown in the confusion matrices in Figures 4.6a and 4.6b.

Figure 4.7 shows a 2D projection of both the simulated and real light curves, using a technique called t-Distributed Stochastic Neighbor Embedding (t-SNE [111]). Each light curve is represented by a vector composed of the 6 most relevant features, according to random forests, and then projected to two-dimensions. The left and center plots show that there is partial overlap between the simulated and real light curves, particularly on supernovae, RR Lyrae and asteroids. On the other hand, there are stronger discrepancies for non–variables and eclipsing binaries, indicating some discordance between simulated and real datasets. The right plot in Figure 4.7 shows the 2D projection of the augmented dataset, i.e. the simulated dataset plus 10 real light curves per class randomly sampled. It can be clearly seen that this new distribution contains samples of eclipsing binaries and non–variables, among others, that are not present on the left plot. Each real sample added can be weighted with more importance through the fine-tuning phase of the RCNN by iterating these examples more times than the simulated ones, or making more copies of each real sample in the augmented training set in the case of the light curve RF classifier.

For practical usage and deployment in an alert streaming classification scenario, the RCNN classifier presents some advantages compared to the light curve RF classifier. First, it is not necessary to compute neither the difference image nor the light curve, which avoids wrong subtractions due to alignment errors or PSF matching, it also reduces the amount of previous computation and information retrieval necessary to compute them. Second, for the RCNN the evaluation time increases linearly with the number of samples within the sequence, while for the light curve RF classifier the evaluation time depends on the complexity of the features used. Preliminary results show that the RCNN model evaluation running in GPU is considerably faster compared to the feature computation of FATS in CPU per example. Further analysis regarding evaluation speed should be done, but the RCNN seems as a good candidate where time limitations are important. Furthermore in the case of a streaming,

with the proposed approach it is not necessary to recompute any feature when a new sample from a source is received, only the current sample and the $n_w - 1$ previous images, plus the network state is needed to evaluate the model while RF requires the complete light curve data retrieving and recomputation of features, this can be improved with a good design of the database, for example, predicting possible data retrieving depending on the observed region of the sky. Also, designing online features to reduce computation and memory, as well as implementing features using GPUs processors when possible, considering that GPU processors are convenient when the operations involved are highly parallelizable. Third, the RCNN model is able to recognize other sources (stars or galaxies) from the image stamps beyond the flux from the main source. The RCNN model learns the distribution of these extra sources from the training set and applies it automatically when the evaluation is performed, without needing additional image pre-processing, to create a flag or feature associated with these extra sources. This is particularly useful when a detailed catalog of nearby objects such as other stars or galaxies is not available. Fourth, the RCNN model trained using simulated images can be fine tuned when receiving real images from a stream, using a few labeled real images.

On the other hand, the feature based light curve RF classifier still presents some desirable properties such as the interpretability of the features, the feature relevance for the classification task, easy implementation (except for the need to recompute features using the entire light curve at every time step) and so far a better generalization capacity without fine tuning. In a realistic setup, combining both approaches may improve the overall performance.

4.3 Visual evaluation of the proposed model

In this section some examples are shown of the image sequence RCNN classifier working on HiTS supernovae examples. From Figures 4.8 to 4.10, the light curve in counts of a supernova and the first detection time (upper plot) is depicted, also probabilities of the objects being of a certain class according to the model through time (mid plot) and the stamps corresponding to each observation date used as input (bottom plot). As mentioned in Section 4.1, the performance of the classifier gets better around the first detection of the supernovae (image number six), for some of the examples shown, the RCNN is able to observe some non-zero flux below the detection threshold.

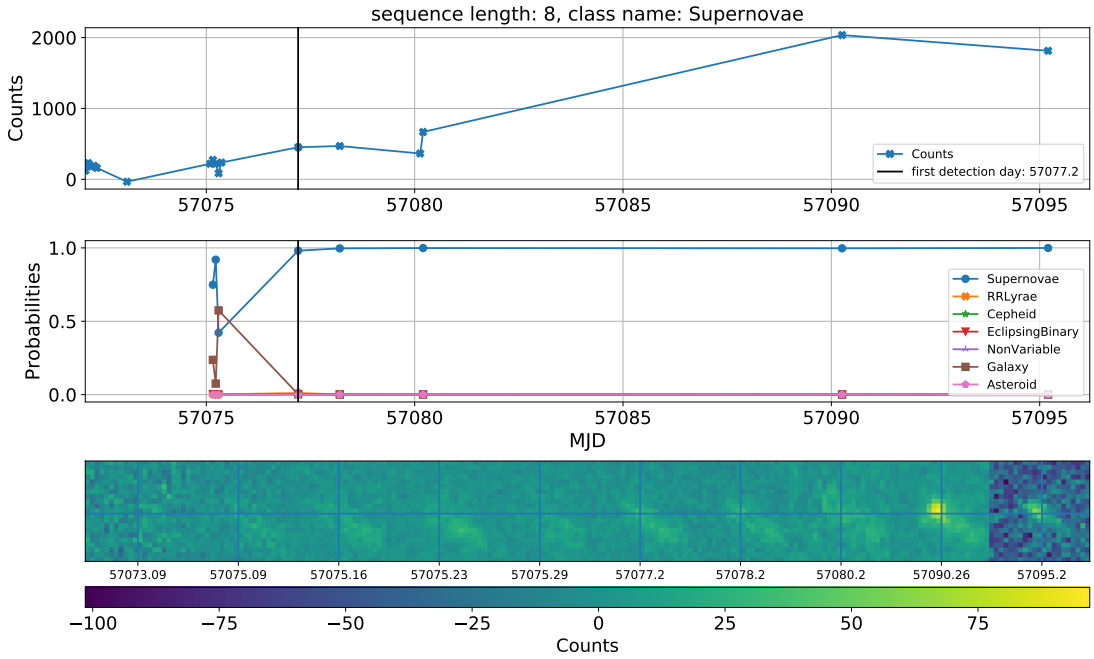


Figure 4.8: Example 1 of SN classification using the image sequence RCNN classifier.

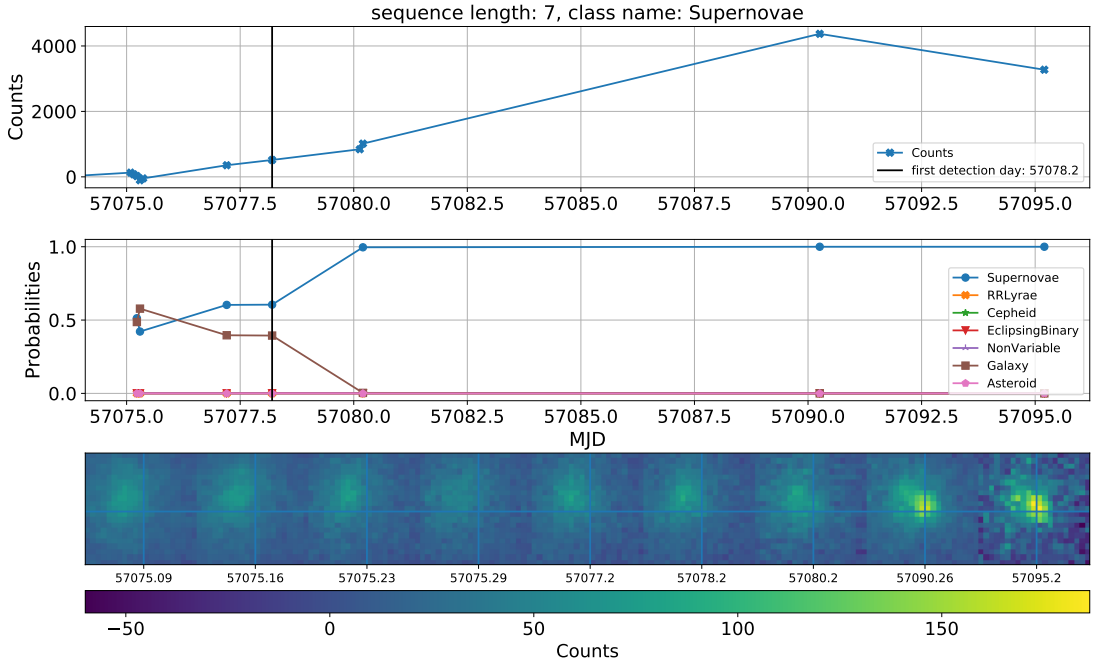


Figure 4.9: Example 2 of SN classification using the image sequence RCNN classifier.

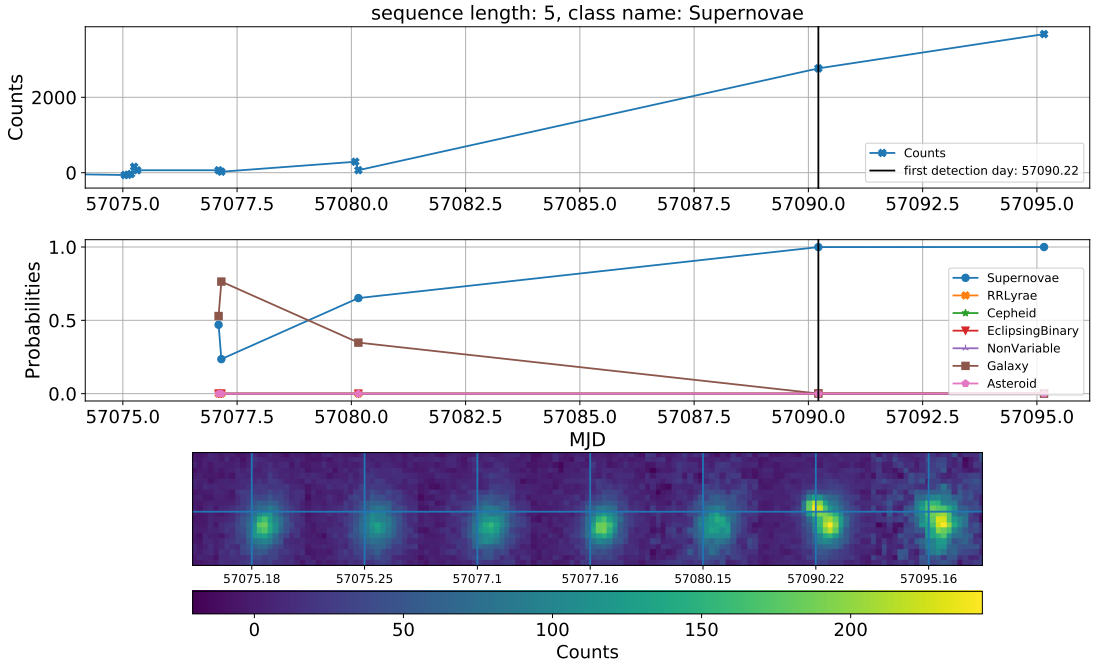


Figure 4.10: Example 3 of SN classification using the image sequence RCNN classifier.

4.4 Current difficulties and proposed solutions

In this section, the assumptions and difficulties of the proposed methodology are discussed. The analysis is divided into two parts. First, the proposed RCNN classifier problems and possible improvements. Then, the image simulation methodology is reviewed, also the main properties and proposals to improve the methodology under certain scenarios.

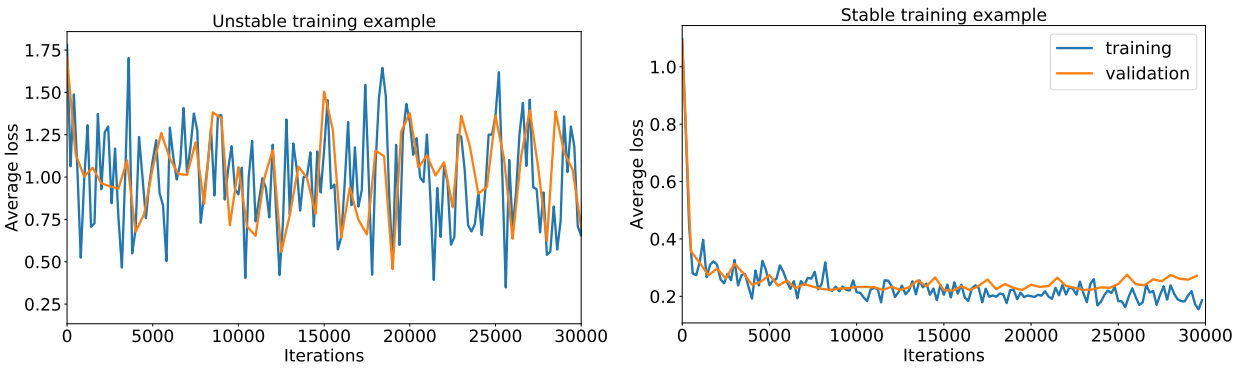
4.4.1 Classification model

The proposed RCNN classifier presents some properties that are discussed in this section, some are disadvantages compared to other classifiers, others are opportunities to extend the current model for better results and usability in real case scenarios.

Hyperparameters and architecture selection: The final proposed RCNN model was a product of trial and error of different architectures. The first model architectures had very unstable training curves. By removing the batch normalization layer between the convolutional layers [98], the training got stable enough to train the model properly. Then adding batch renormalization layers [99], improved the results without producing instability. A comparison of the two learning curves is shown in Figure 4.11a and Figure 4.11b. The cause of this effect is not clear, it could be produced by an implementation error on the batch normalization code. Another cause could be the renormalization factor, which reduces the difference of statistics (mean and variance) between training and inference of the input, that may help the training for a recurrent structure since the statistics of inputs is different between time steps. Furthermore, other normalization techniques were applied, such as layer normalization [112], which also presented acceptable training curve, but more unstable compare to batch renormalization. It is possible that most of the changes in architectures when finding the best model were masked by the unstable training due to normalization, therefore further work of architecture selection and hyperparameter selection (learning rates, dropout probability, gradient method) is recommended.

Adding extra information: Some high-level features could be useful for classification tasks of astronomical objects. The light curve and the estimated period from the light curve are known for being helpful for classification of variable stars. A possible next step could be adding the light curve to the classifier, this could be done by feeding the estimated brightness of the source to the recurrent layer. In the case of the ZTF alert streaming [113], an estimation of the magnitude of the source is reported in the alert package, so it could be added as extra information to the classifier. The period and other useful features could be extracted from the light curve, but in order to not harm the online advantage of the proposed model, ideally the extracted features should be calculated online, in the sense that updating the features does not involve recomputing the entire sequence, but only a subset of the points of the light curve and previous estimations of the feature.

Features related to the context within the image could be useful too. The presence of



(a) Unstable training

(b) Stable training

Figure 4.11: Unstable training curve (a) of the RCNN which was improved to (b) by removing the batch normalization layers between convolutions. It is not clear what is the cause of this effect, it could be either due to an implementation error or due to the recurrent structure.

nearby objects or possible bad pixels seem good options. Also, features that describe the shapes of the source in the difference image could help to detect dipoles or other errors. Previous work on this is the original HiTS paper where they designed features of the science, template and difference image [9]. Furthermore, concatenating the template and difference image within the input is something not tested but should be easy to implement.

Model interpretability: Interpretability of deep neural networks is a challenging problem. Efforts have been done in this area [87, 114]. Particularly, the Layer Relevance Propagation (LRP) method [115, 116] is interesting in this specific case. It could be possible to prove by using this tool that the structure within the image that is not the target source is used by the classifier to make a decision about the class. Furthermore, so far LRP have not been applied to recurrent neural networks combined with convolutions, therefore extending this method for this particular model might be a good way to find interpretability for the proposed model.

In order to find some relationship between the activations and the state of the recurrent model and possible useful features for the classification problem, the following "straightforward" method is proposed: First, project the state of the trained network using the training data, to a 2D or 3D space where the evolution of the state as a function of time steps can be visualized depending on the class, in a way that is easy to identify classes. Then, define a target feature to study, for example, period, variance, amplitude, etc. Generate a synthetic sequence that varies on the selected feature, for example, a frequency modulated signal in order to study how the period affects the classification. When evaluating the synthetic signal, project the state of the recurrent model into the 2D (or 3D) space and see how the projection moves through this space and through different classes. A similar way of doing this last step could be creating a set of sequences with different frequencies and inspect where the projection of the state lies in the 2D space. This proposed method could be understood as seeing the response of the classifier as a system, in a way that it is interpretable through the projected space (output) and the control of the parameters in the input to the model.

Multi-band approach: Convolutional neural networks naturally deal with inputs with more than one channel. For RGB images, for example, colors are represented using one channel per color. In astronomy, more than one band could be used to observe the same object, for example, LSST will use six different bands. The main problem of these observations in different bands is that they are taken at different irregular times, and not at the same time for each band, so the natural way of adding observations in different bands in the input tensor as concatenated images in the channel dimension is probably not the best way for the following reasons.

First, it is not always possible to have every band measured, objects measured only in some of the bands will certainly exist and the fixed number of channels force to some sort of data imputation method, furthermore it is not clear how to integrate the time dependencies to the model. Second, fixing the channel seems reasonable following the RGB image example, but by doing this, it could randomly sort the time order between channels. For instance, assume that the model receives images from band g and r where channel dimension is fixed, this means channel one is for g and channel two is for r. For times t, $t+1$ and $t+2$ it could happen that the model receive images denoted by their respective bands and times as g_t, r_{t+1}, g_{t+2}. Then, concatenating these images in a similar way as was done for the proposed model, the first input could be $[g_t, r_{t+1}]$ and the second $[g_{t+2}, r_{t+1}]$. In the proposed model, the time difference between images Δt in the input tensor is given to the recurrent layer, but it is clear in this case that the second input breaks the time order which leads to $\Delta t < 0$, generating an ambiguity about the meaning of Δt, which could be going back in time of the sequence or the actual difference between the images within the tensor. Processing more than two bands is even more complicated. There could be other ways to integrate information of different band preserving time and using a single input tensor, but it should be possible to solve this by separating the process for each band to integrate then in a common space within the architecture, as shown later.

The model in Figure 4.12 is proposed as a multiband image sequence classifier. This model uses different convolutional neural networks per band that are connected to the same recurrent neural network at the output. This architecture accounts for the differences between images from different bands since different filters will process each band. The statistical sampling time and the corresponding band are added to the model in the common network at the top. The model is evaluated only using the image available at a certain time, then the gradient is propagated only in the corresponding convolutional neural network depending on the band of the available image. This model naturally improves the cadence of the input sequence since it is not necessary to wait for an incoming image of a specific band. Instead, for each time the model is able to process every incoming image using a different part of the network depending on the band, being able also to integrate information between bands in the common space.

Robustness of observation conditions: In this work, real observation conditions were used to simulate the image sequence of each astronomical object. Then, the real images examples to fine tune and evaluate the model have roughly the same observation conditions. Obviously, this is not always possible, especially in cases where observations have not been made, such as the LSST. In order to apply this methodology for future observations, it might

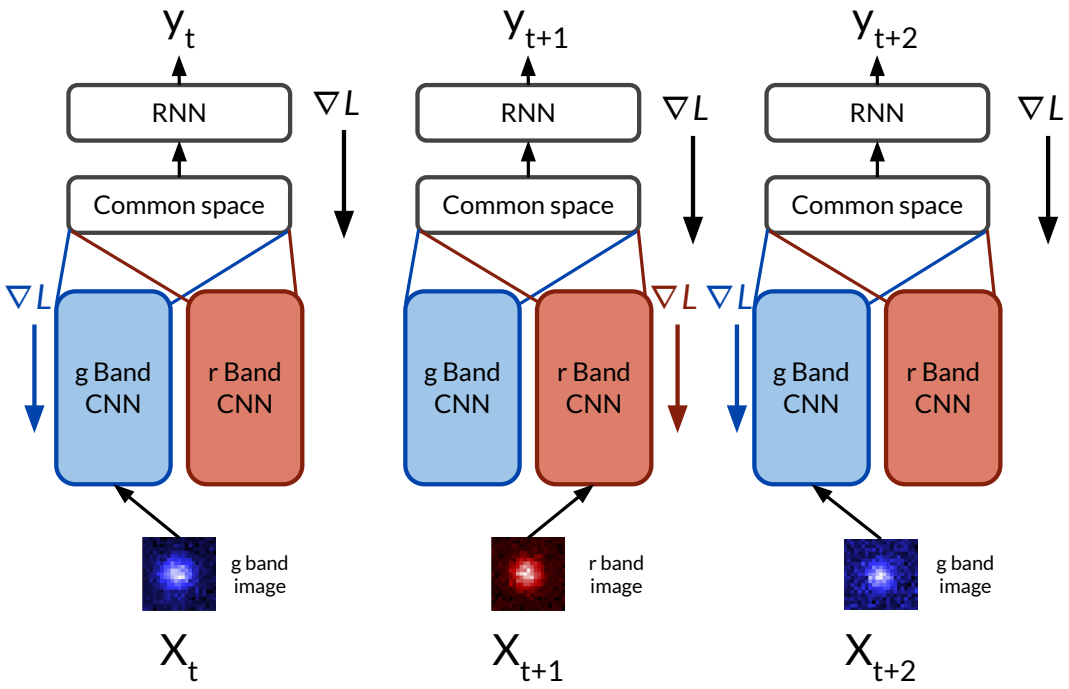


Figure 4.12: Proposed multi-band classifier. Every incoming image is processed by different convolutional layers depending on the band, then the information is projected to a common space, in this way, the model is able to integrate information from different bands and improves the cadence of the input sequence. The gradient is propagated in a regular way depending on the available band for each time step.

be necessary to do observation conditions forecasting to simulate the images. It is reasonable to assume that most forecasting methods will have a prediction error, so it is important to make the model more robust to differences between observation conditions on training data and observation conditions for evaluation data. One way to achieve this is by adding noise somehow or sorting the observation conditions for simulated data, so the model gain invariance or robustness to changes in observation conditions between training and test data.

4.4.2 Simulation-Classification methodology

Applying the proposed method (simulating a sequence of images and training the proposed model as explained in Chapter 3) in a real case scenario, give rise some questions regarding the objects that are possible to find using this methodology. One of the main issues is how to find objects that have not been observed with real instruments, even objects that are not known to exist.

Another issue is the simulation of physical effects and prior assumptions about the image structures that are different from the source. The implemented process for image simulation used in this work only consider a couple of physical effects and single elliptically shaped galaxies, and do not considers, for instance, the correlated noise between nearby pixels, saturation effects, hot pixels, bad columns, crowded fields with many sources, galaxies with more

complex shapes such as arms or irregular galaxies, among others. In order to approach these problems and some of the ones presented in Section 4.4.1, it is proposed the following scheme shown in Figure 4.13 of different systems to isolate important parts of the methodology.

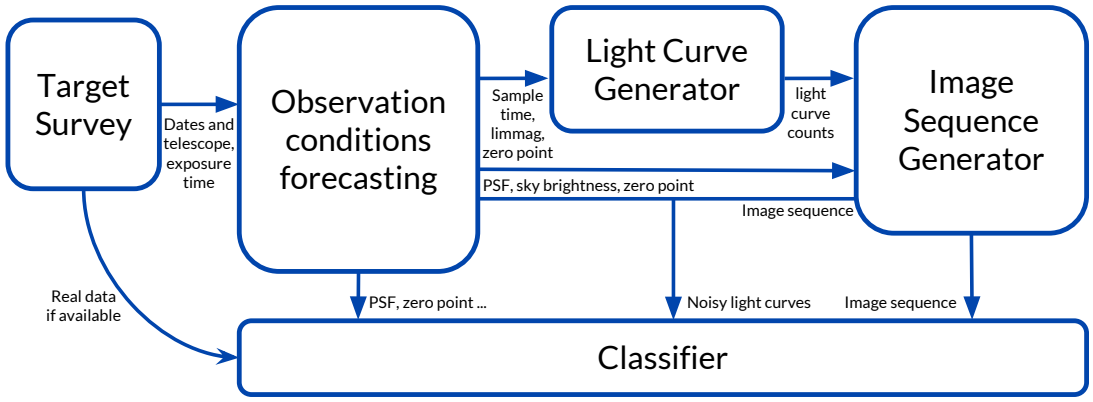


Figure 4.13: Proposed system separation

First, defining an observation plan and the telescope (target survey) determines the observation dates of the survey and the camera parameters, then it should be possible to apply a forecasting method to determine the observation conditions for each exposure. This forecasting process should be isolated from the light curve generator and the image generator since it shouldn't be necessary to consider information from these other two parts.

The information about observation dates, observation conditions, and camera parameters goes from the forecasting block to the light curve generator. Again, the light curve generator should be independent of the image generator. This makes easier to add new astronomical objects to the light curve generator to simulate their images later, which may help to find objects not observed before with real instrument but for which a light curve model is available.

The image generator receives information from both, the forecasting block and generated light curves. It is proposed to simulate images independently of the shape of the light curve and only consider each point of the light curve to create the images, in other words, only instantaneous information of the astronomical objects should be considered, so the image generator could create objects with an arbitrary shaped light curve, as well as arbitrary observation conditions which were defined by the forecasting block. This facilitates the addition of new astronomical objects by only adding the light curve model to the generator. Then the shape of the light curve that describes the object is managed in the light curve generator model and not the image generator.

Although, the image generator should consider instant information from observation conditions and the light curve, there is some sequential information that the image generator must consider to create realistic image sequences, this information corresponds to the structures within a stamp. For example, the presence of galaxies or other sources different from source simulated using the light curve. Through the entire sequence, the underlying structure of other sources should remain constant, independent of the observation conditions or variations in brightness of the source generated by the light curve shape. Furthermore, prior

distributions for the occurrence of certain features within the stamp sequence such as hot pixels, bad columns or the shape of the PSF could be learned from real data.

An option is to use of an architecture based on **Generative Adversarial Networks** (GANs) [117, 118] that manage part of the issues discussed. In Figure 4.14, a general idea of the proposed architecture is depicted, which is a mix of conditional and convolutional GANs [119, 120]. In principle, variables that could be learned from data should be encoded in the latent space through training, variables that are defined by the light curve or observation conditions are given to the model to "condition" the image generation. Then this model could be designed to learn the distributions of occurrence and shapes of effects, for which the user doesn't want to set a prior distribution manually, such as the shape of the galaxy or presence of a bad column in the stamps. At the same time, keeps important variables to generate the image under the control of the user, such as the brightness of the source controlled by the light curve and the observation conditions (background level, PSF size, etc). In summary, for variables for which is better to learn the prior distributions, those are left to be encoded in the latent space. Variables that the user wants to control, could be used to condition the image generation. As mentioned before, some sequential information is still important, this is why the proposed architecture also has a recurrent structure, where should be able to keep sequence information in the state.

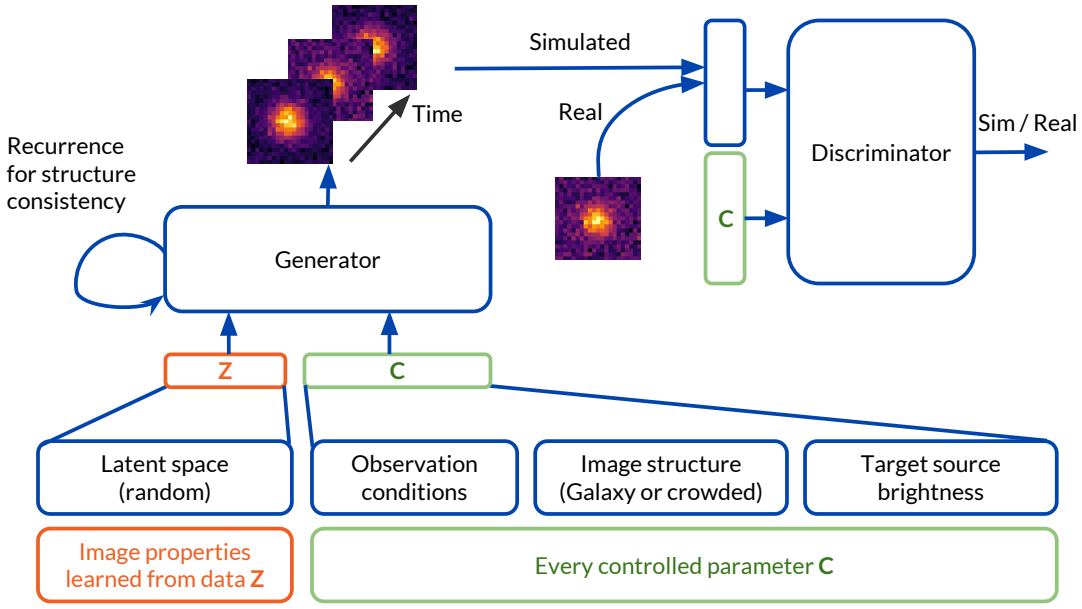


Figure 4.14: Generative model for image simulation. The generator receives a random variable sampled from the distribution of the latent space **Z**, and also extra variables **C** that condition the image generation. The discriminator also receives the controlled parameters **C** to force the generator to correctly produce images with the right parameters.

The proposed generative model does not require labeled object sequences, it only requires sequences of stamps for which the controlled parameters **C** are known, in other words, unlabeled raw images from a telescope is the only data needed since the parameters in **C** could be inferred from the images. Then, sequence of a specific object can be generated and labeled according to the light curve model. This way to simulate sequence of images do not have the advantage of having a dataset before real data is available, so the alternative approach

of adding physical effects manually should not be abandoned. The choice of each approach will depend on the problem and data availability.

The classifier block in Figure 4.13 is the last step, which could receive information from every other previous step. The proposed RCNN developed in this work is a good candidate and could be improved as explained in Section 4.4.1. Keeping each problem in the right block helps to encapsulate each problem and solve them independently. Although the code integration was not discussed in this work, it is an important factor when building large systems. Defining the right communication between each block and considering usability for different cases and applications is just as important as any of the issues discussed in this section.

Chapter 5

Conclusion

In this thesis, a new sequential classification model for astronomical objects based on a recurrent convolutional neural network is proposed, which uses directly sequences of images, and it does not require computing the light curve nor the difference image. Using empirical and astrophysical models of different astronomical objects, data from the High cadence Transient Survey (HiTS) was simulated, generating a dataset of image sequences for seven classes of astronomical objects, considering realistic atmospheric conditions and camera specifications. This synthetic dataset was used to train the image sequence classifier (RCNN), then the proposed model was evaluated on real images from the HiTS survey. Fine tuning was performed to adjust the models obtained with the simulated dataset using a few real samples per class. The results show that the proposed RCNN model is able to classify correctly the simulated test set, as well as the real dataset after performing fine tuning. This is the first time that a sequential classifier based on recurrent convolutional neural networks using sequences of images as inputs, and without computing the light curve or the difference images, has been proposed in time-domain astronomy.

In order to assess the advantage of using images directly, light curves were extracted using optimal photometry on the simulated images and we trained a feature based light curve RF classifier using Feature Analysis for Time Series package (FATS). Although the proposed RCNN model obtained similar results to those of the more traditional approach, it presents several advantages in an alert streaming classification scenario: no need for expensive preprocessing such as difference imaging and light curve computation, a faster evaluation, early detection compared to RF due to extra information aside from the source presented in the stamp such as the presence of a nearby galaxies, and an easy way to fine tune the model after receiving new images from the data streaming.

This work also shows that having images at the location of a transient event from before their first detection, can be very useful for the classification of astronomical alert streams. For example, in the proposed RCNN model having images of a supernova very early rise, before crossing a given threshold on the light curve, helps the classifier report better class probabilities.

Using realistic image simulations makes the domain adaptation problem from synthetic

to real images easier to solve. As mentioned before, the performance of the proposed model was substantially improved by doing fine tuning with only 10 real samples per class. Using synthetic data is a reliable way to train models before acquiring real images from telescopes, as long as we have good light curve models available and the correct parameter distributions to represent well objects for a specific classification task and science goal. The fact that the RCNN classifier works well on real-world images after being trained with synthetic data and improves its performance after fine tuning, encourages further use of this methodology to train classifiers for new telescopes such as the Large Synoptic Survey Telescope [1] and the Zwicky Transient Facility [2]. In this way, it could be possible to have a sequential classifier model available even before receiving real data, and as new data becomes available we can easily adjust the proposed model.

5.1 Future Work

As discussed in section 4.4.1, the proposed model could be improved in many ways. Hyperparameters search and architecture selection is not completely done in this work, it is important to find the best configuration in order to improve performance on simulated and real data, and also reduce the architecture size if possible.

More information could be given to the classifier in order to facilitate the classification task and the domain adaptation problem between simulated and real data. The most obvious candidate of extra information could be the light curve (since is reported in ZTF alert streaming), observation condition information or other contextual feature such the presence of a nearby galaxy. We could also use features used by the random forest. Hopefully, features computed in an efficient fashion to not harm the advantages of the proposed model.

Interpretability of the model is an interesting topic, especially in this case where the model encodes spatial and temporal information of the astronomical object. Layer relevance propagation (LRP) is proposed to interpret how the model discriminates among the classes. LRP has not been applied to a convolutional and recurrent architecture, therefore is a great opportunity to develop a new tool for interpretability in astronomical classification tasks.

The proposed model is currently trained in band g, but it could be adapted to classify the image sequence combining information from more than one band, i.e., a multi-band image sequence classifier. Since images from different bands are sampled at an irregular time and not at the same instant, an architecture (see 4.12) is proposed to process images from different bands with different parts of the network, which also integrates the information from each band in a common space and improves the cadence of the input sequence.

An important assumption in this work is the availability of good observation conditions to simulate images. In order to simulate images for which data is not available yet (LSST for example), it is necessary to estimate or forecast future observation conditions and it is reasonable to expect errors in the forecasting. An alternative is to make the proposed model more robust or invariant to observation conditions, so the discrepancies between training and test data do not harm the performance in classification.

In terms of the proposed methodology, many improvements to the approach could be done considering certain issues. For instance, how to find objects that have never been observed with real instruments, or how to simulate better images considering realistic physical effects.

In this work, it is proposed to separate blocks to tackle each problem independently. Starting from the observation conditions forecasting block, light curve generator, the image generator, and classifier. Each of these blocks should solve their particular problems independently allowing easy integration of the entire system. For example, in order to add a new object to the dataset, in principle it is required a mathematical or empirical model only, that describes the light curve for the corresponding bands of the survey. This enables the attempt to find objects that have not been observed before but for which a model is available.

To improve the image simulator, it is proposed to use a generative model based on Generative Adversarial Networks (GANs). The proposed model (see 4.14) is designed to learn part of the physical effects of the images that are present in the data, and for which it is hard to make assumptions about the prior distributions, such as camera errors or galaxy shapes. At the same time, it keeps control for some of the parameters by conditioning the generator. In particular, it could be used to generate a sequence of images with realistic characteristics for any arbitrary shaped light curve. The original approach of adding physical effects to the simulator should not be abandoned because the generative approach needs real data for training which is not always available.

5.2 Last Remarks

The two main contributions of this work were the image simulator program and the image sequence RCNN classifier. By developing these tools, it was proved that the direct use of images to classify astronomical objects is reliable and it is very suitable for an alert streaming scenario.

The topics discussed in chapter 4 open many possibilities to solve problems such as the following: Interpretation of recurrent neural networks, by using LRP, for example. How to correctly condition and add knowledge to a generative model, like the proposed model for image generation. Integrating useful information within the neural network to solve the classification task, and how to combine the information for a multi-band classifier as the proposed architecture in Figure 4.12.

Of course, before any of these proposed approaches are definite, further analysis and research has to be done, but having a notion of the question or the problem that wants to be solved is a good starting point.

Bibliography

[1] Ivezić, S. M. Kahn, J. A. Tyson, B. Abel, E. Acosta *et al.*, "LSST: from Science Drivers to Reference Design and Anticipated Data Products," *ArXiv e-prints*, vol. 0805, p. arXiv:0805.2366, May 2008.

[2] R. M. Smith, R. G. Dekany, C. Bebek, E. Bellm, K. Bui *et al.*, "The Zwicky transient facility observing system," *Ground-based and Airborne Instrumentation for Astronomy V*, vol. 9147, p. 914779, Jul. 2014.

[3] A. G. Riess, A. V. Filippenko, P. Challis, A. Clocchiatti, A. Diercks *et al.*, "Observational Evidence from Supernovae for an Accelerating Universe and a Cosmological Constant," *The Astronomical Journal*, vol. 116, pp. 1009–1038, Sep. 1998.

[4] B. P. Schmidt, N. B. Suntzeff, M. M. Phillips, R. A. Schommer, A. Clocchiatti *et al.*, "The High-Z Supernova Search: Measuring Cosmic Deceleration and Global Curvature of the Universe Using Type IA Supernovae," *The Astrophysical Journal*, vol. 507, pp. 46–63, Nov. 1998.

[5] C.-C. Ngeow, W. Gieren, and C. Klein, "Distance determination from the Cepheid and RR Lyrae period-luminosity relations," *Proceedings of the International Astronomical Union*, vol. 9, no. S301, pp. 123–128, Aug. 2013.

[6] M. W. Feast, J. W. Menzies, N. Matsunaga, and P. A. Whitelock, "Cepheid variables in the flared outer disk of our galaxy," *Nature*, vol. 509, no. 7500, pp. 342–344, May 2014.

[7] G. Narayan, T. Zaidi, M. D. Soraisam, Z. Wang, M. Lochner *et al.*, "Machine-learning-based Brokers for Real-time Classification of the LSST Alert Stream," *The Astrophysical Journal Supplement Series*, vol. 236, p. 9, May 2018.

[8] T. Naylor, "An optimal extraction algorithm for imaging photometry," *Monthly Notices of the Royal Astronomical Society*, vol. 296, no. 2, pp. 339–346, May 1998.

[9] F. Förster, J. C. Maureira, J. San Martín, M. Hamuy, J. Martínez *et al.*, "The High Cadence Transient Survey (HITS). I. Survey Design and Supernova Shock Breakout Constraints," *The Astrophysical Journal*, vol. 832, p. 155, Dec. 2016.

[10] V. Belokurov, N. W. Evans, and Y. L. Du, "Light-curve classification in massive vari-

ability surveys — I. Microlensing," *Monthly Notices of the Royal Astronomical Society*, vol. 341, no. 4, pp. 1373–1384, Jun. 2003.

[11] V. Belokurov, N. W. Evans, and Y. Le Du, "Light-curve classification in massive variability surveys – II. Transients towards the Large Magellanic Cloud," *Monthly Notices of the Royal Astronomical Society*, vol. 352, no. 1, pp. 233–242, Jul. 2004.

[12] H. Brink, J. W. Richards, D. Poznanski, J. S. Bloom, J. Rice *et al.*, "Using machine learning for discovery in synoptic survey imaging data," *Monthly Notices of the Royal Astronomical Society*, vol. 435, pp. 1047–1060, Oct. 2013.

[13] I. Nun, P. Protopapas, B. Sim, M. Zhu, R. Dave *et al.*, "FATS: Feature Analysis for Time Series," *ArXiv e-prints*, vol. 1506, p. arXiv:1506.00010, May 2015.

[14] P. Benavente, P. Protopapas, and K. Pichara, "Automatic Survey-invariant Classification of Variable Stars," *The Astrophysical Journal*, vol. 845, no. 2, p. 147, 2017.

[15] N. Castro, P. Protopapas, and K. Pichara, "Uncertain Classification of Variable Stars: Handling Observational GAPS and Noise," *The Astronomical Journal*, vol. 155, no. 1, p. 16, 2018.

[16] G. Cabrera-Vives, I. Reyes, F. Förster, P. A. Estévez, and J. C. Maureira, "Supernovae detection by using convolutional neural networks," in *2016 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2016, pp. 251–258.

[17] G. Cabrera-Vives, I. Reyes, F. Förster, P. A. Estévez, and J.-C. Maureira, "Deep-HiTS: Rotation Invariant Convolutional Neural Network for Transient Detection," *The Astrophysical Journal*, vol. 836, p. 97, Feb. 2017.

[18] N. Sedaghat and A. Mahabal, "Effective image differencing with convolutional neural networks for real-time transient hunting," *Monthly Notices of the Royal Astronomical Society*, vol. 476, pp. 5365–5376, Jun. 2018.

[19] D. George and E. A. Huerta, "Deep Learning for real-time gravitational wave detection and parameter estimation: Results with Advanced LIGO data," *Physics Letters B*, vol. 778, pp. 64–70, Mar. 2018.

[20] C. J. Shallue and A. Vanderburg, "Identifying Exoplanets with Deep Learning: A Five-planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90," *The Astronomical Journal*, vol. 155, no. 2, p. 94, 2018.

[21] T. Charnock and A. Moss, "Deep Recurrent Neural Networks for Supernovae Classification," *The Astrophysical Journal Letters*, vol. 837, p. L28, Mar. 2017.

[22] A. Mahabal, K. Sheth, F. Gieseke, A. Pai, S. G. Djorgovski *et al.*, "Deep-learnt classification of light curves," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, Nov. 2017, pp. 1–8.

[23] P. Protopapas, "Recurrent Neural Network Applications for Astronomical Time Series,"

in *American Astronomical Society Meeting Abstracts #230*, ser. American Astronomical Society Meeting Abstracts, vol. 230, Jun. 2017, p. 104.03.

[24] B. Naul, J. S. Bloom, F. Pérez, and S. van der Walt, "A recurrent neural network for classification of unevenly sampled variable stars," *Nature Astronomy*, vol. 2, pp. 151–155, Feb. 2018.

[25] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, Apr. 1980.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. USA: Curran Associates Inc., 2012, pp. 1097–1105.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 1–9.

[28] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 1096–1104.

[29] O. Abdel-Hamid, A. r. Mohamed, H. Jiang, L. Deng, G. Penn *et al.*, "Convolutional Neural Networks for Speech Recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, pp. 1533–1545, Oct. 2014.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[31] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with LSTM," in *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, vol. 2, 1999, pp. 850–855 vol.2.

[32] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *INTERSPEECH*, 2014.

[33] Y. Goldberg, "Neural Network Methods for Natural Language Processing," *Synthesis Lectures on Human Language Technologies*, vol. 10, no. 1, pp. 1–309, Apr. 2017.

[34] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3104–3112.

[35] A. Graves, A. r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and*

Signal Processing, May 2013, pp. 6645–6649.

[36] A. Kimura, I. Takahashi, M. Tanaka, N. Yasuda, N. Ueda *et al.*, "Single-epoch supernova classification with deep convolutional neural networks," *ArXiv e-prints*, vol. 1711, p. arXiv:1711.11526, Nov. 2017.

[37] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2015, pp. 4580–4584.

[38] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama *et al.*, "Long-Term Recurrent Convolutional Networks for Visual Recognition and Description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 677–691, Apr. 2017.

[39] R. Zhao, H. Ali, and P. v. d. Smagt, "Two-stream RNN/CNN for action recognition in 3d videos," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 4260–4267.

[40] Y. Zhao, X. Jin, and X. Hu, "Recurrent convolutional neural network for speech processing," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2017, pp. 5300–5304.

[41] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How Transferable Are Features in Deep Neural Networks?" in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, pp. 3320–3328.

[42] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 1717–1724.

[43] LIGO Scientific Collaboration and Virgo Collaboration, B. Abbott, R. Abbott, T. Abbott, M. Abernathy *et al.*, "Observation of Gravitational Waves from a Binary Black Hole Merger," *Physical Review Letters*, vol. 116, no. 6, p. 061102, Feb. 2016.

[44] I. Collaboration, "Neutrino emission from the direction of the blazar TXS 0506+056 prior to the IceCube-170922a alert," *Science*, vol. 361, no. 6398, pp. 147–151, Jul. 2018.

[45] T. I. Collaboration, Fermi-LAT, Magic, Agile, Asas-Sn *et al.*, "Multimessenger observations of a flaring blazar coincident with high-energy neutrino IceCube-170922a," *Science*, vol. 361, no. 6398, p. eaat1378, Jul. 2018.

[46] T. A. Thompson, A. Burrows, and P. A. Pinto, "Shock Breakout in Core-Collapse Supernovae and Its Neutrino Signature," *The Astrophysical Journal*, vol. 592, pp. 434–456, Jul. 2003.

[47] F. Forster, T. J. Moriya, J. C. Maureira, J. P. Anderson, S. Blinnikov *et al.*, "The delay of shock breakout due to circumstellar material evident in most type II supernovae."

Nature Astronomy, vol. 2, p. 808, 2018.

[48] C. Alcock, R. A. Allsman, D. R. Alves, T. S. Axelrod, A. C. Becker *et al.*, "The MACHO Project: Microlensing Detection Efficiency," *The Astrophysical Journal Supplement Series*, vol. 136, no. 2, p. 439, 2001.

[49] G. Pojmański, "The All Sky Automated Survey," *Astronomische Nachrichten*, vol. 325, no. 6-8, pp. 553–555, 2004.

[50] M. K. Szymanski, "The optical gravitational lensing experiment. internet access to the ogle photometry data set: ogle-II bvi maps and i-band data," *Acta Astron.*, vol. 55, pp. 43–57, 2005.

[51] J. B. Kaler, *Stars and Their Spectra: An Introduction to the Spectral Sequence*. Cambridge University Press, Mar. 1997, google-Books-ID: 4fNhk7m2MGYC.

[52] D. W. Hogg, "Distance measures in cosmology," *arXiv:astro-ph/9905116*, May 1999, arXiv: astro-ph/9905116.

[53] J. Martínez-Palomera, F. Förster, P. Protopapas, J. C. Maureira, Paulina Lira *et al.*, "The High Cadence Transit Survey (HiTS): Compilation and Characterization of Light-curve Catalogs," *The Astronomical Journal*, vol. 156, no. 5, p. 186, 2018.

[54] R. M. Quimby, G. Aldering, J. C. Wheeler, P. Höflich, C. W. Akerlof *et al.*, "SN 2005ap: A Most Brilliant Explosion," *The Astrophysical Journal*, vol. 668, no. 2, pp. L99–L102, Oct. 2007.

[55] J. Vinkó, F. Yuan, R. M. Quimby, J. C. Wheeler, E. Ramirez-Ruiz *et al.*, "A Luminous, Fast Rising UV-transient Discovered by ROTSE: A Tidal Disruption Event?" *The Astrophysical Journal*, vol. 798, no. 1, p. 12, 2015.

[56] J. Wambsganss, "Gravitational Lensing - A brief review," *ArXiv Astrophysics e-prints*, pp. arXiv:astro–ph/0 012 423, Dec. 2000.

[57] P. A. Mazzali, F. K. Röpke, S. Benetti, and W. Hillebrandt, "A Common Explosion Mechanism for Type Ia Supernovae," *Science*, vol. 315, no. 5813, pp. 825–828, Feb. 2007.

[58] E. Y. Hsiao, A. Conley, D. A. Howell, M. Sullivan, C. J. Pritchet *et al.*, "K-Corrections and Spectral Templates of Type Ia Supernovae," *The Astrophysical Journal*, vol. 663, pp. 1187–1200, Jul. 2007.

[59] T. J. Moriya, P. A. Mazzali, N. Tominaga, S. Hachinger, S. I. Blinnikov *et al.*, "Light-curve and spectral properties of ultrastripped core-collapse supernovae leading to binary neutron stars," *Monthly Notices of the Royal Astronomical Society*, vol. 466, pp. 2085–2098, Apr. 2017.

[60] M. Catelan and H. A. Smith, *Pulsating Stars*, 2015, oCLC: 899211454.

[61] C. J. Lintott, K. Schawinski, A. Slosar, K. Land, S. Bamford *et al.*, "Galaxy Zoo: morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey," *Monthly Notices of the Royal Astronomical Society*, vol. 389, pp. 1179–1189, Sep. 2008.

[62] T. Diehl, "The Dark Energy Survey Camera (DECam)," *Physics Procedia*, vol. 37, pp. 1332–1340, Jan. 2012.

[63] T. Fellers and M. W Davidson, "Concepts in Digital Imaging Technology - CCD Saturation and Blooming," Jan. 2014.

[64] B. Flaugher, H. T. Diehl, K. Honscheid, T. M. C. Abbott, O. Alvarez *et al.*, "The Dark Energy Camera," *The Astronomical Journal*, vol. 150, p. 150, Nov. 2015.

[65] R. Widenhorn, M. M. Blouke, A. Weber, A. Rest, and E. Bodegom, "Temperature dependence of dark current in a CCD," in *Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications III*, vol. 4669. International Society for Optics and Photonics, Apr. 2002, pp. 193–202.

[66] N. R. Lomb, "Least-squares frequency analysis of unequally spaced data," *Astrophysics and Space Science*, vol. 39, pp. 447–462, Feb. 1976.

[67] J. D. Scargle, "Studies in astronomical time series analysis. II - Statistical aspects of spectral analysis of unevenly spaced data," *The Astrophysical Journal*, vol. 263, pp. 835–853, Dec. 1982.

[68] L. C. Molina, L. Belanche, and A. Nebot, "Feature selection algorithms: a survey and experimental evaluation," in *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, Dec. 2002, pp. 306–313.

[69] J. R. Vergara and P. A. Estévez, "A review of feature selection methods based on mutual information," *Neural Computing and Applications*, vol. 24, no. 1, pp. 175–186, Jan. 2014.

[70] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[71] J. H. Friedman, "Greedy function approximation: A gradient boosting machine." *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001.

[72] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen *et al.*, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus *et al.*, Eds. Curran Associates, Inc., 2017, pp. 3146–3154.

[73] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. USA: Omnipress, 2010, pp. 807–814.

[74] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics*

of Control, Signals and Systems, vol. 2, no. 4, pp. 303–314, Dec. 1989.

[75] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, Jan. 1991.

[76] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[77] P. J. Werbos, *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*. New York, NY, USA: Wiley-Interscience, 1994.

[78] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, no. 1, pp. 145–151, Jan. 1999.

[79] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, Feb. 2011.

[80] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," *arXiv:1212.5701 [cs]*, Dec. 2012, arXiv: 1212.5701.

[81] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Dec. 2014, arXiv: 1412.6980.

[82] S. J. Reddi, S. Kale, and S. Kumar, "On the Convergence of Adam and Beyond," *ICLR*, Feb. 2018.

[83] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[84] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[85] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, no. 1, pp. 106–154.2, Jan. 1962.

[86] ——, "Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat," *Journal of Neurophysiology*, vol. 28, no. 2, pp. 229–289, Mar. 1965.

[87] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *Computer Vision – ECCV 2014*, ser. Lecture Notes in Computer Science, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer International Publishing, 2014, pp. 818–833.

[88] Zhou and Chellappa, "Computation of optical flow using a neural network," in *IEEE 1988 International Conference on Neural Networks*, Jul. 1988, pp. 71–78 vol.2.

[89] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778.

[90] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," *arXiv:1608.06993 [cs]*, Aug. 2016, arXiv: 1608.06993.

[91] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A Critical Review of Recurrent Neural Networks for Sequence Learning," *arXiv:1506.00019 [cs]*, May 2015, arXiv: 1506.00019.

[92] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[93] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares *et al.*, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1724–1734.

[94] D. Neil, M. Pfeiffer, and S.-C. Liu, "Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 3882–3890.

[95] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[96] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, Feb. 1970.

[97] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[98] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. Lille, France: JMLR.org, 2015, pp. 448–456.

[99] S. Ioffe, "Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus *et al.*, Eds. Curran Associates, Inc., 2017, pp. 1945–1953.

[100] B. Sesar, Ivezić, S. H. Grammer, D. P. Morgan, A. C. Becker *et al.*, "Light Curve Templates and Galactic Distribution of RR Lyrae Stars from Sloan Digital Sky Survey Stripe 82," *The Astrophysical Journal*, vol. 708, pp. 717–741, Jan. 2010.

[101] J. D. Hartman, D. Bersier, K. Z. Stanek, J.-P. Beaulieu, J. Kaluzny *et al.*, "Deep Canada-France-Hawaii Telescope photometric survey of the entire M33 galaxy - I. Cat-

alogue of 36000 variable point sources," *Monthly Notices of the Royal Astronomical Society*, vol. 371, pp. 1405–1417, Sep. 2006.

[102] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[103] M. R. Blanton, M. A. Bershady, B. Abolfathi, F. D. Albareti, C. Allende Prieto *et al.*, "Sloan Digital Sky Survey IV: Mapping the Milky Way, Nearby Galaxies, and the Distant Universe," *The Astronomical Journal*, vol. 154, p. 28, Jul. 2017.

[104] G. E. Medina, R. R. Muñoz, A. K. Vivas, J. L. Carlin, F. Förster *et al.*, "Discovery of Distant RR Lyrae Stars in the Milky Way Using DECam," *The Astrophysical Journal*, vol. 855, p. 43, Mar. 2018.

[105] K. C. Chambers, E. A. Magnier, N. Metcalfe, H. A. Flewelling, M. E. Huber *et al.*, "The Pan-STARRS1 Surveys," *ArXiv e-prints*, vol. 1612, p. arXiv:1612.05560, Dec. 2016.

[106] J. L. Sérsic, "Influence of the atmospheric and instrumental dispersion on the brightness distribution in a galaxy," *Boletin de la Asociacion Argentina de Astronomia La Plata Argentina*, vol. 6, p. 41, 1963.

[107] L. Ciotti, "Stellar systems following the R exp 1/m luminosity law," *Astronomy and Astrophysics*, vol. 249, pp. 99–106, Sep. 1991.

[108] K. Pichara and P. Protopapas, "Automatic Classification of Variable Stars in Catalogs with Missing Data," *The Astrophysical Journal*, vol. 777, no. 2, p. 83, 2013.

[109] D.-W. Kim, P. Protopapas, C. A. L. Bailer-Jones, Y.-I. Byun, S.-W. Chang *et al.*, "The EPOCH Project - I. Periodic variable stars in the EROS-2 LMC database," *Astronomy & Astrophysics*, vol. 566, p. A43, Jun. 2014.

[110] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen *et al.*, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015.

[111] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[112] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," *arXiv:1607.06450 [cs, stat]*, Jul. 2016, arXiv: 1607.06450.

[113] F. J. Masci, R. R. Laher, B. Rusholme, D. L. Shupe, S. Groom *et al.*, "The Zwicky Transient Facility: Data Processing, Products, and Archive," *Publications of the Astronomical Society of the Pacific*, vol. 131, no. 995, p. 018003, 2018.

[114] Q.-s. Zhang and S.-c. Zhu, "Visual interpretability for deep learning: a survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 1, pp. 27–39, Jan. 2018.

[115] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller *et al.*, "On Pixel-Wise Ex-

planations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation," *PLoS ONE*, vol. 10, no. 7, Jul. 2015.

[116] E. Reyes, P. A. Estévez, I. Reyes, G. Cabrera-Vives, P. Huijse *et al.*, "Enhanced Rotational Invariant Convolutional Neural Network for Supernovae Detection," in *2018 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2018, pp. 1–8.

[117] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley *et al.*, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680.

[118] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta *et al.*, "Generative Adversarial Networks: An Overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, Jan. 2018.

[119] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," *arXiv:1411.1784 [cs, stat]*, Nov. 2014, arXiv: 1411.1784.

[120] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *arXiv:1511.06434 [cs]*, Nov. 2015, arXiv: 1511.06434.

Appendix

FATS features

FATS features used in this work, see FATS documentation for more details [13].

Amplitude, AndersonDarling, Autocor_length, Beyond1Std, CAR_mean, CAR_sigma, CAR_tau, Color, Con, Eta_color, Eta_e, FluxPercentileRatioMid20, FluxPercentileRatioMid35, FluxPercentileRatioMid50, FluxPercentileRatioMid65, FluxPercentileRatioMid80 Freq$i$_harmonics_amplitude_j ($i \in \{1, 2, 3\}$, $j \in \{0, 1, 2, 3\}$), LinearTrend, MaxSlope, Mean, Meanvariance, MedianAbsDev, MedianBRP, PairSlopeTrend, PercentAmplitude, PercentDifferenceFluxPercentile, PeriodLS, Period_fit, Psi_CS, Psi_eta, Q31, Q31_color, Rcs, Skew, SlottedA_length, SmallKurtosis, Std, StetsonJ, StetsonK, StetsonK_AC, StetsonL.