FEDERAL UNIVERSITY OF CEARÁ

CENTER OF SCIENCE

DEPARTMENT OF PHYSICS

POST-GRADUATION PROGRAM IN PHYSICS

MASTER DEGREE IN PHYSICS

PAULO CLEBER FARIAS DA SILVA FILHO

LIGHT CURVE IMAGING FOR EXOPLANET DETECTION WITH DEEP LEARNING: A CONCEPTUAL TRIAL

FORTALEZA

2024

PAULO CLEBER FARIAS DA SILVA FILHO

LIGHT CURVE IMAGING FOR EXOPLANET DETECTION WITH DEEP LEARNING: A CONCEPTUAL TRIAL

Dissertação apresentada ao Curso de Master Degree in Physics do Post-Graduation Program in Physics do Center of Science da Federal University of Ceará, como requisito parcial à obtenção do título de mestre em Physics. Área de Concentração: Astrophysics..

Orientador: Prof. Dr. Daniel Brito de Freitas.
Coorientador: Prof. Dr. Saulo Davi Soares e Reis.

FORTALEZA

2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S582l Silva Filho, Paulo Cleber Farias da.
Light Curve Imaging for Exoplanet Detection with Deep Learning : A Conceptual Trial / Paulo Cleber Farias da Silva Filho. – 2024.
125 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Física, Fortaleza, 2024.
Orientação: Prof. Dr. Daniel Brito de Freitas.
Coorientação: Prof. Dr. Saulo Davi Soares e Reis.

1. Exoplanets. 2. Light Curve Imaging. 3. Deep Learning. 4. Convolutional Neural Networks. I. Título.
CDD 530

PAULO CLEBER FARIAS DA SILVA FILHO

LIGHT CURVE IMAGING FOR EXOPLANET DETECTION WITH DEEP LEARNING: A CONCEPTUAL TRIAL

Dissertação apresentada ao Curso de Master Degree in Physics do Post-Graduation Program in Physics do Center of Science da Federal University of Ceará, como requisito parcial à obtenção do título de mestre em Physics. Área de Concentração: Astrophysics..

Aprovada em: February 16th, 2024.

To the friendly people who played an important role on this journey.

ACKNOWLEDGEMENTS

All long-distance hikes are full of the most varied obstacles and challenges. I could not have overcome mine without the influence of many people to whom I dedicate this space to thank. Firstly, to my mother, Alice, for her support and unconditional love throughout every stage of my life. I would also like to thank my family for all their support.

I thank my advisor, Professor Daniel Brito de Freitas, for his companionship, trust and for opening the doors to the universe for me. I would like to thank Professor Saulo Davi Soares e Reis, my co-supervisor, for being one of my greatest intellectual inspirations during my undergraduate studies and, in particular, for encouraging and providing resources to carry out this work. I thank Professor Mackson Matheus França Nepomuceno for accepting to be part of the defense committee for this dissertation and Professor Luidhy Santana da Silva for accepting to be part of the defense committee for the qualification of this work.

I thank all the members of the Stellar Team, our research group, for helping to keep astrophysics in the state of Ceará alive.

I thank Sarah Gomes, Brício Warney, Laura Machado, Vasco Stascxak and Lara Hissa, who were the most present and memorable people during this master's degree period. Thank you for the scientific discussions and other moments shared.

I would like to thank Carlos Miguel, Cassimiro Albuquerque and Isabel Castro, friends who come from my undergraduate years. I would also like to thank all the people with whom I exchanged experiences throughout my master's degree. Among them, Isis do Vale, Ivo Fernandes, Júnior Alencar, Germano Sousa, Hermes Urquijo, Thais Vasconcelos, Raul Batista, Israel Silva (Koala), Willian Wallace, Genivaldo Vasconcelos, João Pedro, Wagner Coelho, Vitória Guitti, Danilo Menezes, Luana Karem and much more.

I thank Tayná Alves for all the support, companionship and teachings she gave me. I thank Luanna Sousa, Felipe Lima, Wilker Cunha and Daniel Oliveira for more than a decade of friendship and for all their support. I would like to thank Cleidson Costa, Eduardo Morais, Alexandre Laureano, Thales Wylmar, Gabriel Albano, Yury Mota and Vanessa Santos, for sharing with me one of my best experiences as a student through the national and international science fairs we participated in. I carry all these friendships with me for life.

This study was financed in part by the Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES) - Finance Code 001.

Last but not least, I thank myself for never listening to the various reasons that motivated me to give up.

“First Law of Technology: ‘Technology is neither good nor bad; nor is it neutral.’”

(*Dr. Melvin Kranzberg*)

ABSTRACT

The ever-growing number of space missions has made manual searching for exoplanet candidates infeasible due to the increasing volume of data. Consequently, the astrophysics community has extensively employed machine learning methods not only to handle the sheer amount of available data but also to enhance the sensitivity of detections concerning the signal noise inherent in relevant observational cases. This work builds upon and refines a previous review study, also presenting a conceptual trial for an alternative method to those previously discussed in the literature for classifying potential exoplanet signals using deep learning. We developed, trained, and evaluated Convolutional Neural Network (CNN) models to analyze light curves from the Kepler space mission, allowing inference on whether a given signal refers to an exoplanet or not. The distinction of this work lies in the imaging of these light curves before they are passed to the CNNs, which practically increases the number of dimensions available for analysis and enables the use of powerful and successful computer vision techniques for classification problems. Our best model ranks plausible planet signals higher than false-positive signals 97.22% of the time in our test dataset and demonstrates promising performance on entirely new data from other datasets. Our best model also shows a moderate capacity to generalize what it learned with data from other space missions, such as K2 and TESS. A good performance on entirely new data is a critical characteristic for upcoming space missions such as PLATO, and is work in progress at time of writing. Additionally, we provide new perspectives on how this imaging method can be further explored and tested in future works.

Keywords: exoplanets; light curve imaging; deep learning; convolutional neural networks.

RESUMO

O número cada vez maior de missões espaciais tornou inviável a busca manual por candidatos a exoplanetas devido ao crescente volume de dados. Consequentemente, a comunidade astrofísica tem empregado extensivamente métodos de aprendizagem automática não só para lidar com a grande quantidade de dados disponíveis, mas também para aumentar a sensibilidade das detecções relativas ao ruído do sinal inerente em casos observacionais relevantes. Este trabalho baseia-se e refina um estudo de revisão anterior, apresentando também um ensaio conceitual para um método alternativo aos discutidos anteriormente na literatura para classificação de potenciais sinais de exoplanetas usando aprendizagem profunda. Desenvolvemos, treinamos e avaliamos modelos de Rede Neural Convolucional (CNN) para analisar curvas de luz da missão espacial Kepler, permitindo inferir se um determinado sinal se refere a um exoplaneta ou não. O diferencial deste trabalho está na geração de imagens dessas curvas de luz antes de serem passadas para as CNNs, o que praticamente aumenta o número de dimensões disponíveis para análise e permite o uso de técnicas poderosas e bem-sucedidas de visão computacional para problemas de classificação. Nosso melhor modelo classifica os sinais plausíveis do planeta acima dos sinais falso-positivos 97,22% das vezes em nosso conjunto de dados de teste e demonstra um desempenho promissor em dados inteiramente novos de outros conjuntos de dados. Nosso melhor modelo também mostra uma capacidade moderada de generalizar o que aprendeu com dados de outras missões espaciais, como K2 e TESS. Um bom desempenho em dados inteiramente novos é uma característica crítica para as próximas missões espaciais, como a PLATO, e é um trabalho em andamento no momento da escrita. Além disso, fornecemos novas perspectivas sobre como este método de imagem pode ser mais explorado e testado em trabalhos futuros.

Palavras-chave: exoplanetas; imageamento de curvas de luz; deep learning; redes neurais convolucionais.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

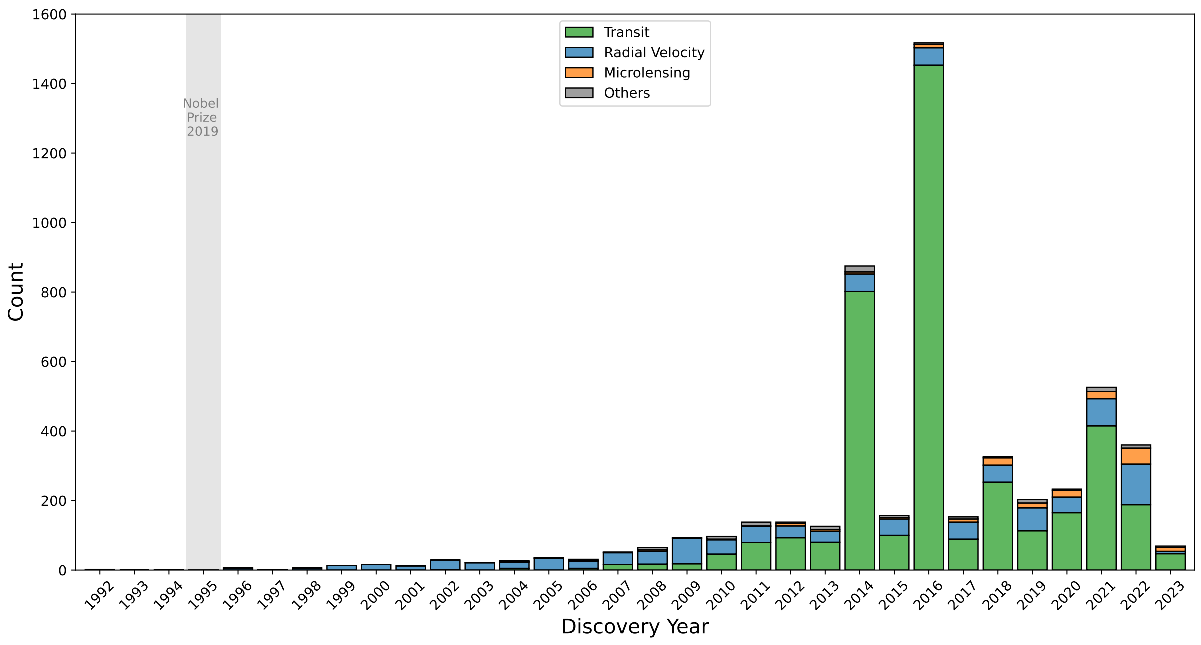
A celestial body is defined as a planet if (*I*) it orbits around a star, (*II*) has sufficient mass for its own gravity to overcome rigid body forces and maintain hydrostatic equilibrium, and (*III*) clears its orbital neighborhood (Etangs; Lissauer, 2022). Planets located outside the solar system are referred to as extrasolar planets or simply exoplanets. The study of these celestial bodies is important as it enables a better understanding of the formation and evolution of planetary systems and their components. Specifically, we can develop more accurate models to study the process of Earth's formation, as well as its current physical and chemical structure. This allows us to search for planets with characteristics similar to ours, in the quest for habitable worlds with life as we know it so far.

The concept of extrasolar planets as a serious research topic is relatively new when considering the history of physics and astronomy. A few notable milestones in the 20th century mark significant points (Greicius, 2017; Campbell *et al.*, 1988; Wolszczan; Frail, 1992), with a major breakthrough occurring in 1995 (Mayor; Queloz, 1995). This achievement, which led to the first detection of an exoplanet orbiting a star similar to the Sun, went on to win the 2019 Nobel Prize. The then-emerging field of exoplanetology made great use of the technological advancements in the following century. Earth-based and space telescopes were planned and put into operation with the main objective, or one of the main objectives, of searching for planets outside the solar system (Hellier, 2019; Boisnard; Auvergne, 2006; Borucki, 2016; Howell *et al.*, 2014; Ricker *et al.*, 2014).

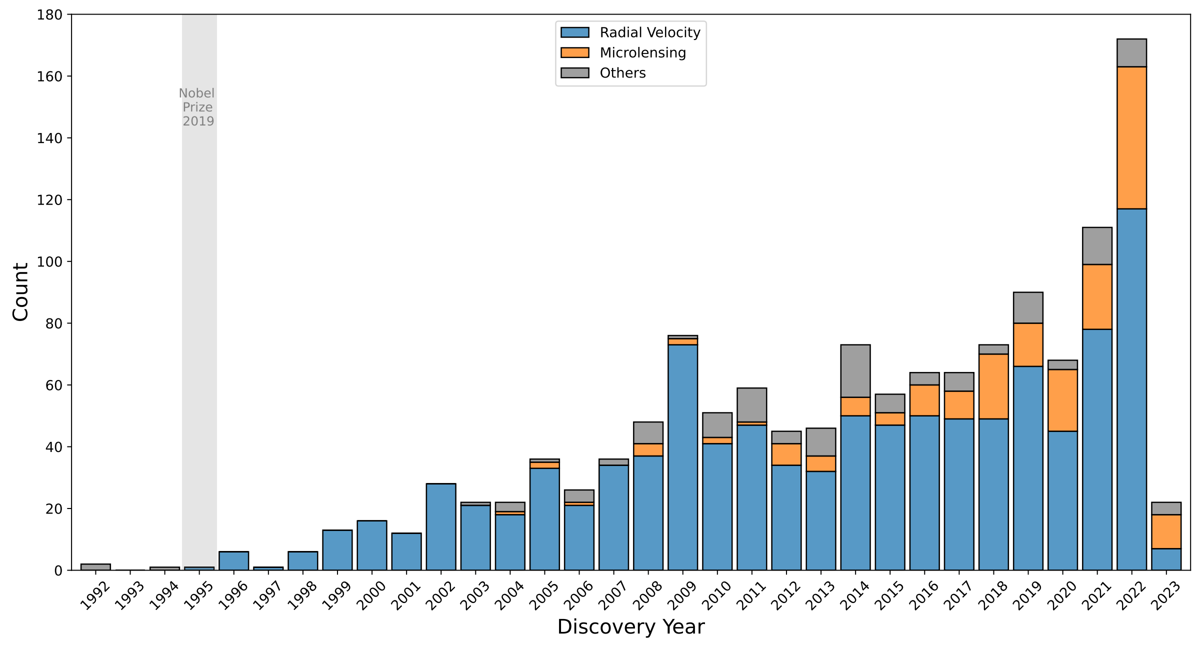
As of the writing of this text, there are exactly 5,539 confirmed exoplanets (Brennan, 2023). This number is highly dynamic, with weekly or even daily updates being common during certain periods. Figure 1 illustrates the quantity of confirmed exoplanets over the years using different detection methods. Additionally, there are more than 9,400 planet candidates, which is also a significant number. Data from space telescopes contribute to the majority of these figures. Specifically, up until its last mission conclusion in October 2018, the Kepler space telescope was responsible for discovering 2,662 planets throughout its two missions, known as Kepler and K2, over nine years of operation (NASA, 2018b). Today, that number has reached 3,326 (NASA, 2023).

Figure 1 – Number of exoplanets discovered over the years and their respective detection methods

(a) General data



(b) Data without the planetary transit method



Note: Up to 17, April 2023.
Source: Self elaboration. Data from *NASA Exoplanet Archive*, available at https://exoplanetarchive.ipac.caltech.edu. Accessed 17 April 2023.

One of the conclusions drawn from population studies based on data from the Kepler telescope is that planets outnumber stars in our galaxy (NASA, 2018a). Small planets between the sizes of Earth and Neptune, which are more difficult to detect, substantially outnumber Jupiter-sized planets. (Howard, 2013). For comparison, the *Global Astrometric Interferometer* *for Astrophysics* (Gaia) space telescope currently collects data from over 1 billion stars (Vallenari *et al.*, 2022). These numbers highlight a significant challenge as the scientific community has not been able to fully digest all the information from completed missions, such as Kepler, or ongoing missions such as the *Transiting Exoplanet Survey Satellite* (TESS) (NASA, 2019). In this decade alone, for example, we will witness the launch of another space telescope, *PLAnetary Transits and Oscillations of stars* (PLATO), which promises to provide an even larger and more precise volume of data on exoplanets than its predecessors (Rauer; Heras, 2018).

Given this scenario, the scientific community has begun to explore ways of cataloging exoplanets in a more automated manner. The Robovetter project, for instance, has successfully utilized decision trees to automate the process of filtering out false exoplanet signals, mimicking human judgment (Coughlin *et al.*, 2016). Similarly, the Autovetter project has leveraged machine learning techniques, specifically a random forest model, to classify data related to exoplanets (McCauliff *et al.*, 2015).

The AstroNet was the first deep learning model used to detect an exoplanet (Shallue; Vanderburg, 2018). It is an algorithm that uses a deep neural network to classify exoplanet data. This work gained prominence by discovering the first planetary system with eight planets, equating to our own Solar System. Originally developed to work with data from the Kepler mission, AstroNet was modified to work with the K2 mission data (Dattilo *et al.*, 2019) and further adapted to handle data from TESS (Yu *et al.*, 2019). Following the benefits of deep learning usage, GPC (Armstrong *et al.*, 2021) was able to validate 50 new exoplanets and more recently ExoMiner (Valizadegan *et al.*, 2022) achieved the incredible mark of 301 new exoplanets validations.

The models mentioned work based on the exoplanet detection method known as planetary transit, which essentially consists of observing drops in the brightness of a target star caused by the transit of a planet. As shown in Figure 1, this method is the most successful so far, and we will delve into its details in the next chapter. Some members involved in AstroNet were also part of another project that uses the same type of neural network to work with an appropriate approach to another detection method: radial velocity, which involves observing small oscillations in the motion of a target star caused by gravitational interaction of the planet-star system in question (de Beurs *et al.*, 2022). This method is the second most successful in the literature and will also be briefly discussed in the next chapter. Both of these mentioned methods analyze data that can be characterized as time series. In the case of the transit method, these series are called light curves.

Numerous other research groups have proposed implementing their own machine learning or deep learning models to enhance the search for exoplanets. Some are based on the approach of AstroNet itself, such as the algorithm Nigraha algorithm (Rao *et al.*, 2021). Others continue to analyze radial velocity data, such as the ExoplANNET algorithm (Nieto; Díaz, 2023).

In this work, we present a general review of the main methods for detecting exoplanets and show how machine learning and deep learning techniques can be applied in their search. As a conceptual test, we present and incorporate techniques of temporal series imaging for the construction of a robust processing pipeline for light curves. In brief, imaging methods transform time series, that can be described by one-dimensional arrays, into images or two-dimensional arrays (Wang; Oates, 2015). The advantage of this is that powerful and successful techniques from deep learning and computer vision can be used to classify signals in this data as planets or non-planets. In particular, we will make use of *Convolutional Neural Network* (CNN) models to evaluate if the time series imaging technique is viable and helpful on augment the performance of deep learning models. To the best of our knowledge, this methodology has not yet been explored by the astrophysics community.

The structure of this dissertation is presented as follows. First, in Chapter 2, we will give a brief introduction on the main exoplanet detection methods in order to understand how they influence space missions for the observation of targets of interest. We will delve deeper into the details of the Kepler space mission in order to understand the structure of the data worked on and how they are normally processed and evaluated by the astrophysical community. In Chapter 3, we will introduce some of the general concepts and methods of machine learning and deep learning. With this, we will show some possibilities for constructing the classification models used in this work and how they can be implemented. In Chapter 4, we will discuss the method of temporal series imaging, how it is being implemented in this work, and how we can reduce its products size while preserving the relevant information. In Chapter 5 we explain the models architectures and settings used in this work and evaluate their performance metrics on a portion of our original dataset and on a sample of targets from K2 and TESS missions. Finally, in Chapter 6, we make an overview of the whole work and show some perspectives with the results we achieved.

2 EXOPLANET DETECTION

Numerous exoplanets with various characteristics have been detected and confirmed over the years. To achieve this, astrophysicists employ different detection methods that take into account the dynamics of physical properties of the target star-planet system. Table 1 presents all the methods found in the literature and their contributions to official confirmations. Among these, the transit method, radial velocity, and gravitational microlensing stand out.

Table 1 – Count of planets confirmed by each detection method

Method	Count
Transit	3985
Radial Velocity	1032
Microlensing	187
Imaging	65
Transit Timing Variations	25
Eclipse Timing Variations	17
Orbital Brightness Modulation	9
Pulsar Timing	7
Astrometry	2
Pulsation Timing Variations	2
Disk Kinematics	1

Note: Data updated up to the date of access.
Source: NASA Exoplanet Archive, available at https://exoplanetarchive.ipac.caltech.edu. Accessed 17 April 2023.

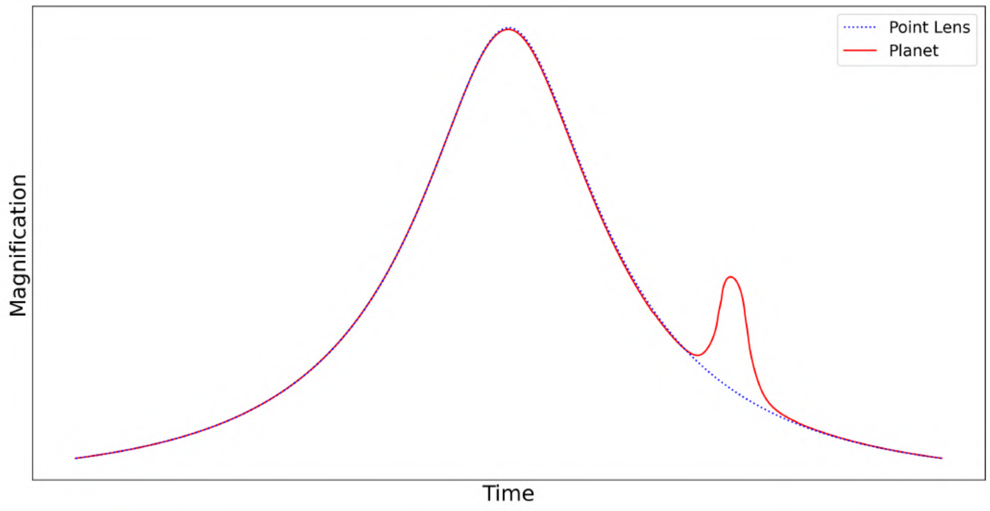
2.1 Physical Characteristics of the Main Methods

The following subsections focus on discussing the physical characteristics of the three main methods, with particular emphasis on the transit method, which is used in this work. The gravitational microlensing and radial velocity methods are briefly presented to complement the discussion on the implementation of temporal series imaging. Understanding these concepts is crucial for the analyses conducted throughout the rest of this work.

2.1.1 *The Gravitational Microlensing Method*

Einstein's theory of general relativity predicts that massive objects warp the fabric of spacetime. As a consequence, a beam of light traveling in a straight line on an undeformed portion of this tissue suffers a deviation in its path when passing close to a massive object. When a star passes in front of a more distant one in relation to the line of sight, the more distant one appears to be brighter than normal for a short period of time due to the light deflected by the gravitational effect. When the nearest star has an exoplanet orbiting it, we can observe a secondary peak of the luminosity observed by the telescope, as given in figure Figure 2.

Figure 2 – Example of a typical graph generated by gravitational microlensing observations



Note: Gravitational microlensing event model. The solid line expresses the behavior of the graph generated by observing a microlensing event where the closest star to the observer has a planet orbiting it. The dotted line represents the same situation, but disregarding the planet.
Source: Self-made using MulensModel software (Poleski; Yee, 2019).

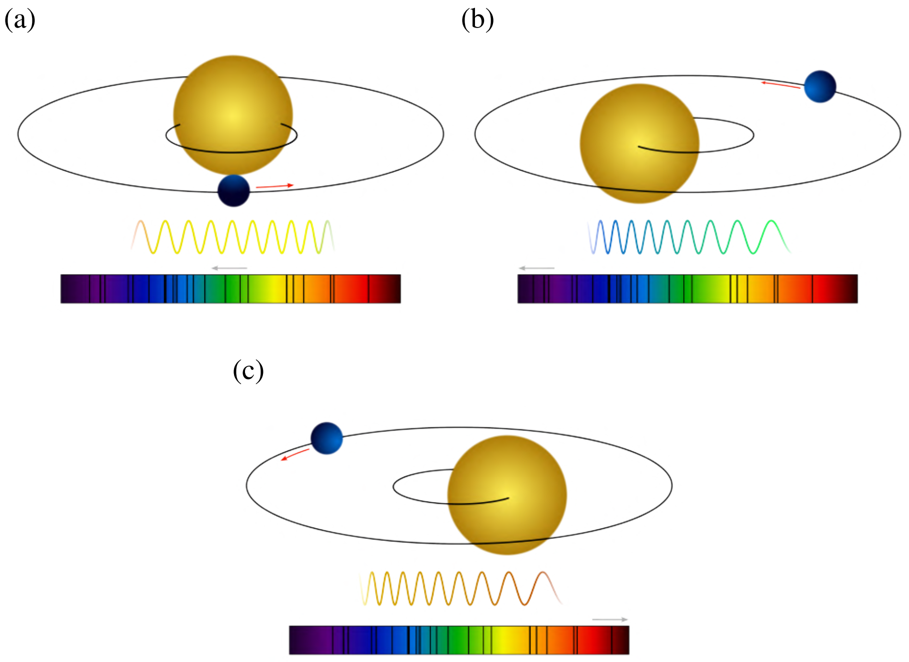
We can use machine learning to investigate whether this secondary peak is caused by a planet or some other type of perturbation. This is interesting because often this analysis is challenging, both due to the inherent noise in the measurements and other factors in the observation itself. As a time series, the imaging method presented in this work can be used in contrast to more conventional methods. The reader may refer to chapter 5 of Perryman (2018) for details on the microlensing method.

2.1.2 *The Radial Velocity Method*

The presence of a planet generates periodic disturbances in the motion of a star due to the gravitational interaction present. Consider a two-body system consisting of a star and a planet. Through the theory of the problem of central force for two bodies, considering Newton's law of universal gravitation, we can see that both planet and star orbit the barycenter of the system. Each body moves in a closed elliptical orbit, with the center of mass at one of the foci. As the difference in mass between the star and the planet is very large, this effect is very small, but still appreciable.

Relative to an observer's line of sight, the star moves away as the planet approaches and vice versa. From a spectroscopic point of view, this movement generates periodic dynamics in the star's spectral lines, where it is possible to observe Doppler shifts. A representation of this can be seen in Figure 3.

Figure 3 – Example of observing the spectral dynamics of a star

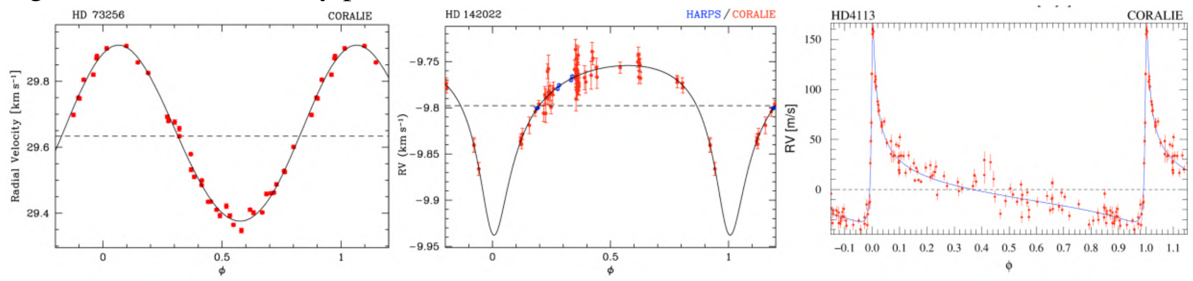


Note: Considering the observer in the same perspective as the reader, when the star approaches and the planet moves away, we observe a deviation of the spectral lines to the left (blueshift). When the planet approaches and the star moves away, we observe a deviation of the spectral lines to the right (redshift). Elements out of scale.
Source: Self elaboration.

Through the amplitude of the oscillation, it is possible to calculate the *Radial Velocity* (RV) of the star and estimate parameters such as the orbital period, the mass, among others. The radial velocity method is also known in the literature as *Doppler Spectroscopy*. We can generate theoretical profiles for radial velocity curves, which can help us in fitting the measured data. Some examples can be seen in Figure 4.

The analysis of data measured by radial velocity should take stellar activity into account. Factors such as stellar oscillation, granulation, and short-term and long-term stellar

Figure 4 – Radial velocity profiles for stars HD 73256, HD 142022 and HD 4113



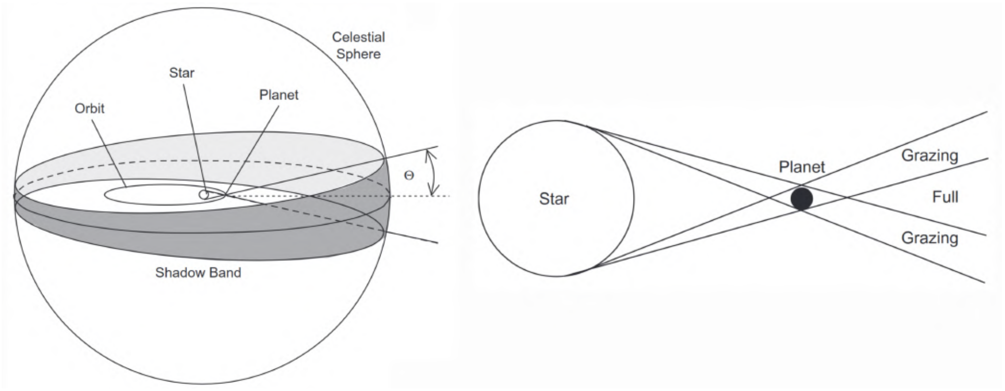
Source: Extracted from Udry *et al.* (2003), Eggenberger *et al.* (2006) and Tamuz *et al.* (2008), respectively.

activities can significantly reduce the accuracy of the measurements. The reader may refer to chapter 2 of Perryman (2018) for details on the radial velocity method. Machine learning algorithms can analyze this data in order to eliminate or reduce the influence of these effects and facilitate planet detection. As this data constitutes a time series, the application of an imaging method to enhance the analysis might be helpful in this context.

2.1.3 The Planetary Transit Method

Similarly to the gravitational microlensing method, we also observe the brightness of the star over time in the planetary transit method. However, in this case, we do not rely on another star but rather on the star that constitutes the planetary system of interest. Due to stellar activities and other external factors, fluctuations in the data are common. In a situation where a planet orbits a star and the inclination of the system's orbit is favorable for observation, as illustrated in Figure 5, we can measure a decrease in brightness of the star due to the passage of the planet in front of it. In other words, as the planet transits in front of the star relative to the line of sight, it blocks a portion of the light reaching the telescope.

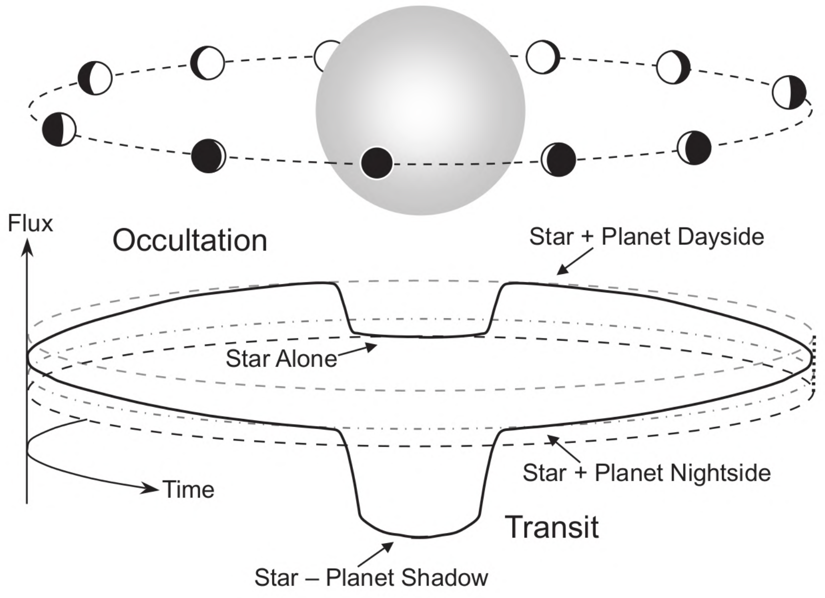
Figure 5 – Angular region where the transit can be observed



Source: Adapted from Seager (2010).

Figure 6 illustrates the photometric influence of the planet on the star's observed brightness. It is depicted in a circular shape to illustrate the cyclical nature of the phenomenon. There are two main situations occurring. The first one is the already described planetary transit, where the planet blocks part of the starlight as it passes between the star and the observer. This results in a primary drop in the light flux. The second occurs when the planet is behind the star in relation to the observer, where it is also possible to observe a drop in the flux, the secondary drop or planetary occultation. It occurs because the flux of light received by the telescope is greater than the flux of the star alone due to the reflection of light from the star itself on the surface of the planet. The illuminated side of the planet, or the "day side", will impact the increase in flux observed by the telescope depending on the planet's albedo (reflection coefficient) (Mallama, 2017).

Figure 6 – Scheme of a planetary transit and occultation

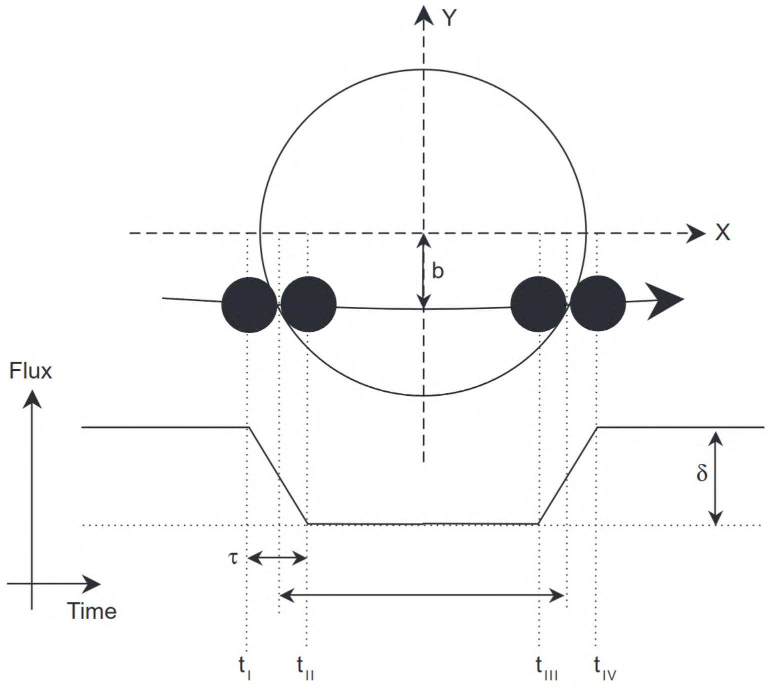


Source: Extracted from Seager (2010).

Due to the translational motion, the light flux as a function of time considered in this example is periodic. The time interval between one primary transit and its neighboring transit (or any reference point with its corresponding neighbor) is called the orbital period. Additionally, the transit event is characterized by four time instants, as illustrated in Figure 7.

These times occur at the instants when the planet's disk touches the star's disk, called contact points. The contact points t_{II} and t_{III} do not occur when the planet is seen in the penumbra region. The total transit duration is given by $T_{tot} = t_{IV} - t_I$. The duration of the complete transit, i.e., the time interval in which the planet's disk is completely contained in

Figure 7 – Characteristic instants of a planetary transit



Source: Extracted from Seager (2010).

the star's disk, is given by $T_{full} = t_{III} - t_{II}$. Planet's ingress occurs when the planet's disk is entering the star's disk region and its duration is given by $\tau_{ing} = t_{II} - t_I$. Similarly, planet's egress occurs when the planet's disk is leaving the star's disk region and its duration is given by $\tau_{egr} = t_{IV} - t_{III}$. In general, considering eccentric orbits, $\tau_{ing} \neq \tau_{egr}$. Regarding the flux decrease δ, for transits we have that

$$\delta_{tra} \approx k^2 \left[1 - \frac{I_p(t_{tra})}{I_s}\right], \tag{2.1}$$

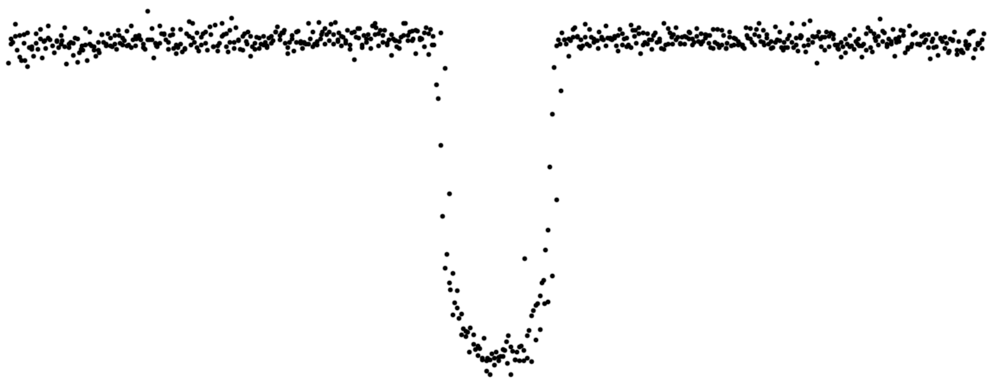
being usual to approximate $\delta_{tra} \approx k^2$. In this expression, $k = R_p/R_s$ is the ratio of the radius of the planet to that of the star, $I_p(t_{tra})$ and I_s are the disk-averaged intensities of the planet during transit and the star, respectively. The reader can refer to chapter 2 of Seager (2010) and chapter 6 of Perryman (2018) for more details.

The trapezoidal approximation of the transit illustrated in Figure 7 assumes a linear variation of flux with respect to time. However, this is not what is actually observed due to the non-uniform motion of both the star and the planet discs, as the overlapping area does not vary linearly over time, and because a real stellar disc does not have a uniform density. This last point is essentially the effect known as limb darkening[1], where the center of the star is brighter than the edges. This results in a drop in flux greater than k^2 when the planet is close to the center and

[1]Latin, "*limbus*" means edge.

less than k^2 when close to the edges of the stellar disk. Because of this, the bottom of the graph given in Figure 7 becomes more rounded and, therefore, the points of contact t_{II} and t_{III} are obstructed. Limb darkening occurs due to the variation in temperature and opacity with height in the star's atmosphere. An actual example of a planetary transit graph is shown in Figure 8.

Figure 8 – example of an observed transit.



Source: Self elaboration.

We can see that this light curve has some noise, but we can easily observe a similar drop as seen in Figure 7 when taking into account the aforementioned effects, which smooth out the bottom of the graph. In most cases, this pattern is not easily noticeable. This particular drop represents a blocking of only 0.8% of the star's flux by the planet. The most challenging cases for the transit method occur when the planet's radius is very small compared to that of the star. This is because, according to equation 2.1, the smaller the planet's radius, the smaller the drop in flux. This can cause the planet's signal to be overshadowed by the noise from the observation. The aim of this work is to overcome this difficulty through the imaging method, enabling deep learning models to better recognize transits even in noisy data.

2.2 Photometry Data for the Transit Method

After providing a general context of the topic and understanding the nature of the problem we aim to solve, the next step in this work is to obtain, explore, and prepare the data for training the algorithm. All data produced by Kepler for the Autovetter catalog were downloaded from the Mikulski Archive for Space Telescopes (MAST). This section will focus on the technical understanding of the observation instrument so that we can properly explore and work with the data. We also make a contrast on how manual classification of planets are made.

2.2.1 The Kepler Mission Data

The Kepler mission, initiated in March 2009, was developed to determine the frequency of planets with sizes similar to Earth that are in the habitable zone of stars similar to the Sun (Borucki *et al.*, 2010). The habitable zone is a region around a star where liquid water can exist on the surface of rocky planets for an extended period of time (Gonzalez *et al.*, 2001). For this purpose, the telescope spent four years observing brightness variations of approximately 160,000 target stars in a fixed region near the Cygnus and Lyra constellations[2] (Thompson *et al.*, 2016).

Raw data, directly obtained from the telescope, are rarely ready to be used for their intended purposes. Therefore, they need to undergo a series of specific operations, collectively known as a data pipeline (Quemy, 2019). The raw data from Kepler are grouped quarterly into packages called quarters, referenced as Q0, Q1, Q2, and so on. They are then unpacked and grouped by cadence and pixel type (Jenkins *et al.*, 2010c). The exception to this pattern is the first two quarters. The first quarter, Q0, refers to the first 10 days of the mission. Q1 refers to approximately the 34 days following Q0. All other quarters span around 90 days.

Cadence refers to the interval at which the data is collected. In the case of Kepler, there are long cadence and short cadence data. Long cadence is characterized by 29.4-minute intervals (Jenkins *et al.*, 2010a), while short cadence, widely used in asteroseismology studies, is characterized by approximately 1-minute intervals (Gilliland *et al.*, 2010). In this work, we will focus on long cadence data. Additionally, the pixels in each collected image can be of three types: target pixels, background pixels, or collateral pixels used for calibration purposes.

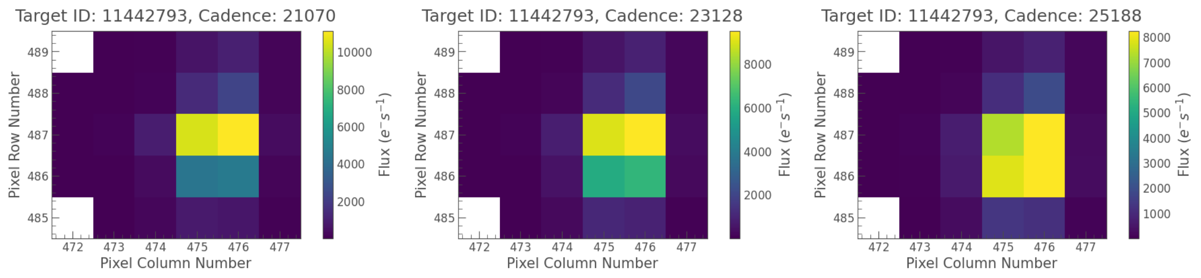
Once separated, these data are processed by the *Kepler Science Processing Pipeline* and subsequently archived. This pipeline is responsible, among other things, for providing calibrated pixels and corrected photometric flux data (Jenkins *et al.*, 2010b). After this processing, the data is finally stored in files in the *Flexible Image Transport System* (FITS) format, which is the standard for astronomical data. Due to enhancements in this pipeline, raw data is processed and archived multiple times, with each processing uniquely classified with a *Data Release* (DR) number.

As a product of this pipeline, we have processed *Target Pixel Files* (TPFs) and light curves, as well as the raw data itself, which are useful for some more specific treatments. The TPFs are also used in the K2 and TESS missions, and each of them consists of a set of images

[2]Figure 77 illustrates the field of view of Kepler.

and other data related to a target star (Barclay, 2023). Naturally, there is an image for each cadence, as shown in Figure 9, generated with the help of the *Lightkurve* package in Python (Lightkurve Collaboration *et al.*, 2018).

Figure 9 – Images generated from the TPF of the star Kepler-90's Q6.



Note: The time interval between one cadence and another is 30 minutes. The images above show the variation of the star's flux over approximately 2.3 hours.
Source: Self elaboration.

In this image, it is only possible to see the brightness of the pixels from the target star, as the background brightness is already removed in the Kepler pipeline. The background brightness typically consists of *zodiacal light*, which is the light from stars and other luminous sources reflected by cosmic dust, sometimes hindering the visualization of small flux variations (Morris *et al.*, 2020). The background is then cleaned, but its original data is also stored in the FITS file and can be accessed for analysis when pertinent.

From the TPF of a star, it is possible to extract information about its light flux as a function time[3]. This a relation called is called a *light curve* and is generated using aperture photometry techniques (Cleve *et al.*, 2016). The most common technique is the *Simple Aperture Photometry* (SAP), which involves selecting the pixels that maximize the *Signal-to-Noise Ratio* (SNR) of the target and summing them to generate a single point corresponding to the light flux at a given moment in time. The ordered set of these points forms a one-dimensional light curve (Bryson *et al.*, 2010). The signal-to-noise ratio, as the name suggests, is the power of the signal (P_S) divided by the power of its inherent noise (P_R) (Johnson, 2006), given by (DÍAZ *et al.*, 2014):

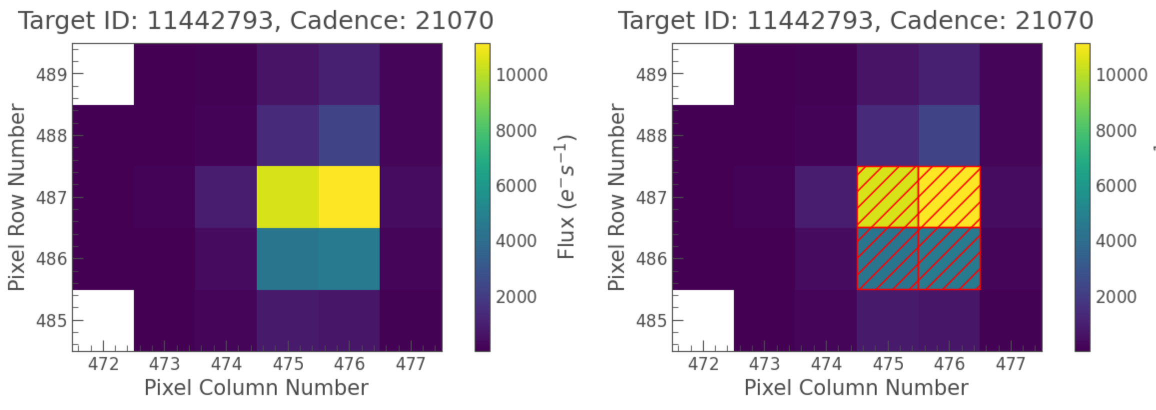
$$\text{SNR} = \frac{P_S}{P_R} = \frac{\delta}{\sigma}\sqrt{N_t}, \tag{2.2}$$

where δ is the transit depth, σ is the standard deviation of out-of-transit observations and N_t is the number of in-transit data-points.

[3]Note, however, that the flux unit $e^{-}s^{-1}$ indicated in Figure 9 is more commonly associated with the measurement of electron flux or the rate of incoming electrons rather than light flux. Light flux is typically measured in

The selection of optimal pixels is performed by the Kepler pipeline itself through an aperture mask and is stored in the FITS file. However, it is possible to use any previously created mask. Figure 10 shows an example of selecting the best pixels for a given cadence using the mask created by the Kepler pipeline.

Figure 10 – Example of selecting the pixels that optimize the star's SNR



Source: Self elaboration.

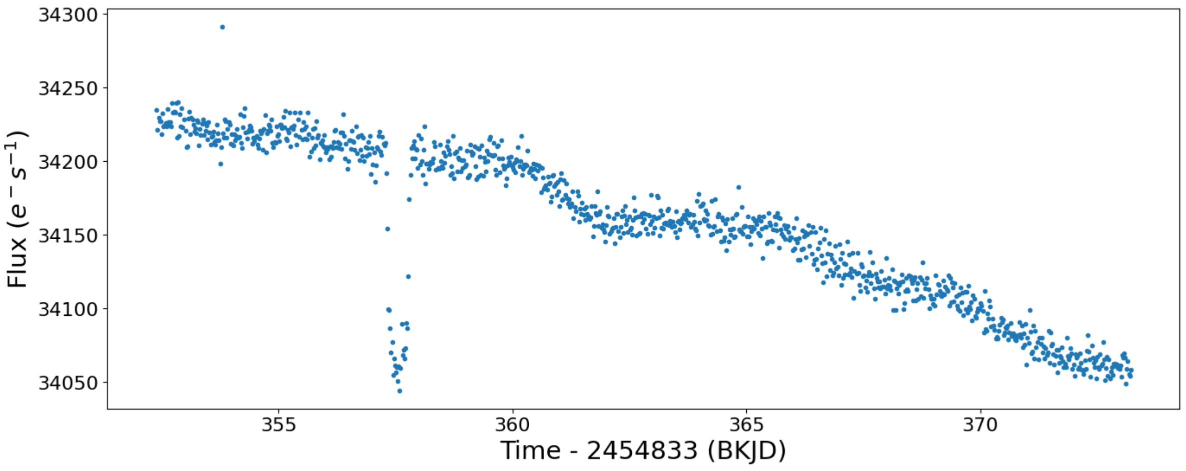
This process is performed for all cadences in the file (both for the fluxes and their errors and other measurement parameters), and the results are stored in a table. It is possible to generate plots like the one in Figure 11 using the generated flux and time data. The time is displayed on the *Barycentric Kepler Julian Date* (BKJD) scale, which is a unit derived from *Barycentric Julian Date* (BJD) (Eastman *et al.*, 2010; Cleve; Caldwell, 2016), and it is the reference time used in counting Julian days with corrections for the Earth's position relative to the barycenter of the Solar System, minus a constant of 2,454,833 days. This constant corresponds to 12:00 on January 1, 2009, and was defined this way to allow the satellite to use less memory when processing time values.

The light curve in Figure 11 exhibits a decreasing trend caused by systematic errors stemming from various astrophysical or instrumental sources (Hippke; Angerhausen, 2015; Twicken *et al.*, 2010). A common method, among several in literature, to address this issue is using the Savitzky-Golay filter (Savitzky; Golay, 1964), which removes these undesired trends while preserving important features of the curve that will be discussed in the next section. This filter is even employed in the Kepler pipeline itself, specifically in the component called

units like photons per second, watts, or other related measures. For astrophotometry, the concept of electron flux can be important when dealing with detectors, such as *Charge-Coupled Devices* (CCDs), which are commonly used in astronomy to detect and measure the intensity of light. The rate of incoming electrons on the detector is related to the amount of light received (Mackay, 1986). From hereon, we will follow the literature standards and use only the term *flux* instead of light flux or electron flux, having this observation in mind.

Presearch Data Conditioning (PDC) (Twicken *et al.*, 2010).

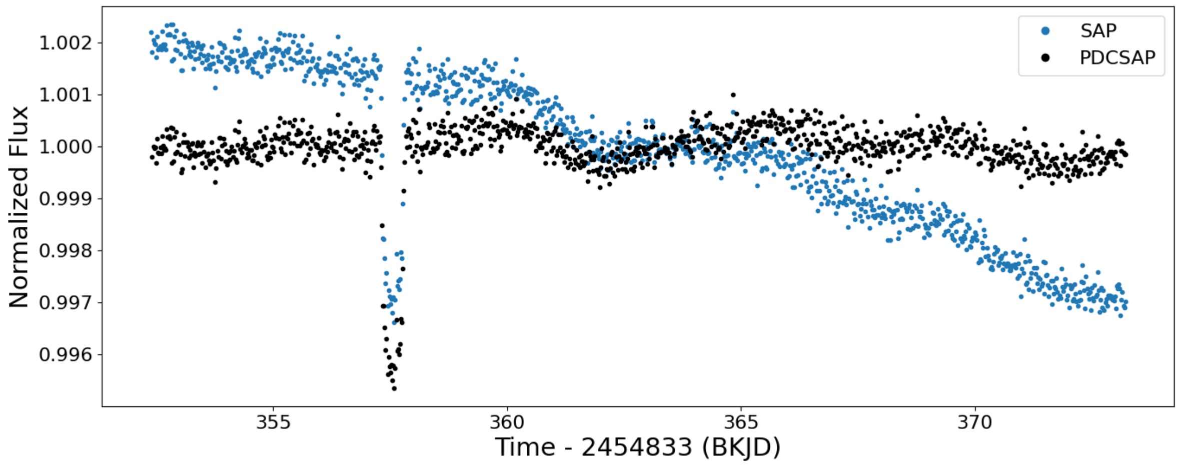
Figure 11 – Example of light curve generated by SAP data for Quarter 4 of Kepler 90



Source: Self elaboration.

The pipeline-processed flux is labeled as PDCSAP, and an example can be seen in Figure 12. However, Hippke *et al.* (2019) points out that the Savitzky-Golay filter misses 16 out of every 100 planet signals (which will be discussed shortly) in their test set. Compared to the performance of other types of filters, the Savitzky-Golay filter is not excellent for this work. Instead, the filtering method used here is based on splines, described in Section 2.3 of Vanderburg and Johnson (2014) and time-windowed sliders (Hippke *et al.*, 2019).

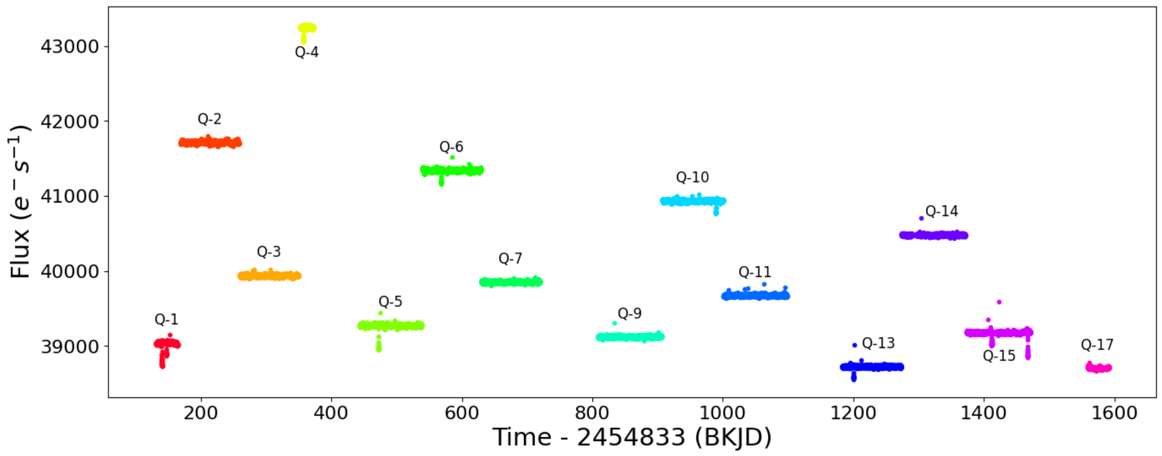
Figure 12 – Comparison between SAP and PDCSAP data for Quarter 4 of Kepler 90



Note: Both have been normalized to be viewed at the same scale.
Source: Self elaboration.

All of this processing was done for just one of the star's quarters. By repeating this filtering process for all quarters and combining them into a single graph, we obtain the result shown in Figure 13. The reason why each quarter is on a different scale is due to the approximately 90° rotation that Kepler performed to keep its solar panels facing the Sun when necessary, without changing its field of view. This caused the star to be observed in a different area of the telescope's focal plane, and the variation in flux magnitude from one quarter to another occurs because each area of this plane generally had a different photometric sensitivity compared to the others (Cleve; Caldwell, 2016).

Figure 13 – All available quarters from the star Kepler-90 individually filtered

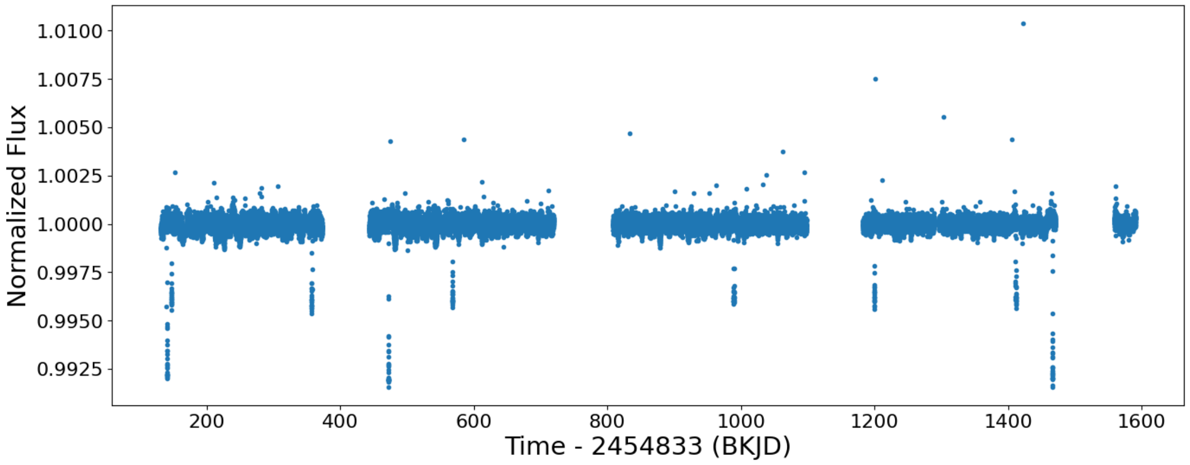


Source: Self elaboration.

This problem can be solved by normalizing the flux of each individual quarter. Figure 14 shows the normalized light curve. This light curve summarizes all the observations of the Kepler-90 star made by the telescope throughout the Kepler mission. Some discontinuities (or gaps) in the curve are observed and have various causes. One of the most common types of gap is related to the monthly data download, where the telescope changes its orientation to point its *High Gain Antenna* (HGA) towards Earth. This lasts for approximately one day, and the telescope does not collect data during this period. Temperature variations in the telescope's components also cause gaps (Cleve; Caldwell, 2016). Missing measurements naturally also result in gaps and can occur for various reasons. In addition, data with low photometric precision is discarded by the Kepler pipeline and replaced with NaNs (Cleve; Caldwell, 2016; Pascual-Granado *et al.*, 2015). There is a space in the FITS file dedicated exclusively to the quality specifications of the measurements.

Based on this normalization, data points far from the value 1 on the ordinate axis can characterize outliers, which are values that notably deviate from the others (Grubbs, 1969). Considering that the transit method essentially consists of observing flux drops, only the upper outliers of the light curve were removed in order to preserve the transits. The chosen value for

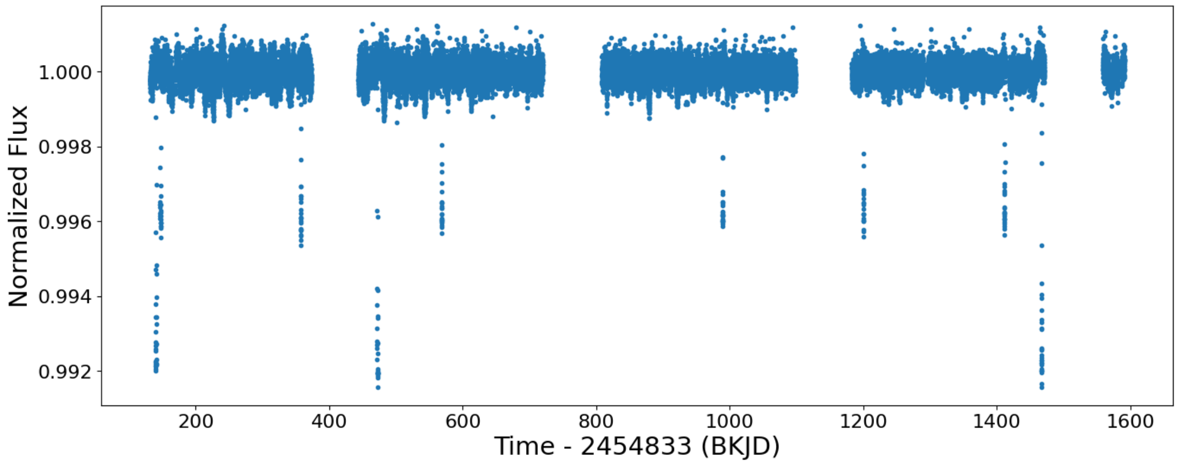
Figure 14 – Complete light curve of the star Kepler-90



Source: Self elaboration.

this removal was 3σ, and the result can be seen in Figure 15.

Figure 15 – Kepler-90 star light curve without outliers.



Source: Self elaboration.

2.2.2 Finding Periodic Signals in Light Curves

As discussed in subsection 2.1.3, planetary transit is a periodic event. Once the light curve has been processed up to the state shown in Figure 15, we are able to search for periodic signals within it. Specifically, these signals should exhibit a characteristic decrease in flux. This decrease, in turn, should have similar characteristics to those illustrated in Figure 7. We have seen that the time interval between flux drops is referred to as the *orbital period*. Such a drop also has a *depth* and a *duration*. Regarding an isolated transit, we refer to the temporal reference where the first occurrence of this transit in our light curve happens or *should* happen as an *epoch*. For later convenience, we will use t_0 instead of “epoch”. Thus, the t_0 of the deepest transit in

Figure 15, for example, is approximately 140 BKJD. In that figure, we easily observe sharp drops in the flux. However, some drops are larger than others. This is because they are signatures of two distinct planets, Kepler-90g and Kepler-90h.

All these properties are used to characterize a planetary transit. A periodic signal detected by the Kepler pipeline in the light curve, such as drops that have transit-like characteristics, are referred to as *Threshold Crossing Event*s (TCEs) (Batalha; others, 2010). However, not every TCE in the light curve indicates an exoplanet. In fact, the *modus operandi* for validating exoplanets in this way is to evaluate all other astrophysical and instrumental possibilities that could explain such behavior in the flux. If nothing other than an exoplanet can explain a given TCE, then validation is achieved. There are many programs that perform automated searches for TCEs, such as VARTOOLS (HARTMAN; BAKOS, 2016) and AstroPy (ASTROPY COLLABORATION *et al.*, 2018). Here, we will understand the general idea of this process.

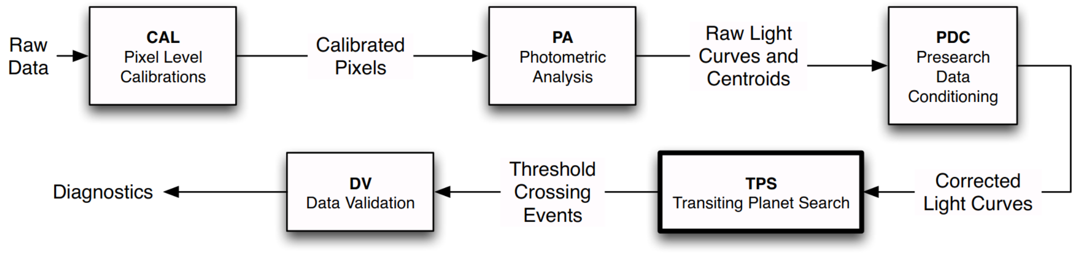
In the case of Figure 15, a qualitative identification of periodic signals is immediately visible to the naked eye for two of the eight planets orbiting Kepler-90. Nevertheless, considering the cadence of measurements, a single drop often does not precisely show us the shape of the transit due to the relatively limited number of observations throughout the event, especially if its duration is short. Therefore, we can use the *phase folding* technique from light curve analysis (Gregory; Loredo, 1992), as explained in the paragraphs preceding Figure 21.

So far, we considered the standard procedure of an astronomer dealing with partially processed data from the *Kepler Science Operation Center (SOC) Science Processing Pipeline*. It is instructive to have a glance at it before working with any further examples. Figure 16 shows a representation of the data flow for this pipeline. As Jenkins *et al.* (2010c) points out:

> Several processing steps must be completed before TPS, which lies at the heart of the Pipeline, can perform its search for signatures of transiting planets. First, CAL calibrates the raw pixels downlinked from the spacecraft to remove on-chip artifacts and to place the measurements on a linear scale with estimated uncertainties. Second, PA identifies and removes cosmic rays from the pixel time series, estimates and subtracts the background flux and then sums the resulting pixel values over the photometric aperture underneath each target star image. Third, PDC identifies and removes signatures of systematic effects in the photometric time series, such as changes in pointing or focus, and fills any gaps to condition the time series for TPS. TPS then searches the corrected flux time series for signatures of periodic pulse trains indicative of transiting planets. Threshold-crossing events flagged by TPS are examined in detail by DV to establish or break confidence in the transit-like features as planetary signatures.

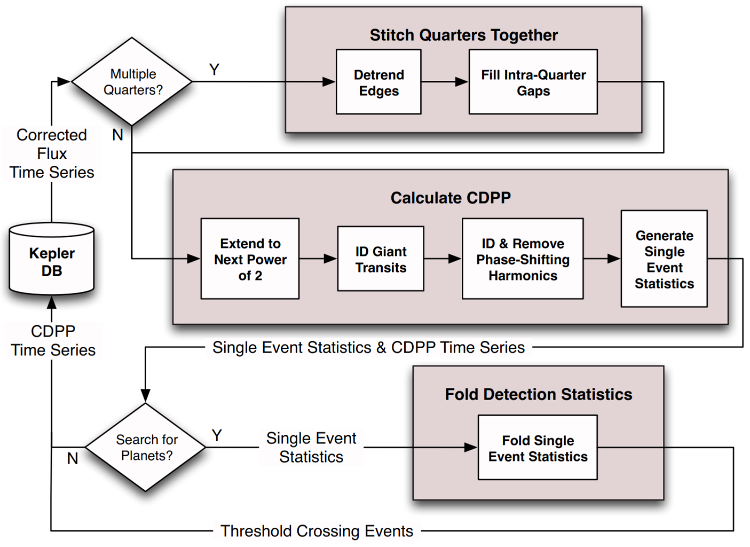
The *Transiting Planet Search* (TPS) of Kepler's pipeline, as illustrated in Figure 17, is described in details in Jenkins (2002) or, more briefly, in section 2 of Tenenbaum and others (2012).

Figure 16 – Data flow diagram for the SOC Science Processing Pipeline



Source: Extracted from Jenkins *et al.* (2010c).

Figure 17 – Block diagram for TPS

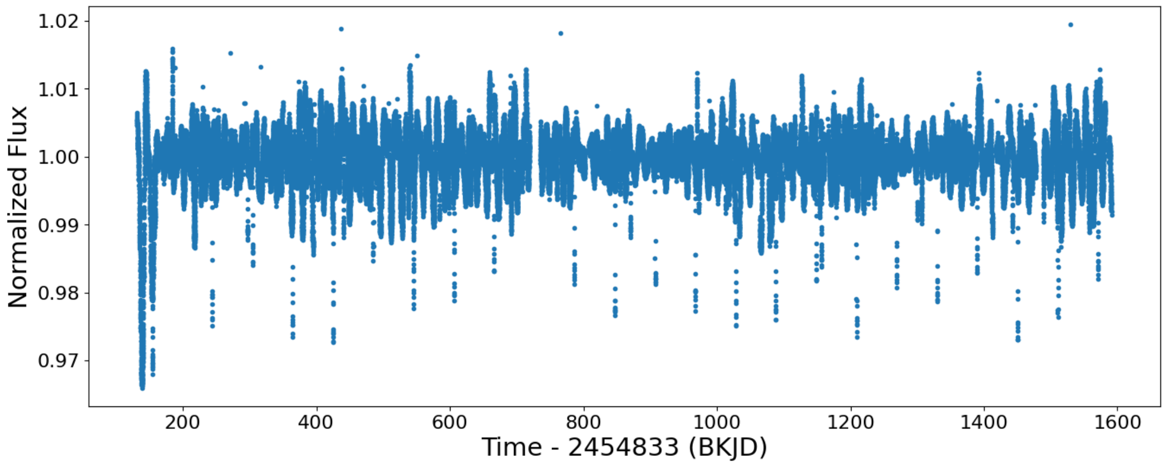

Multiple Quarters?
Y
N
Stitch Quarters Together
Detrend Edges
Fill Intra-Quarter Gaps
Corrected Flux Time Series
Kepler DB
CDPP Time Series
Calculate CDPP
Extend to Next Power of 2
ID Giant Transits
ID & Remove Phase-Shifting Harmonics
Generate Single Event Statistics
Single Event Statistics & CDPP Time Series
Search for Planets?
Y
N
Single Event Statistics
Fold Detection Statistics
Fold Single Event Statistics
Threshold Crossing Events

Source: Extracted from Jenkins *et al.* (2010c).

In summary, it decomposes the light curve into its frequency components using a wavelet technique (Daubechies, 1988) to search for individual transit-like signals over a range of transit durations from 1 to 12 hours. The time-varying noise in each frequency band is computed and a *Single Event Statistic* (SES) time series is extracted. For each sample in the original flux time series, this new time series describes the significance of the detection of the signal. The search is performed repeatedly using a discrete set of different model transit and a SES time series is produced for each model transit used in the search. Then, this results are used to search transit-like features that are periodic by folding the SES across time into *Multiple Event Statistics* (MES). At each tested transit period, the search analyses a set of possible phases and a vector of MES versus period is yield. The final step is to remove from further analysis the signals that do not satisfy certain criteria. The most important one is to cut any star for which all of the MES for all periods and all pulse durations fall below 7.1σ. This values fits a good trade-off between the number of expected false-positive detections due to pure statistical fluctuations and the sensitivity to Earth analogs (Jenkins *et al.*, 2002).

The process above illustrates how *planet candidates* are classified in the Kepler pipeline. In practice, to study this candidates or to search for new ones the pipeline might have missed, let's follow the example of the light curve from Kepler-30 star, treated as explained in section 2.2.1 and presented in Figure 18.

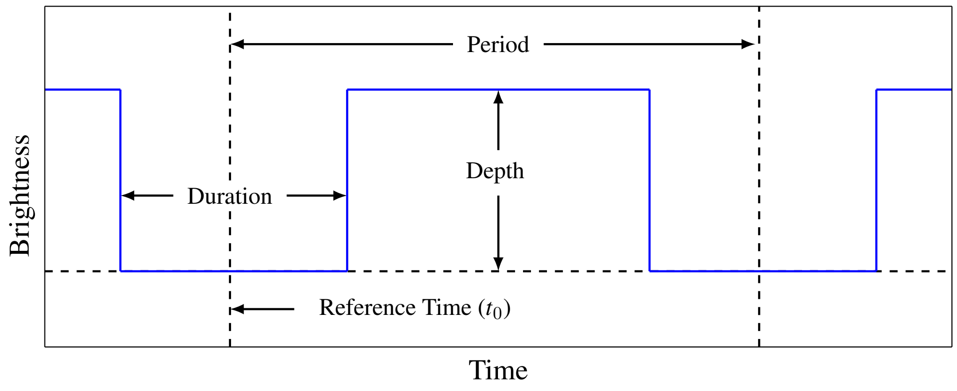
Figure 18 – Treated light curve for quarter 4 of the star Kepler-30



Source: Self elaboration.

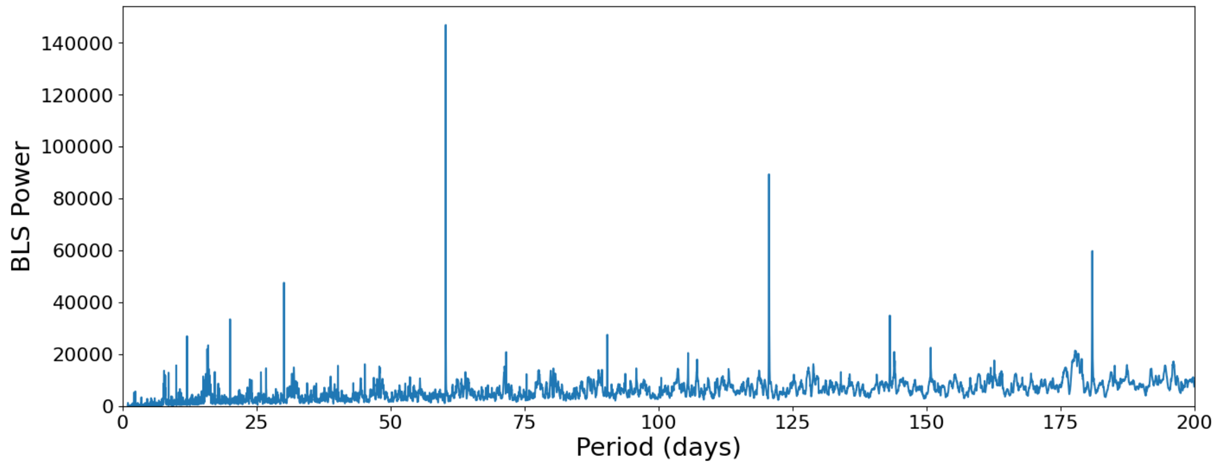
To find periodic signals in a time series like this, we need to generate a spectrum of it, so that we can assess the significance of the signals present in terms of their frequencies or periods. For this purpose, it is common to apply a Fourier transform, generate a Lomb-Scargle periodogram (Lomb, 1976; Scargle, 1982; Zechmeister; KÜrster, 2009), or generate this spectrum using the *Box Least Squares* (BLS) method (KovÁcs *et al.*, 2002). In the case of the BLS method, which is easier to show and comprehend without giving much mathematical details, the periodic signals sought in the curve are those that can be approximated by a rectangular-shaped fit, as shown in Figure 19. Figure 20 displays the periodogram returned by the search for this case.

Figure 19 – Transit model searched by the BLS method.



Source: Self elaboration.

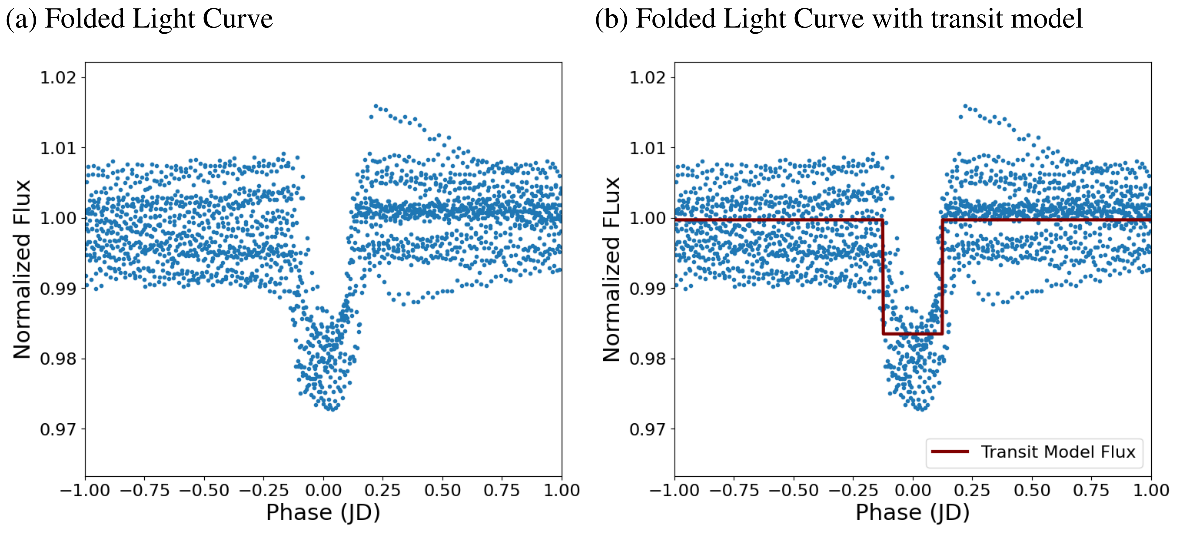
Figure 20 – Periodogram generated for the star Kepler-30



Source: Self elaboration.

We observe a very significant peak at around 60 days. More precisely, the peak occurs at 60.3 days. Speaking arbitrarily, phase folding involves dividing the light curve into cycles defined by a given period and then overlaying these cycles, centering them around a reference time t_0. By doing this, we gain more information about the transit shape, and the flux is now expressed in terms of phase instead of time. We can then use the obtained value to fold the original light curve in phase and assess if the signal is consistent with a planetary transit. Figure 21(a) shows the result of this process.

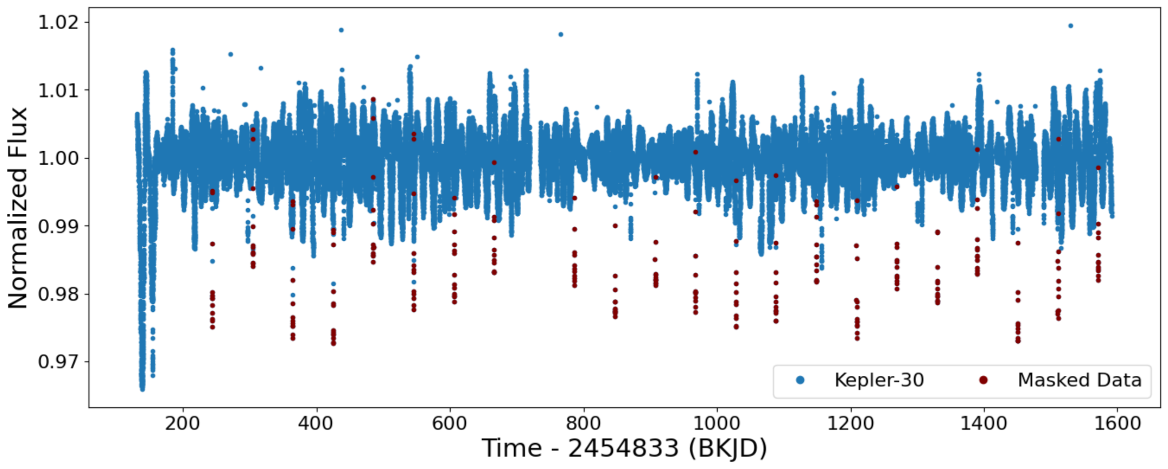
Figure 21 – Light curve of Kepler-30



Source: Self elaboration.

We can see that this is indeed a strong candidate for a planet. With BLS, we can create a model for this transit. Essentially, it fits the data with horizontal lines until the point where they change abruptly, as is the case with a flux drop. The transit data is fitted separately. Figure 21(b) shows the result of this fit. Based on this model, we can determine that the duration of the transit is 0.25 days. Combining this information with the period found in the analysis of Figure 20, we can mark all the points in the original light curve that represent the transit with a mask, as shown in Figure 22.

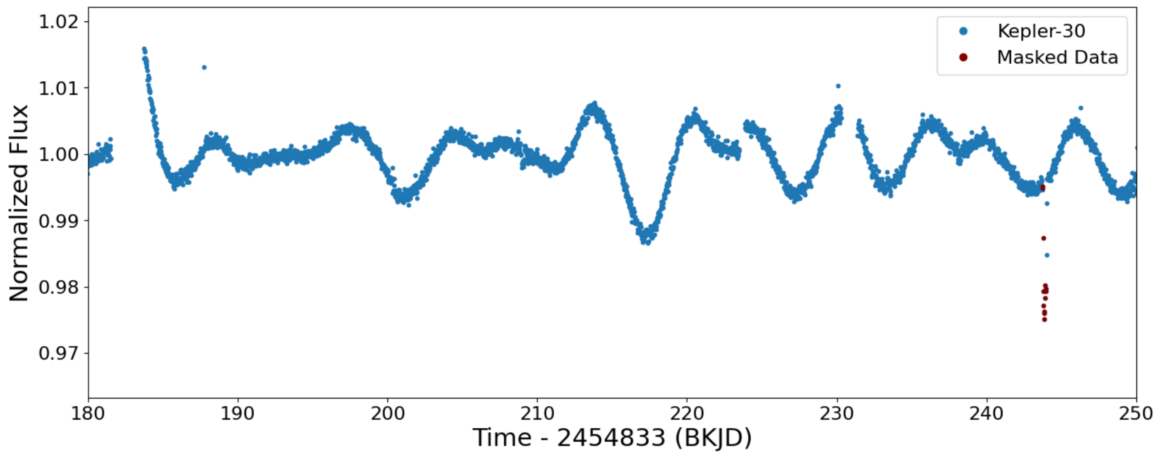
Figure 22 – Light curve of Kepler-30 with the transit mask



Source: Self elaboration.

At first glance, we could say that t_0 occurs a little over 240 BKJD, but this analysis should also consider the transit period found and the time of the curve's start on the graph. This is because it's possible that some of the transits are lost in gaps, including the first one, as shown in Figure 23. In this case, t_0 is at 183.5 BKJD, precisely within the observed gap on the left side of the data. Thus, we obtain all the necessary parameters to analyze this explained TCE. By the way, it refers to the planet Kepler-30c. Other TCEs can also be found in the same curve as shown

Figure 23 – Beginning of the light curve of Kepler-30

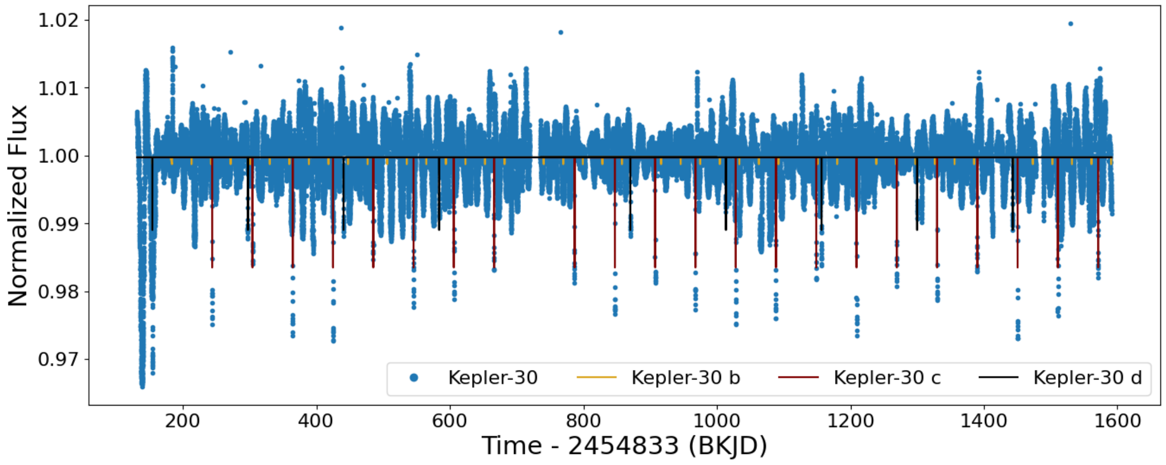


Source: Self elaboration.

in Figure 18. For this, it's necessary to analyze the other relevant signals in Figure 20 or generate similar ones with different period intervals.

Following the same methodology, it is possible to find the planet Kepler-30d, with a period of 143.3 days. It's also possible to find the planet Kepler-30b, but its signal is very weak and therefore prone to be confused with noise. In situations like this, a list of TCEs can be generated as in the previous steps. Most of them are likely not signals of interest, but all can be analyzed with machine learning or deep learrning (see Chapter 3). The period of Kepler-30b is 29.3 days. The highlighted transits of all the planets can be seen in Figure 24.

Figure 24 – Light curve with the transits of the planets from Kepler-30 highlighted



Source: Self elaboration.

Everything that has been done so far illustrates the practical features for the search for exoplanets through the transit method. Operationally speaking, this method has two stages: detection and validation (or confirmation). What we have been presented here are possible steps for the *detection* stage. The second stage takes these signals into account and investigates them thoroughly to confirm whether they correspond to an exoplanet or not. Confirmations can be made through statistical methods, based on the available data and other existing work, or they can be observational, where more data is collected to complement the investigations. It is common for observational confirmations to combine information from another detection method.

The most sensitive and error-prone part of this detection process occurs when analyzing periodograms like the one in Figure 20. In many cases, this is due to the peaks of exoplanet signals being obscured by signals of any other nature, as discussed in section 2.1. At this point, manually analyzing all possible peaks for just one target star can become overwhelming, even considering signals with a minimum SNR value. The situation becomes more challenging when searching for small planets, where the SNR falls below this minimum threshold. This is because the false positive rate becomes very high, which can be very costly in the confirmation stage. Therefore, all this problematic situation justifies the use of automatic detection methods, such as machine learning, as discussed in Chapter 3.

2.2.3 Manual and Machine Vetting of Exoplanet Candidates

In the previous subsection, we gave an overview on how to detect potential transit signals on a light curve. Now, we focus on how to confirm or validate this signals. As discussed previously, Kepler and also TESS pipelines have a very complex algorithm to label a pattern as a *planetary candidate* (PC). The signals that were considered by the pipeline as such, but failed some consistency test are labeled *false positives* (FP). To *confirm* or *validate* a candidate, it has to pass on all the consistency tests and a review of all the evidence by the entire space mission's Science Team has to be made. The reader can refer to Borucki and others (2011) for details of these tests. When possible, high-precision RV measurements (Borucki; others, 2010; Jenkins; others, 2010) or transit timing variations (Holman *et al.*, 2010; Lissauer *et al.*, 2011) methods are used to confirm exoplanets. When this is not the case, an extensive analysis of the spacecraft and ground-based data is done and it may allow the statistical *validation* of an exoplanet. This is done by showing that the planetary interpretation is at least 100 times as probable as a false positive (Lissauer *et al.*, 2011; Torres; others, 2010).

In what regards FP transit events, Kepler/TESS pipeline might confuse planetary transits with sources like *Eclipsing Binaries* (EBs), *Background Eclipsing Binaries* (BGEBs) planet transiting background stars, stellar variability, and instrument-induced artifacts (Borucki; others, 2011). EBs are binary star systems in which the orbital plane of the two stars is aligned in such a way that one star periodically passes in front of the other, as observed from the telescope. This results in a regular decrease in the system's brightness, mimicking the signature of a transiting exoplanet. BGEBs refer to eclipsing binary star systems that are not associated with the target star being observed for exoplanet transits. These binaries are located in the background, meaning they are not the star of interest, but their light overlaps with the light from the target star.

Both Kepler and TESS pipelines have a *Data Validation* (DV) step, where several diagnostic tests are provided to identify many possibilities of a FP flag for a candidate (Twicken; others, 2018; Twicken *et al.*, 2020). For each target star with at least one TCE in a given Pipeline run, a comprehensive DV Report in a PDF format is automatically generated and delivered to the NASA Exoplanet Archive at *NASA Exoplanet Science Institute* (NExScI), where they are accessible by the science community and general public (Twicken; others, 2018). This PDF file contains a one-page DV summary report, which includes:

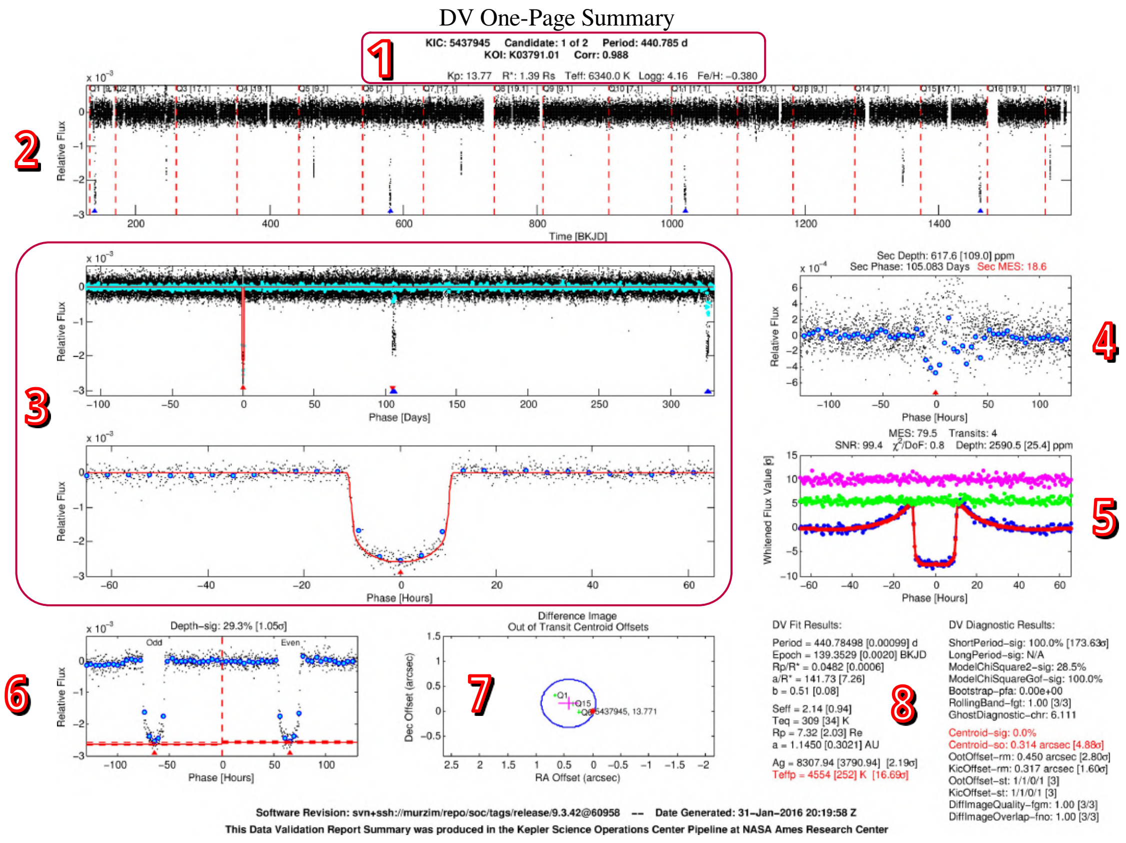
1. TCE information and stellar parameters regarding the target star;
2. Unfolded light curve flux data;
3. The phase-folded full orbit and the transit view of the flux data to determine the existence of a secondary eclipse and shape of the signal;
4. The phase-folded transit view of the secondary eclipse;
5. The phase-folded whitened transit view;
6. The phase-folded odd and even transit view;
7. The difference image related to background image of the target;
8. An analysis table with values related to the model fit and various DV diagnostic parameters.

As explained in Twicken and others (2018), all of these informations are relevant in the process of validating a PC. This points will not be discussed in details in this work, but we strongly encourage the reader to refer to the cited paper. An example of a one-page DV summary report is presented in Figure 25.

All this manual analysis for each TCE is very time-consuming and machine learning methods are well suited for reducing the amount of time required for finding exoplanets in this kind of situation. Among many machine classifiers models (not necessarily machine learning models) present in literature, some gained special attention due to their impact in the time they were proposed and due their influence on other works. They are vespa (Morton; Johnson, 2011; Morton, 2012; Morton *et al.*, 2016), Robovetter (Coughlin; others, 2015; Coughlin *et al.*, 2016), Autovetter (Jenkins *et al.*, 2012; McCauliff *et al.*, 2015), AstroNet (Shallue; Vanderburg, 2018), ExoNet (Ansdell; others, 2018), GPC/RFC (Armstrong *et al.*, 2021) and ExoMiner (Valizadegan *et al.*, 2022).

AstroNet marked a historical point in the exoplanet search field, not only being the first deep learning model used to find a new planet, but also helping to confirm the first known eight-planet system. GPC was able to validate 50 new exoplanets and more recently ExoMiner achieved the incredible mark of 301 new exoplanets validations. This later model represents the state-of-the-art in terms of deep learning applied to exoplanet search. Considering the types of

Figure 25 – Example of a one-page data validation summary report



DV One-Page Summary

KIC: 5437945 Candidate: 1 of 2 Period: 440.785 d
KOI: K03791.01 Corr: 0.988
Kp: 13.77 R*: 1.39 Rs Teff: 6340.0 K Logg: 4.16 Fe/H: –0.380

DV Fit Results:

Period = 440.78498 [0.00099] d
Epoch = 139.3529 [0.0020] BKJD
Rp/R* = 0.0482 [0.0006]
a/R* = 141.73 [7.26]
b = 0.51 [0.08]

Seff = 2.14 [0.94]
Teq = 309 [34] K
Rp = 7.32 [2.03] Re
a = 1.1450 [0.3021] AU

Ag = 8307.94 [3790.94] [2.19σ]
Teffp = 4554 [252] K [16.69σ]

DV Diagnostic Results:

ShortPeriod–sig: 100.0% [173.63σ]
LongPeriod–sig: N/A
ModelChiSquare2–sig: 28.5%
ModelChiSquareGof–sig: 100.0%
Bootstrap–pfa: 0.00e+00
RollingBand–fgt: 1.00 [3/3]
GhostDiagnostic–chr: 6.111

Centroid–sig: 0.0%
Centroid–so: 0.314 arcsec [4.88σ]
OotOffset–rm: 0.450 arcsec [2.80σ]
KicOffset–rm: 0.317 arcsec [1.60σ]
OotOffset–st: 1/1/0/1 [3]
KicOffset–st: 1/1/0/1 [3]
DiffImageQuality–fgm: 1.00 [3/3]
DiffImageOverlap–fno: 1.00 [3/3]

Software Revision: svn+ssh://murzim/repo/soc/tags/release/9.3.42@60958 –– Date Generated: 31–Jan–2016 20:19:58 Z
This Data Validation Report Summary was produced in the Kepler Science Operations Center Pipeline at NASA Ames Research Center

Note: The red big numbers indicate the corresponding enumeration given on the text above.
Source: NASA Ames Research Center. Available via download at https://mast.stsci.edu/portal/Mashup/Clients/Mast/Portal.html. Accessed on 15 May 2023.

information and parameters used on manual vetting, Table 2 shows what kind of data the cited models use.

Table 2 – Comparison of different models in literature for exoplanet classification

		vespa	Robovetter	Autovetter	AstroNet	ExoNet	GPC	RFC	ExoMiner
Scalar inputs	Primary properties	✓	✓	✓			✓	✓	✓
	Transit fit model	✓	✓	✓			✓	✓	
	Stellar parameters			✓		✓	✓	✓	✓
	Optical ghost test		✓	✓			✓	✓	✓
	Bootstrap fap		✓	✓			✓	✓	✓
	Rolling band-fgt		✓	✓					✓
	Secondary properties		✓	✓			✓	✓	✓
Nonscalar inputs	Unfolded flux		+						
	Phase-folded flux		+	+	✓	✓	✓	+	✓
	Odd and even views		+	+			+	+	✓
	Weak secondary flux		+	+			+	+	✓
	Difference image		+	+			+		+
	Centroid motion test		+	+		✓			✓

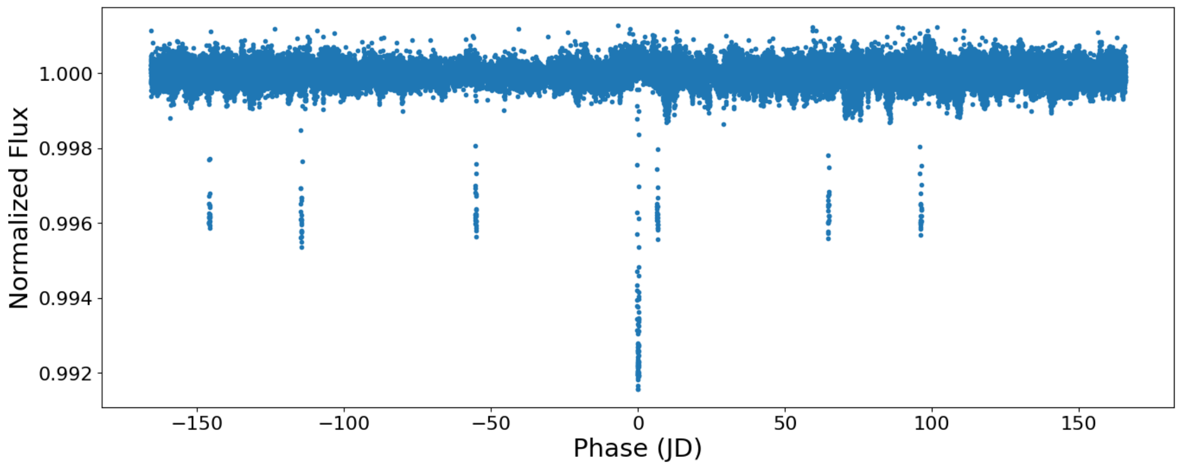
Note: The symbol ✓means that the model uses the original nonscalar time series or image as its input; + means that the model uses only a few scalar values summarizing the time series or image as its input. Primary properties includes MES, planet radius, transit depth and duration, etc. Secondary properties includes geometric albedo, planet effective temperature, and MES for secondary transit.
Source: Adapted from Vallenari *et al.* (2022).

Although not being the best model nowadays, we chose to follow in the footsteps of AstroNet to perform a *conceptual trial* on the time series imaging, discussed in chapter 4. As we can see in Table 2, it only uses one type of astrophysical data to make its predictions, which simplify some things. Recent literature indicates that flux analysis alone is not enough to achieve state-of-the-art performance on exoplanet detection. However, we stress that the aim of this work is no other than to evaluate if the time series imaging technique is viable and, most importantly, helpful to improve a deep learning model performance in this context.

2.3 Preparing the Data for Machine Learning Algorithms

In this section, we illustrate the process of adjusting the light curves discussed in the previous sections to ensure compatibility with machine learning algorithms. Standardizing the light curves is essential to facilitate seamless integration with these algorithms. The need for standardization arises because machine learning models operate with a predetermined number of inputs, whereas our light curves typically consist of varying numbers of measured flux points. To exemplify the standardization procedure, let's revisit the example of the Kepler-90 star. After the phase folding process with respect to the most prominent signal, we have $t_0 = 140.480$ BKJD and its period $p = 331.603$ BKJD. Coincidentally, this happens to be the signal associated with the planet Kepler90-h (Cabrera *et al.*, 2013), and the result of this process is shown in Figure 26.

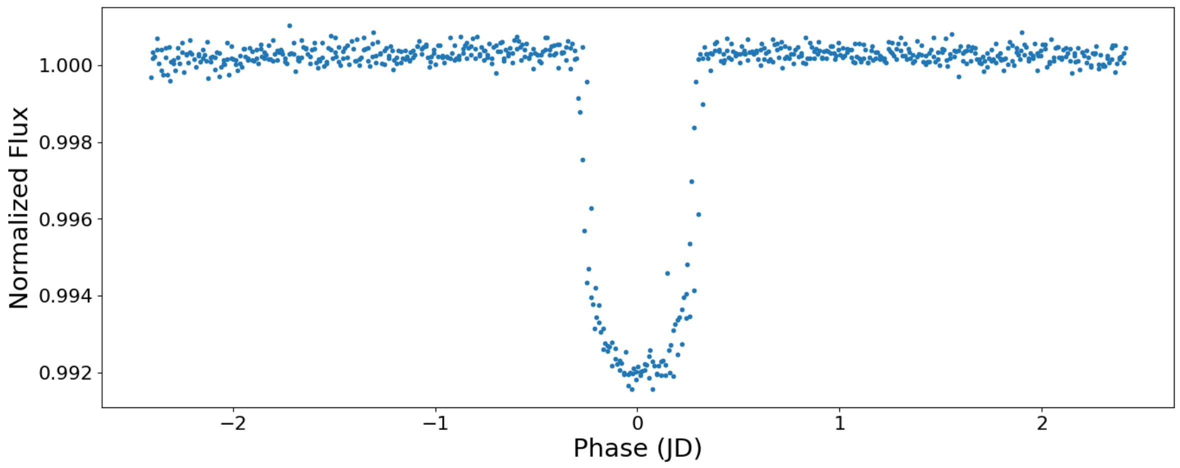
Figure 26 – Phase diagram of the star Kepler-90 for the transit of the planet Kepler-90 h



Source: Self elaboration.

The range of phase values on the x-axis is equal to the signal period (from $-\frac{p}{2}$ to $\frac{p}{2}$) and is given in Julian days. Comparing Figures 15 and 26, it is possible to observe a greater number of points describing the central transit in the second figure than in any of the individual flux drops for that TCE in the first figure. However, we should take a closer look at this event to analyze it properly. We can select only the flux and time data that surrounds the transit of interest, as shown in the in Figure 27.

Figure 27 – Detailed transit of the planet Kepler-90h



Source: Self elaboration.

We then have a localized view of the transit, which will be referred to as the local view. To maintain a standard when performing the same procedure for the other light curves, all local views have their TCEs approximately located in the center of the graph and a data interval equal to k transit durations to the left and right of them. The choice of this value is somewhat arbitrary, and the convention in this work is $k = 4$. Continuing with the idea of standardizing all the data, a problem that arises when expanding the transit in this way is the number of points that each light curve will have. More periodic transits, for example, will have more points constituting their graph in the phase diagram.

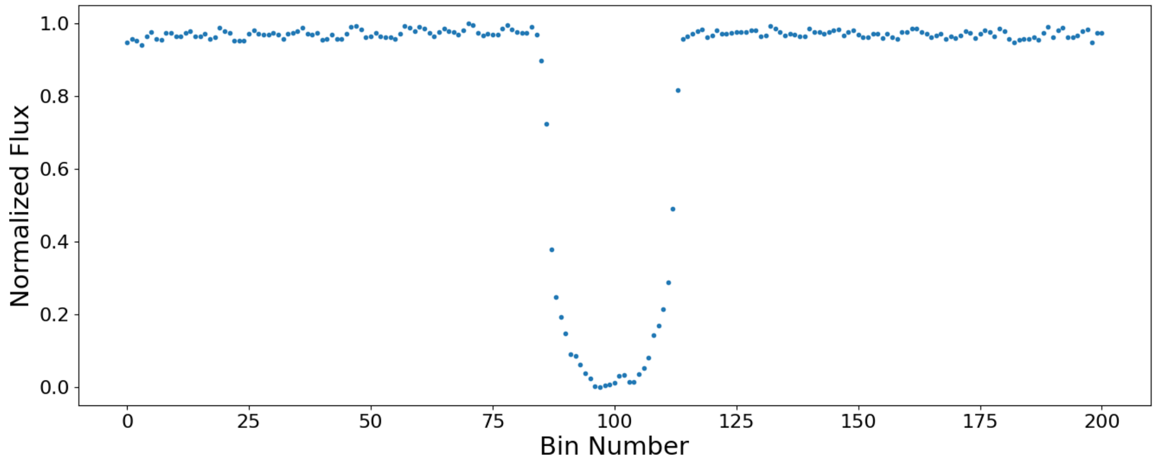
This can be resolved using a technique called *data binning* (Deepchecks, 2023; Roberts, 2023). Essentially, data binning involves grouping the data. More precisely, we define several intervals of width δ on the time axis (or on the phase axis in this case), separated by a distance λ, and choose how to categorize the flux data contained in each of these intervals. We choose to represent them by the median of the values in that interval, as it is a statistical representation less affected by potential outliers and better preserves the shape of the transits. The values of these two parameters are also arbitrary to some extent, but it was observed that some transits became better distinguishable when $\delta > \lambda$, where the intervals overlap. This technique is a very similar to a *Piecewise Aggregate Approximation* (PAA) (Keogh; Pazzani, 2000), which divides a time series into N equi-sized parts and takes the mean of the data that falls within each interval.

By performing this procedure, another parameter that naturally arises is the number of *bins* that the new light curve will have, that is, the number of points that will describe the light curve. To maintain a general standard for the light curves, we established that all local light curves will have exactly 201 points, or bins. This odd number is chosen so that the curve has a central point where the transit can ideally be positioned symmetrically whenever possible. Since a fixed number of bins has been established, the relationship that gives us the parameter values is

$$\lambda = \frac{\delta}{2} = \frac{\Delta d}{N_b}, \tag{2.3}$$

where N_b is the number of bins and Δd refers to the amount of data (points) from the original curve that describes it in the considered time interval. Finally, since we are only interested in studying the transit shape from planets of various sizes, the flux values were normalized between 0 and 1 to prevent the transit depth from exerting too much influence on the planet characterization. The final local curve for the example of the planet Kepler-90 h is shown in Figure 28.

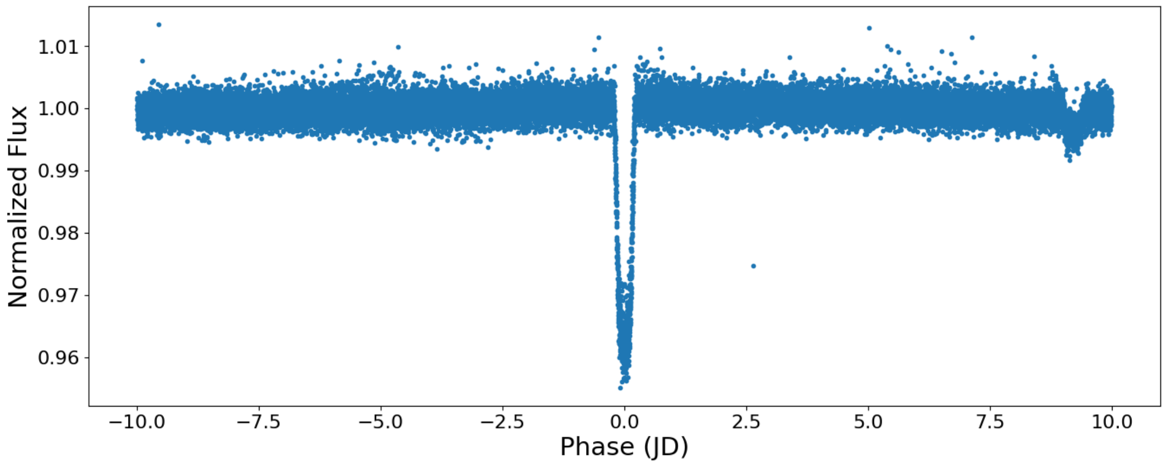
Figure 28 – Local view of the Kepler-90h transit



Source: Self elaboration.

At the end of this process, all transit information becomes described by a one-dimensional array. This is particularly interesting from the perspective of machine learning, as with binning, we will have less noise in the data, in addition to more efficient and faster-to-train models (Ke *et al.*, 2017). However, we will be losing valuable information if we only consider the local view of a TCE. By performing the entire data processing process on the light curve of the star KIC 5130380 and selecting the first of its two TCEs in the catalog table, we generate the phase diagram shown in Figure 29.

Figure 29 – KIC 5130380 star phase diagram for one of its TCEs



Source: Self elaboration.

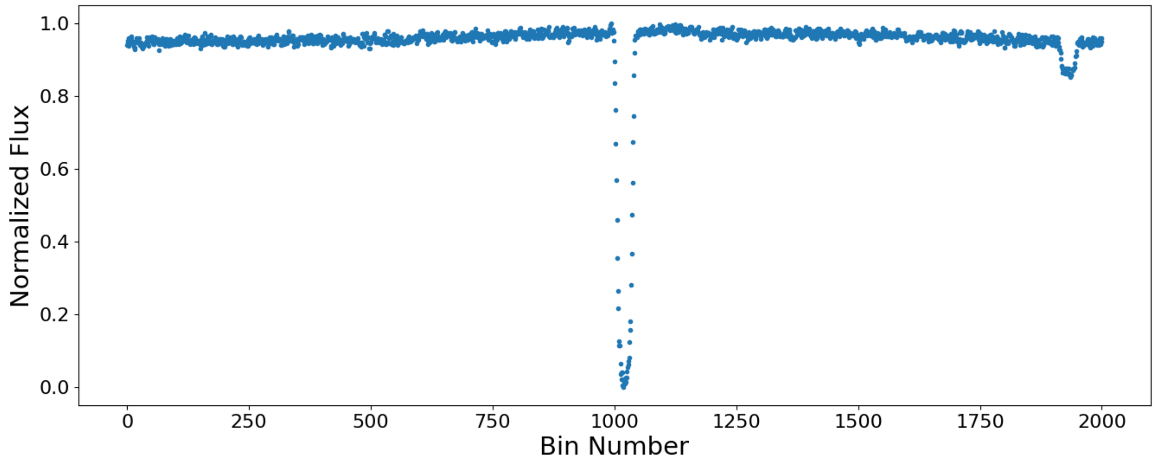
If we fold the light curve based on the period of a specific TCE, we expect that any signal, apart from the one of interest, that has the same period will add up at a fixed location on the curve. This is precisely what happens in this case, as we can observe two well-defined dips in this diagram. The centrally located dip corresponds to the chosen TCE for analysis, while the other dip corresponds to the second TCE of this star. We see that the second dip has the same period as the highlighted TCE, albeit with a phase difference. This provides an indication that this system is a binary star[4] (Prša; Zwitter, 2005) rather than necessarily a transiting planet. In fact, we can observe from Figure 26 that a transit with a different period (that of the planet Kepler-90 g) from the one we are analyzing is unevenly distributed in the plot.

Thus, it is also important to consider what happens throughout the phase diagram. We followed a similar procedure to the local curves regarding data binning. Since the Δd is now larger, it is natural to represent the curve with a larger number of bins to avoid the loss of important information, as seen in Figure 29. We refer to these larger curves as global curves and define a fixed number of bins equal to 2001. Equation 2.3 also applies in this case. The result of this process can be seen in Figure 30.

One reason for not using only the global light curves is that more subtle information is lost in the data binning process in some cases. Specifically, very short transits can end up

[4]A binary star is a system composed of two stars that are close to each other. They orbit the center of mass of the system, and due to their proximity, the light from one star can be “hidden” by the other for an observer situated far from the system when the stars align. In addition to the secondary dip, another very recurrent characteristic is the “V” shape that the curve assumes, with or without a secondary transit.

Figure 30 – KIC 5130380.01 global light curve

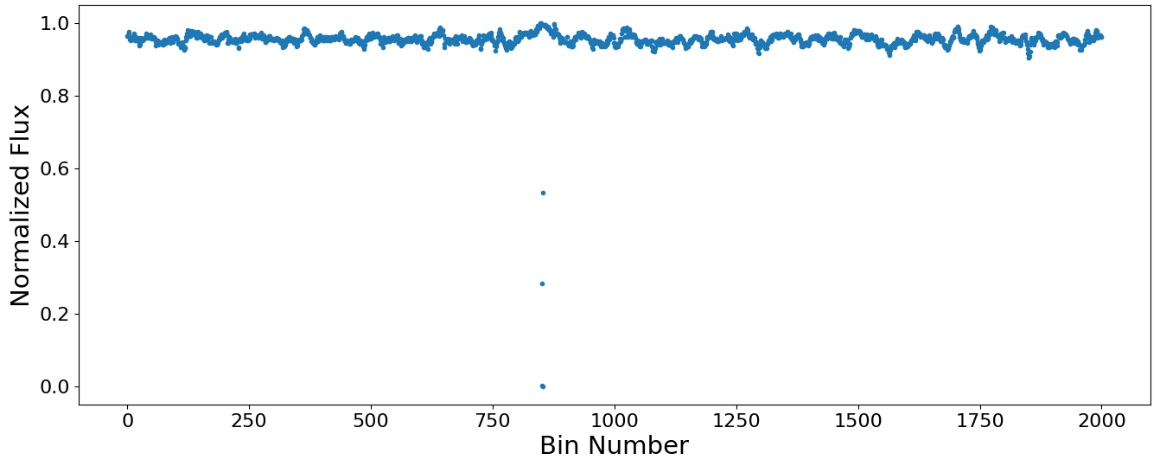


Source: Self elaboration.

completely within the width δ, causing their shape to be partially or completely lost in a global view, as shown in the global curve of Kepler-90 h itself in Figure 31.

The binning process removed all significant influence that the transit of Kepler-90 g could have had on the curve, but it also did so for Kepler-90 h itself. It is hard to expect the algorithm to understand that there is a planet in this global curve, but there is no doubt when looking at its local representation. In cases like those in Figure 30, it is expected that the global curve is more useful in indicating that the TCE is not a planet rather than the opposite. With this, the data processing is complete, and the curves can now be used by machine learning algorithms.

Figure 31 – Kepler-90 h global light curve



Source: Self elaboration.

2.4 Dataset Statistics

The data used in this study comes from the *Autovetter Planet Candidate Catalog for Q1-Q17 DR24*[5] (Catanzarite, 2015). This catalog was generated by the autovetter (McCauliff *et al.*, 2015), which is a machine learning algorithm for the automatic classification of TCEs. The catalog has a direct but independent relationship with the one produced by the robovetter (Coughlin; others, 2015), which is also an algorithm for the automatic classification of TCEs, but does not use machine learning. Both algorithms make use of data processed by the Kepler pipeline to generate their products. More precisely, the autovetter classifies TCEs with labels described by Table 3.

Table 3 – Types of labels for TCE classification by the autovetter.

Label	Description
Planet Candidate (PC)	Contains signals that are consistent with planetary transits and there is no reason to discard the hypothesis that it is a planet.
Astrophysical False Positive (AFP)	Contains signals of astrophysical origin that can mimic planetary transits, such as binary eclipses, pulsating stars, sunspots, and other periodic signals that provide strong evidence to discard the hypothesis of a transit due to a planet.
Non-Transiting Phenomenon (NTP)	Contains signals that are evidently of instrumental or noise origin.
Unknown (UNK)	Refers to TCEs not included in the dataset of this work but part of the rest of the used dataset.

Source: Adapted text from McCauliff *et al.* (2015) and Catanzarite (2015).

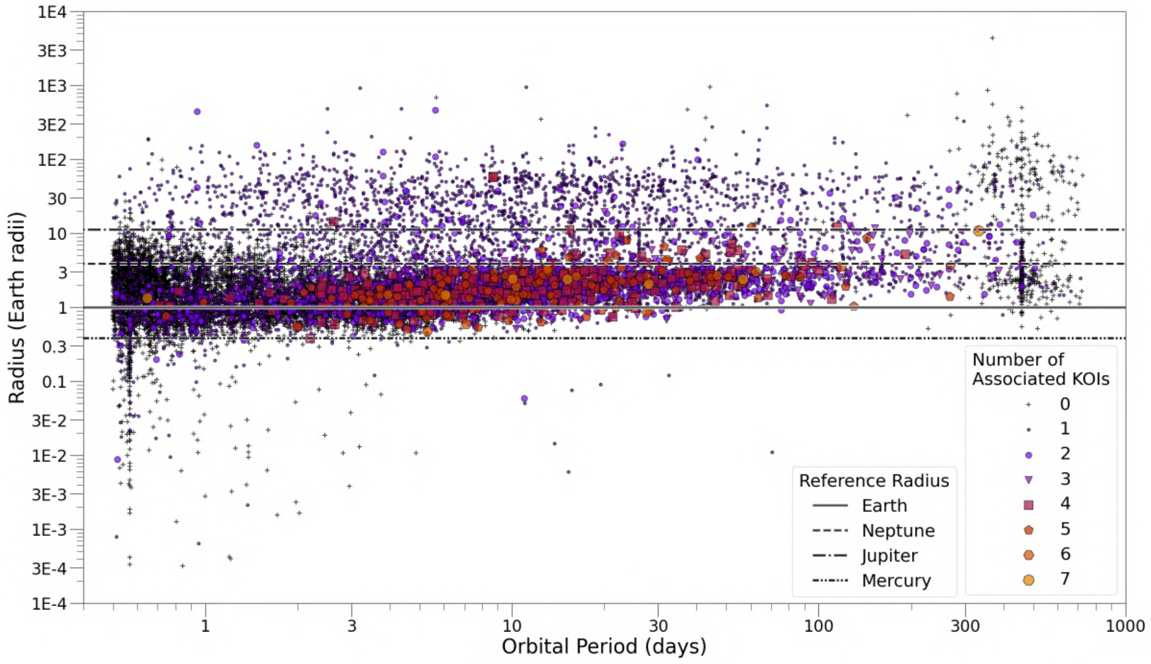
There are more recent Kepler catalogs, such as the final official Kepler Data Release 25, Q1–Q17 (Thompson; others, 2018) and updated versions of it (Lissauer *et al.*, 2023). However, DR24 was chosen so we could use the autovetter labels, as indicated in Table 3. There are 20367 TCEs in total, but the usable data for this work's purpose were chosen to be only those whose labels are PC, *Astrophysical False Positive* (AFP) and *Non-Transiting Phenomenon* (NTP). The filtered dataset ended up with 15737 TCEs. From these, 9596 are labeled as AFP, 3600 as *Planet Candidate* (PC) and 2541 as NTP. For a binary classification problem, we are interested only on what is planet or not. Therefore, we are dealing with a skewed dataset, since PC class only represents 22,88% of all data.

Next step is to see if the data is, indeed, representative with the available data we currently have from exoplanets. A classic plot is the radius-period distribution, were, along with many other distributions, one can try to find clues to distinguish between potential planetary

[5]The TCE interactive table for this data can be found here: https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=q1_q17_dr24_tce

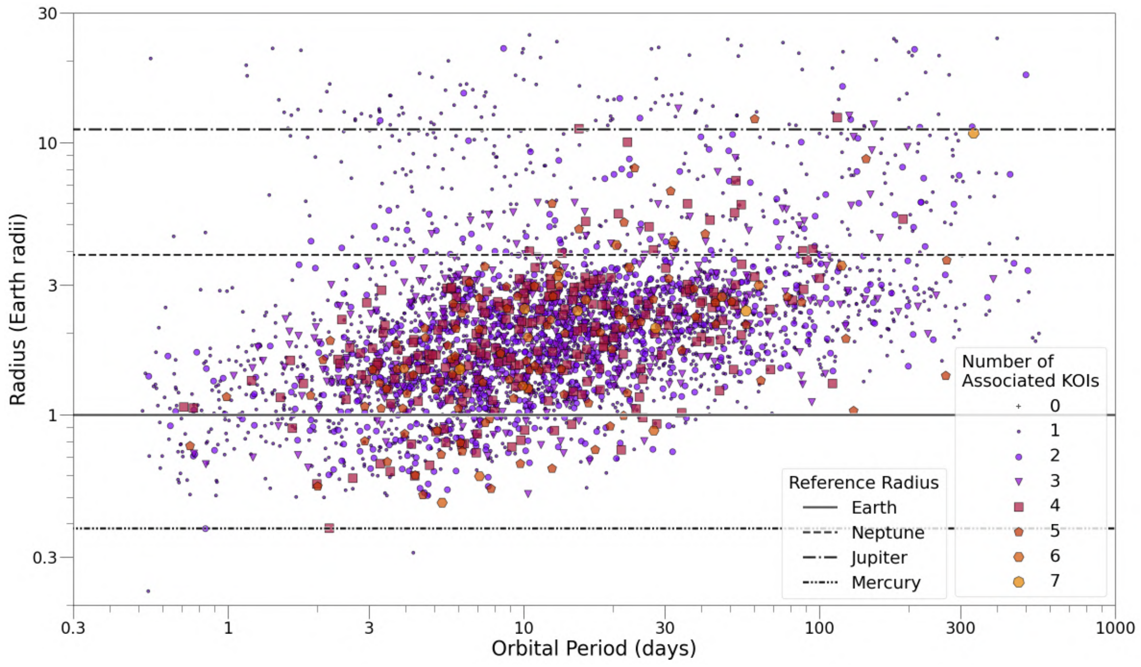
formation mechanisms (BEAUGE; NESVORNỲ, 2012; UZSOY *et al.*, 2021). Figure 32 shows this particular distribution for all TCE classes, whilst Figure 33 shows the same distribution, but only for the PCs.

Figure 32 – Radius-period distribution for all TCEs in the used dataset



Source: Self elaboration.

Figure 33 – Radius-period distribution for all PC-labeled TCEs in the used dataset



Source: Self elaboration.

On both plots, TCEs are distinguished by the number of associated KOIs (#KOIs). A *Kepler Object of Interest* (KOI) is a star identified by the Kepler space telescope that exhibits periodic dimming events suggestive of an exoplanet transit. Therefore, this distinction takes into account the total number of TCEs detected in a KOI. Table 4 shows the counting of TCEs with all available number of associated KOIs. Interestingly, PC-labeled TCEs do not have any null number of associated KOIs, whilst all non-PCs have the maximum count of them. We see a clear proportion between PC rate and #KOIs. However, since the number of associated KOIs is a measure strongly related to Kepler's pipeline, this observation must be taken with prudence.

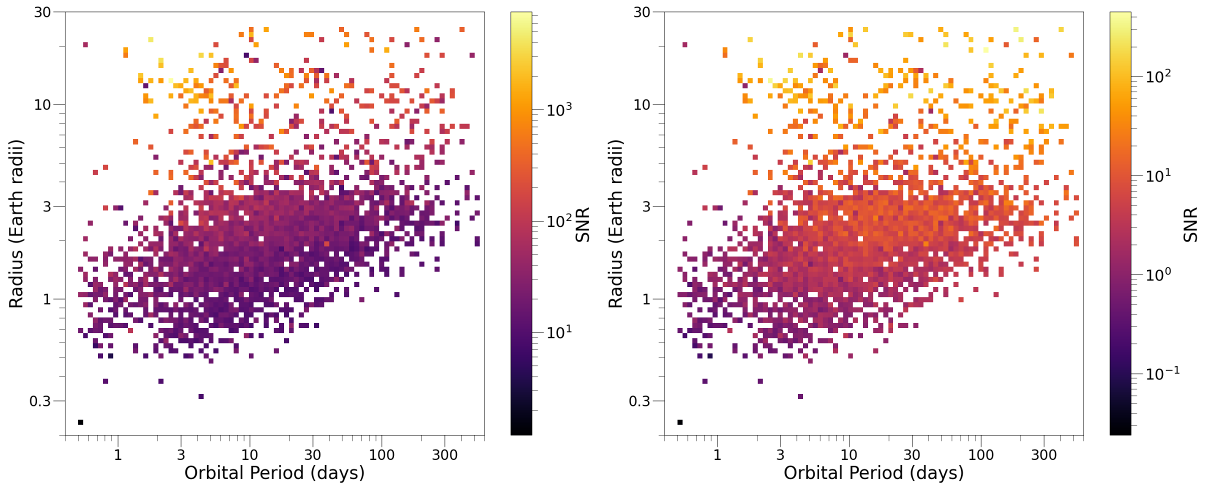
Table 4 – Counting of the number of associated KOIs for TCEs in the dataset

All TCEs		**PCs**		**Non PCs**	
# Associated KOI	Count	# Associated KOI	Count	# Associated KOI	Count
0	7376	0	0	0	7376
1	6435	1	1945	1	4490
2	1084	2	856	2	228
3	474	3	443	3	31
4	238	4	229	4	9
5	108	5	106	5	2
6	15	6	15	6	0
7	7	7	6	7	1

Source: Self elaboration.

Other information of interest is the SNR of the TCE. As explained in subsection 2.2.1, it is the power of the signal divided by the power of inherent noise, estimated via equation 2.2. We can take leverage of the distribution from Figure 33 to see the dependency of the SNR on planetary radii and orbital period, as shown in Figure 34.

Figure 34 – SNR of TCEs as a function of location in the period-radius plane
(a) Total SNR (b) Mean SNR

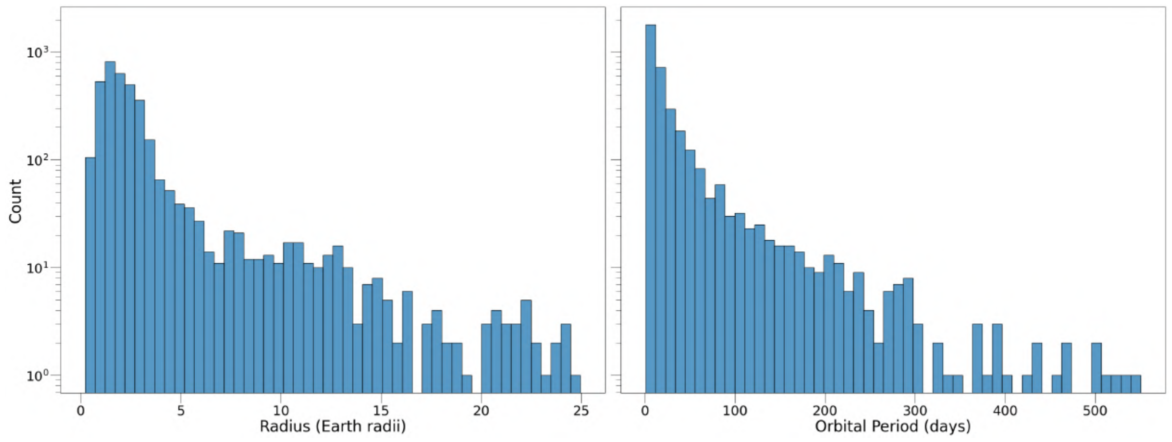


Source: Self elaboration.

On both panels, each square represents 1 to 11 planets and they have a factor of ≈ 1.09 in period an ≈ 1.06 in radius. Figure 34(a) shows the total SNR of a region in the plane and Figure 34(b) shows the average (mean) SNR per transit. The average SNR was calculated by dividing the total SNR by the square root of the number of observed transits for each PC (Lissauer *et al.*, 2023). This is an important analysis to have in mind because SNR, along with MES, have threshold values for cutting away many TCE from further analysis in Kepler's pipeline (Kunimoto *et al.*, 2018).

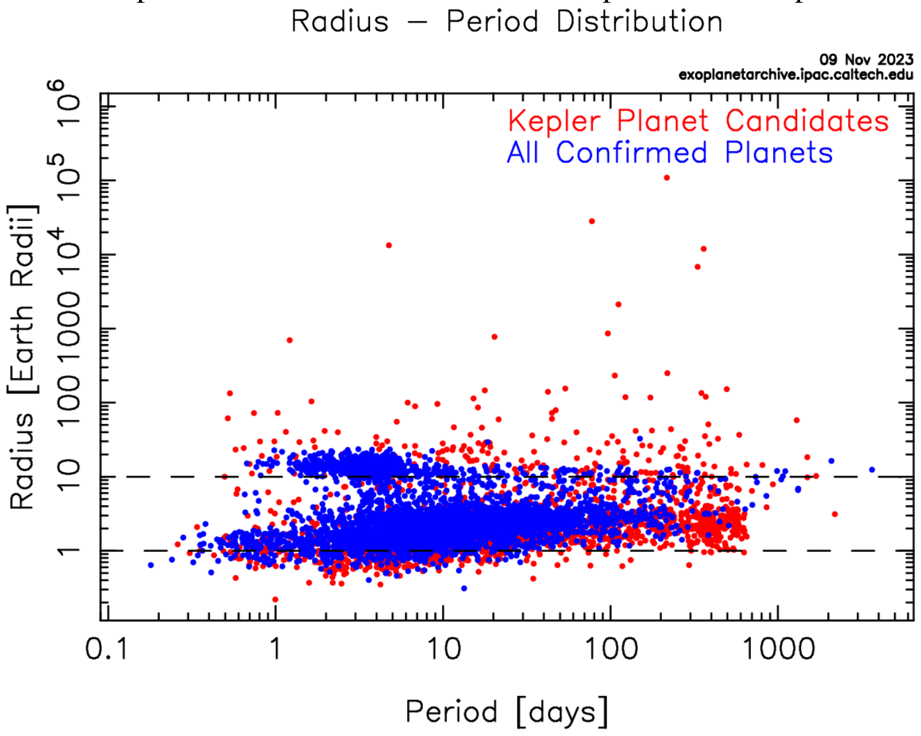
Figure 35 shows the counts of radii and orbital period for all planets in the dataset. Many other analysis could be done, but comparing this findings with Figure 36 shows that our planet data is diverse and representative.

Figure 35 – Radius and orbital period counts for planets in the used dataset



Source: Self elaboration.

Figure 36 – Radius-period distribution for all confirmed planets and all planet candidates



Note: Up to November 9th, 2023.
Source: NASA Exoplanet Archive, https://exoplanetarchive.ipac.caltech.edu/exoplanetplots/. Access on: 19 nov. 2023.

3 ARTIFICIAL INTELLIGENCE

The *Autovetter Planet Candidate Catalog for Q1-Q17 DR24* (Catanzarite, 2015) brings an interesting approach opportunity for machine learning purposes. There are other works in the literature that use artificial intelligence for the classification of TCEs, each with their own characteristics. In the particular autovetter case, it uses a machine learning algorithm known as Random Forest (Breiman, 2001), which is famous for being fast to train and producing high-quality models. However, our interest is to recognize patterns in light curves, and there are algorithms in the literature that have gained strong relevance in this type of task: *Artificial Neural Networks*, or simply *Neural Network* (NN).

In short, an artificial neural network is a machine learning model inspired by the functioning of the biological neural networks in our brains, where neural activity can be treated using logical-mathematical concepts (McCulloch; Pitts, 1943). With the advancement in the complexity of these models, a subfield of machine learning known as deep learning emerged, which delivers increasingly accurate results (SCHMIDHUBER, 2014; Mu; Zeng, 2019). These concepts will be further addressed in the following sections, but it is worth mentioning that deep learning is very successful in tasks such as image classification and recognition of complex patterns, being used in various areas where complex analyses are required, such as medicine and biomedicine (Suganyadevi *et al.*, 2022; Hafiz; Bhat, 2020; Liu *et al.*, 2022; Luo *et al.*, 2021), engineering (Dimiduk *et al.*, 2018; Zhang *et al.*, 2021; Fan *et al.*, 2019), economics (Long *et al.*, 2019; Andres *et al.*, 2021; Maliar *et al.*, 2021), social and political sciences (Kottursamy, 2021; Chatsiou; Mikhaylov, 2020; Balaji *et al.*, 2021; NYAWA *et al.*, 2022), among many others.

As mentioned in Chapter 1, this work is a continuation of a previous study. Inspired by the work of Shallue and Vanderburg (2018), we created a neural network capable of ranking plausible individual planet signals higher than false-positive signals by the likelihood that they are indeed planets 90% of the time, compared to the 98.8% from the deep learning model in the reference work. However, the previously created algorithm did not fit as an actual deep learning algorithm and its hyperparameters were not adequately adjusted.

In this chapter, we will provide a brief review of the basic concepts of machine learning and discuss one more part of the methodology being employed in this work.

3.1 A Brief Introduction to Machine Learning

Machine Learning (ML) is a subfield of what is known as Artificial Intelligence. The classical definition is given by Samuel (1959) (Samuel, 1959): "Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed". The concept of "learning" became clearer with Mitchell (1997) (Mitchell, 1997): "A computer program is said to learn from experience **E** with respect to some task **T** and performance measure **P**, if its performance on **T**, as measured by **P**, improves with experience **E**".

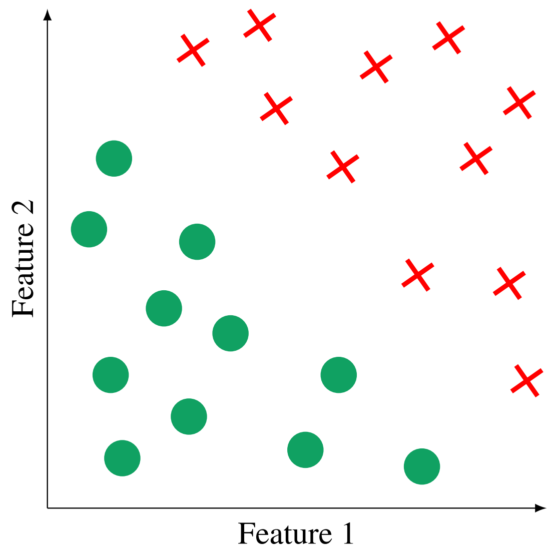
The objective of our algorithm, or its *task* (***T***), is to analyze examples of light curves that represent planets or non-planets so that the algorithm can learn from them and develop a model that differentiates and correctly *classifies* them. These examples, referred to as the *training set*, serve as the *experience* (***E***) for the algorithm. We will further discuss in section 3.5 some performance measures (***P***), but a simple example would be the ratio of the number of correct classifications to the total number of classifications. This particular measure is called *accuracy*. Given the aforementioned specifications, we are dealing with supervised training of a classification algorithm. It is supervised because we know in advance which data represents planets or non-planets.

The process of constructing the algorithm consists of implementing various machine learning models and evaluating which one performs best in recognizing exoplanets. Once the most promising model is chosen, it will be further refined through adjusting *hyperparameters* to deliver even better results. Hyperparameters are parameters that control various aspects of the learning process of a model and directly affect its performance (Claesen; Moor, 2015). In general, we have the freedom to arbitrarily choose their values. Next, we will discuss two of the most common machine learning models suitable for the present case.

3.1.1 Logistic Regression Model

The training process of the logistic regression algorithm essentially consists of determining a hypothesis that best describes the data of interest. As a basic example of this application, we can assume that TCEs are described by two generic features, so they can be represented in a 2D space like in Figure 37. In this pictorial example, TCEs corresponding to planets are represented by circles, while non-planetary ones are represented by crosses. Naturally, this is a binary classification, which was the case considered in this work. TCEs with AFP and NTP labels were all treated as non-planets, and those with the UNK label were discarded. Despite a few errors observed through manual inspection of some TCEs, all labels were treated as ground truth.

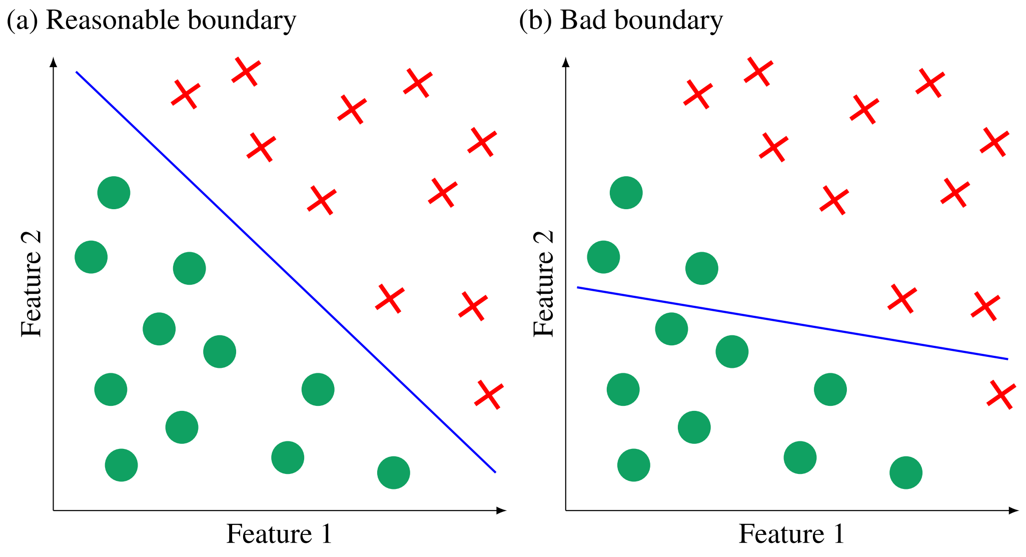
Figure 37 – Representation of an arbitrary two-dimensional feature space



Source: Self elaboration.

In this example case, the hypothesis will take into account a linear function that separates the two classes into well-defined regions in feature space. This function is called the *decision boundary* and is expressed in terms of these features. Figure 38 shows two examples of boundaries that the algorithm can determine.

Figure 38 – Examples of decision boundaries



Source: Self elaboration.

The TCEs depicted in the figures are part of the training set. When the algorithm learns from the training data and establishes a decision boundary, it can classify new TCEs by evaluating which region of the feature space that new data belongs to. In the specific case of this example, this boundary is defined by

$$X = \theta_0 + \theta_1 x_1 + \theta_2 x_2, \tag{3.1}$$

in which x_1 and x_2 are the two generic features considered in this example, the parameters θ_1 and θ_2 represent the weights assigned to the features in order for the decision boundary to fit the data, and the parameter θ_0 is a constant called the bias term, also used in fitting the decision boundary. There are situations where a linear boundary is not able to separate the data well, as shown in Figure 72. Equation 3.1 can be modified to take into account non-linear terms for such situations.

In our case, the feature space for local curves has 201 dimensions, while that of global curves has 2001 dimensions. Thus, the decision boundary separates the respective hyperspaces into appropriate regions for data classification. Equation 3.1 can be generalized to describe data with more features and written in a matrix form as

$$\boldsymbol{X}^{(i)} = \boldsymbol{\theta}^T \boldsymbol{x}^{(i)}, \quad \text{where} \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}, \quad \boldsymbol{x}^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \quad \text{and} \quad x_0^{(i)} = 1. \tag{3.2}$$

The indices i indicate which of the m samples we are considering. Since the term $x_0^{(i)}$ was included for the sake of matrix multiplication coherence, due to the bia term $\theta_0^{(i)}$, the feature space is now $\mathbb{R}^{n+1}$. This term is always equal to 1 for any sample in the dataset.

Roughly speaking, the algorithm learns when it finds the optimal values of $\boldsymbol{\theta}$ that define a decision boundary that accurately describes the training data and can generalize the results to new data. We refer to the situation where a boundary allows a model to learn well from the training data but fails to generalize well to test data as *overfitting*. In contrast, *underfitting* refers to the situation where the model does not learn well from the training data and does not make accurate classifications with test data. Given a decision boundary, we define our hypothesis as

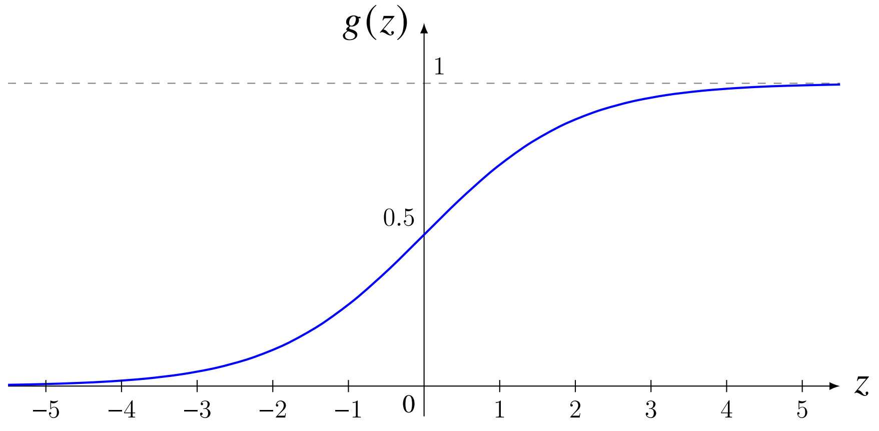
$$h_\theta(\boldsymbol{x}^{(i)}) = g\left(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}\right). \tag{3.3}$$

The function $g(z)$ is non-linear and its role is to evaluate if a given instance $\boldsymbol{x}^{(i)}$ belongs to a class or not. A common choice for this is the function known as *logistic function* or *sigmoidal function*, which is given by

$$g(z) = \frac{1}{1 + e^{-z}}. \tag{3.4}$$

Figure 39 shows its graph. The possible values of $g(z)$ are in the range between 0 and 1.

Figure 39 – Logistic function



Source: Self elaboration.

When $z > 0$, $g(z)$ is greater than 0.5 and the algorithm considers that data to be a positive one[6]. Setting $z = \boldsymbol{\theta}^T \boldsymbol{x}^{(i)}$, we then have

$$h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}}}. \tag{3.5}$$

Thus, we can understand that the hypothesis returns the probability that a TCE is a planet, given its features $\boldsymbol{x}$, parameterized by $\boldsymbol{\theta}$. This is mathematically equivalent to

$$h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) = P(y = 1|\boldsymbol{x}^{(i)}; \boldsymbol{\theta}). \tag{3.6}$$

In this case, y is the classification that the model gives to the TCE. If it is a planet, then it is established that $y = 1$; otherwise, $y = 0$. Naturally, the sum of the probabilities of a TCE being a planet and not being a planet is 100%, i.e.,

$$P(y = 0|\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) + P(y = 1|\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) = 1. \tag{3.7}$$

In order for the algorithm to choose the parameters $\boldsymbol{\theta}$, it must take into account how certain the hypothesis is. From the equations above, it is possible to find a function that measures

[6]This choice is arbitrary, it could be the other way around.

the error the hypothesis makes in relation to the data it should predict. We call this function the *cost function*, and it is given by

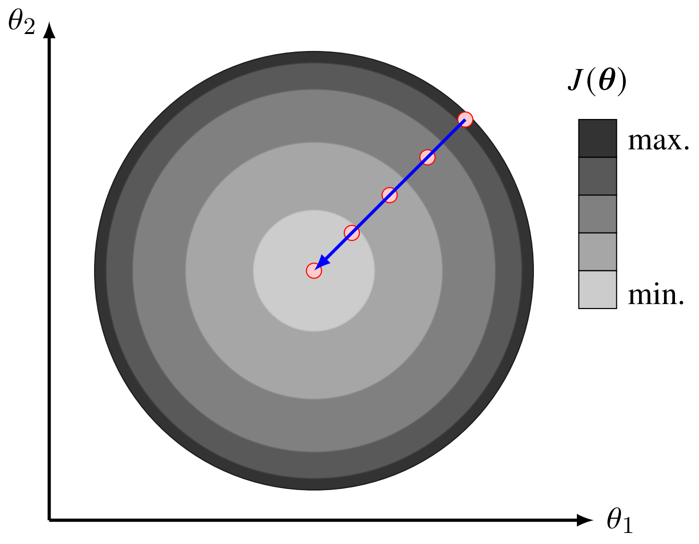
$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left[\boldsymbol{y}^{(i)}\log\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right) + (1-\boldsymbol{y}^{(i)})\log\left(1-h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right)\right] \tag{3.8}$$

The best values of $\boldsymbol{\theta}$ occur when this error is minimized. The algorithm used to minimize this cost function is called *gradient descent*. Its operation consists of initializing random values for all[7] $n+1$ parameters θ and iteratively adding or subtracting a given quantity from all of them in order to decrease the value of $J(\boldsymbol{\theta})$ throughout the iterations. Therefore, the gradient descent algorithm involves repeating the operation simultaneously for all θ_j, where

$$\theta_j = \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\boldsymbol{\theta}) \quad \Rightarrow \quad \theta_j = \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}\left[h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - \boldsymbol{y}^{(i)}\right]\boldsymbol{x}^{(i)}. \tag{3.9}$$

This is repeated until the cost function converges to a minimum value. The (hyper)parameter α is called the learning rate and determines how quickly this minimization occurs. Its value directly affects the model's performance, as very low values make the training time-consuming, while very high values can cause the values to diverge. In the case of the example with the two features shown in Figure 37, the representation of the path that the gradient descent takes in the space of $\boldsymbol{\theta}$ until it converges to a minimum value is given by Figure 40. Given a random initial point in this space, the gradient will follow the path that leads to a point where the derivative of $J(\boldsymbol{\theta})$ is zero.

Figure 40 – Gradient Descent



Source: Self elaboration.

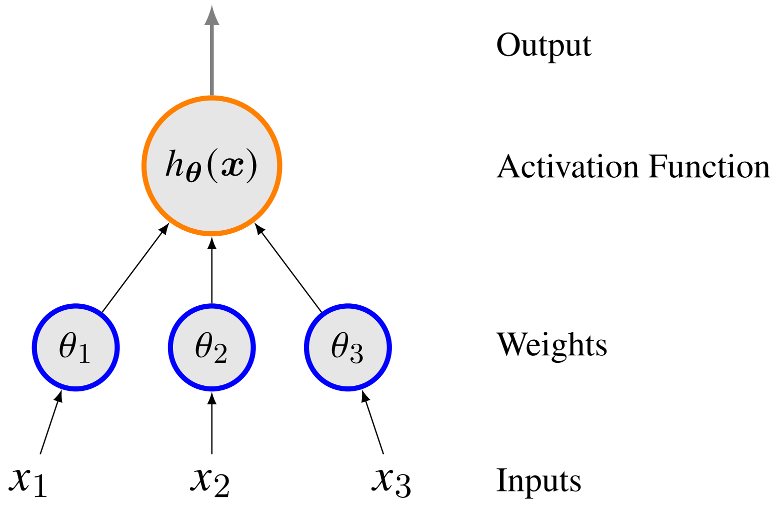
[7]Here, n represents the number of features in the training set. The "+1" is included due to the bias term θ_0.

There are many other optimization algorithms, such as Stochastic Gradient Descent (BOTTOU *et al.*, 1991) or Adam (Kingma; Ba, 2017). The reader can refer to the respective papers for details, but the general idea is that they also search for a optimum parameter configuration in order to increase a model's performance.

3.1.2 Neural Network Model

An *artificial neural network* is a machine learning model that consists of a series of logical operations inspired by the biological neural networks in our brains. In this comparison, the biological neuron is represented by a *logistic unit* or *perceptron* (Lefkowitz, 2019), as shown in Figure 41.

Figure 41 – Scheme of a logistic unit

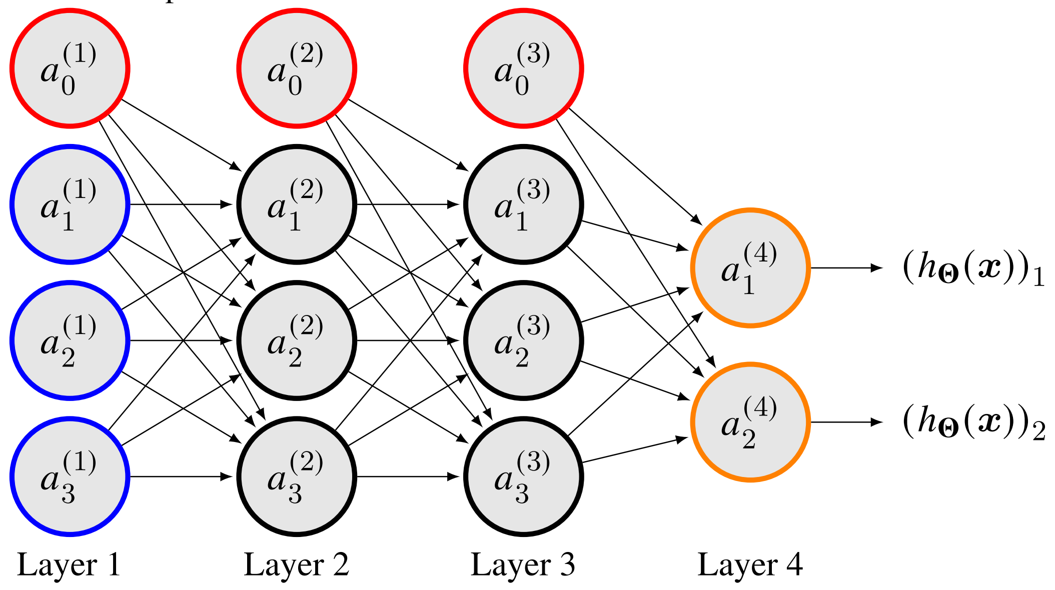


Source: Self elaboration.

This unit receives input data, multiplies each of them by a parameter θ, sums these products, and then calculates a new value by inserting this sum into a function called the *activation function*. This is exactly what happens in the case of equation 3.3 for logistic regression. An important note is that equation 3.3 takes into account a bias term (x_0 and θ_0) that is not included in the figure but should be considered in the model. The great advantage of a neural network is its ability to relate these units in a network to describe complex data in an optimized way. The way these units are connected is called the *network architecture*, and an example is shown in Figure 42.

The notation for the mathematical descriptions of neural networks is slightly different from that used in logistic regression. The terms $a_i^{(j)}$ represent the activation function of logistic

Figure 42 – Example of a neural network



Note: The bias terms $a_0^{(j)}$ were explicitly displayed.
Source: Self elaboration.

unit i in layer j. In particular, layer $j = 1$ is called the *input layer* because it is where the data enters to be analyzed ($a_i^{(1)} = x_i$). Additionally, $a_0^{(j)}$ is the bias term of the j-th layer (which, despite being shown in the figure, is usually omitted and implicit). We then have that[8]

$$\boldsymbol{a}^{(j)} = g\left(\boldsymbol{z}^{(j)}\right), \tag{3.10}$$

where $\boldsymbol{a}^{(j)}$ is the vector with all functions of layer j and $\boldsymbol{z}^{(j)} = \boldsymbol{a}^{(j-1)}\mathbf{\Theta}^{(j)}$.

Instead of a vector to represent the weights, we now use a matrix $\mathbf{\Theta}^{(j)}$ that maps the parameters of a layer j to a layer $j+1$. If a layer j has s_j units and a layer $j+1$ has s_{j+1}, then the dimension of the matrix $\mathbf{\Theta}^{(j)}$ is $s_{j+1} \times (1+s_j)$. The last layer is called *output layer*, as it is where the processing result comes out and is displayed to us. With that, we have $(h_{\mathbf{\Theta}}(\boldsymbol{x}))_i = a_i^{(L)}$, where L is the total number of layers (and therefore last in this notation). The other layers between the input and output layers are culturally called *hidden layers*, as we don't see their values directly. All layers, except the output layer, have a bias unit.

Upon receiving data, the flow of operations of a neural network starts at the input layer and goes to the output layer. This is called *forward propagation*. The most popular algorithm for training neural networks is *backpropagation*. As the name suggests, the processing flow of this algorithm goes in the opposite direction of what was seen before and the calculations have a lot to do with the gradient descent algorithm.

[8]Bold terms in the context of neural networks will represent matrix notation.

For reasons of objectivity of the work, the mathematical details of this process will not be addressed here, but the reader can consult any introductory material on NN, such as Rojas and Rojas (1996). However, a qualitative description can be made. Very succinctly, it takes place by dividing the training set into small groups (called *mini-batches*) and passing them to the neural network, which has its parameters $\mathbf{\Theta}$ randomly initialized. We say that the algorithm has completed an *epoch* when it uses all these groups once. This pass is the forward propagation commented above. All results from each logistic unit are saved to be used in the backpropagation step.

When reaching the end of the network with a sample, the error between the prediction made by the last layer and the expected result is evaluated, as well as how much each logistic unit contributed to it. This is done by computing the cost gradient in that layer. From there, the algorithm evaluates how much each unit of the previous layer contributed to the error, also using the gradient, and does so until it reaches the beginning of the network. This measurement of errors takes into account the results of each unit generated and stored in the forward propagation process. Using all these errors, the algorithm performs a gradient descent to find the parameter values that minimize the errors made by the network units and adjusts them accordingly. This process is repeated for all samples of a mini-batch, then for all mini-batches of the training set and repeats everything again in a previously defined amount of epochs. The number of epochs is chosen so that the algorithm can converge to an optimized result and varies in each situation. Because of this, the epoch is a hyperparameter that directly influences the performance of the model.

In the discussion about equation 3.3, the sigmoidal function was chosen for $g(z)$. However, this is not the only possible choice. In fact, other common choices for the activation function are the hyperbolic tangent, Rectified Linear Unit (ReLU), and Softplus (see Figure 73). Among these, ReLU proves to be particularly advantageous in this kind of network compared to sigmoidal and hyperbolic tangent. This is because when using the latter two, the gradients of the model practically cancel out as the backpropagation algorithm approaches the earlier layers, which causes a problem known as the “vanishing gradient problem”. In addition to avoiding this problem, computations performed with ReLU are much faster due to its linear nature.

It is possible to combine multiple activation functions in the same network. In the case of neural networks for classification problems, there are two more functions that are useful when used in the output layer: ArgMax and SoftMax. As an example, suppose, in the network of Figure 42, that all layers use ReLU as the activation function, except for the last one. As currently constructed, the network returns the "probabilities"[9] that a sample is *non-planet* in the first unit of the output layer and *planet* in the second unit. SoftMax takes all the values from each logistic unit that the output layer receives from the previous layer and normalizes them to values between 0 and 1. The equation that describes it is

$$\boldsymbol{a}^{(\boldsymbol{L}+1)} = \left[\sum_i \exp\left(a_i^{(L)}\right)\right] \exp\left(\boldsymbol{a}^{(\boldsymbol{L})}\right) \tag{3.11}$$

In this expression, $L + 1$ is just a convenient notation to indicate that SoftMax performs its calculations based on the values received by the output layer and returns something extra. In a didactic way, we can think of an additional layer beyond the output layer whose function is to process the "raw" values received from the latter before displaying the final results. This caveat arises because the network representation in Figure 63 shows a relationship of two logistic units in the output layer to three units in the previous layer, which would be incompatible with equation 3.11 considering that the network only returns two values.

Since the classes are mutually exclusive, SoftMax ensures that the sum of these probabilities is 1. Additionally, this function can be used in the training phase of the network, as it has a convenient derivative for use in backpropagation. This is not the case, for example, when using the ArgMax activation function in the final layer. It works with the same analogy as SoftMax, but returns the value 1 for the most probable class and 0 for the others. Due to this characteristic, it is only used after the network is trained. With ArgMax, the network can provide its classifications in a more direct and assertive manner in a context where the model is already in its real-world application phase.

3.2 A Brief Introduction to Deep Learning

Neural networks given in the previous section are the most basic cases of their interpretation and application. In this section, we briefly explore more advanced and robust NN algorithms, that enters in the realm of what is known today as *Deep Learning* (DL).

[9]In quotation marks because these values depend on how the network's parameters were initialized. This is because the algorithm can converge to a different minimum in each training, depending on its starting point in parameter space. A more appropriate term for this value might be the "confidence" that a model has, based on what it has learned and how it has learned (depending on the configuration of its hyperparameters), that a sample belongs to a given class.

3.2.1 Definition

As mentioned earlier, deep learning is a sub-field of machine learning. Neural networks serve as a conceptual bridge between these two branches, since more complex networks than the ones we saw before can have a larger number of parameters to be processed during training and application. Based on this, it is conceivable to have a primitive notion of what *deep learning* is, for starts, which involves a more extensive and complex use of machine learning. A clear consensus does not seem to have been reached in the literature, though. For example, in the decade of 1990, the concept of deep learning would refer to a neural network with more than two hidden layers. Today, with significantly greater computational power, deep learning would involve a network with dozens or even hundreds of hidden layers.

Although this conception indicates a correct line of thought, it is limited. In fact, deep learning is indeed a more complex concept, but it is not merely about operational complexity. It is important to consider the conceptual complexity inherent in the observed capabilities of deep learning algorithms. Several authors, in an attempt to provide a coherent definition for this concept, directly or indirectly agree that deep learning involves gaining knowledge from a hierarchy of features (Bengio; others, 2009; LeCun *et al.*, 2015; Bengio *et al.*, 2013; SCHMIDHUBER, 2015). However, this consensus is challenged when we question the definition of features. We have seen that features represent the characteristics of the data of the problem, but this already highlights the vague nature of the definition, as the data depends on the application.

On the other hand, some argue that deep learning algorithms, in addition to hierarchical learning, acquire multiple levels of representations that correspond to different levels of abstraction. In this case, these levels would form a hierarchy of concepts (Deng *et al.*, 2014). Once again, we face the problem of establishing definitions with vague terms, as the meaning of "abstraction" remains unclear. These definitions, naturally, have their merits but also present vague aspects. Nevertheless, there is a particularly cohesive definition presented by Zhang *et al.* (2018). The authors consider a modeling of the nature of learning itself[10]. In simplified terms, they argue that *learning* is a process of establishing the relationship between two or more variables or instances. Then, they define *deep learning* as a process not only for learning the relationship between variables but also for acquiring the knowledge that governs that relationship and the knowledge that gives meaning to that relationship. In practical terms, the abstraction capability (as explained in the last lines of the previous paragraph) of deep learning algorithms

[10]See Lin and Zhang (2004), Zhang *et al.* (2005), Zhang *et al.* (2018) for more information.

enables a better performance in the tasks they are designed for, compared to more common machine learning algorithms in general. This improvement in performance becomes gradually evident as the complexity of the task increases.

3.2.2 Convolutional Neural Networks [11]

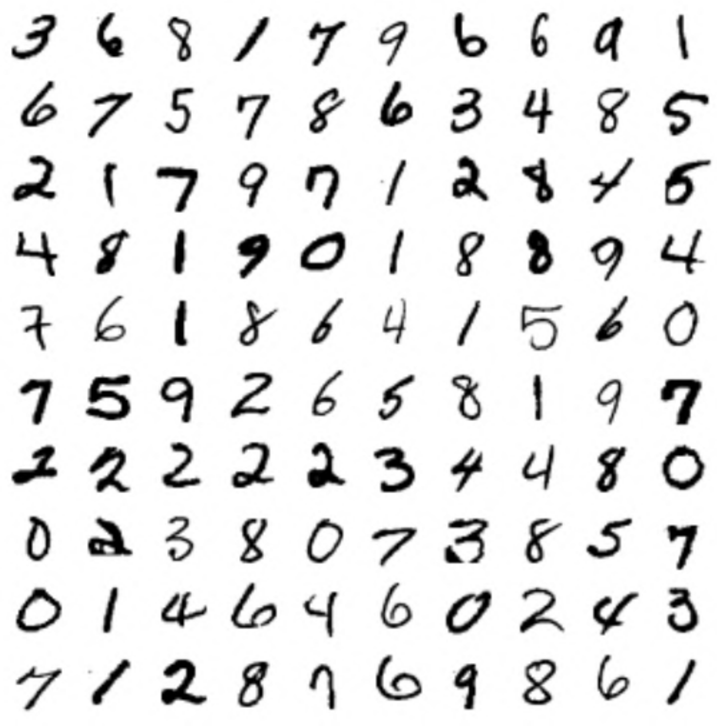
As commented in section 3.1.2, biological neural networks served as inspiration for artificial ones. Both aim to receive and process information through neural synapses or by operating an activation function. In order to recognize patterns in images, artificial neural networks began to be used to mimic the specific behavior of the visual cortex in living beings, as was the case with the neocognitron algorithm (Fukushima, 1980). A classic work in the field of pattern recognition in images is the one by LeCun *et al.* (1998), who introduced the LeNet-5 network architecture for recognizing handwritten numerical digits, trained on the NIST image database. This database was later updated with more data and improvements and became known as MNIST, which now contains over 70,000 images. It has become a classic reference and is widely used as a "safe haven" for various algorithm tests and educational purposes. Figure 43 shows some samples of the images.

Each MNIST image has a resolution of 28×28 pixels (LeCun *et al.*, 2009). With the tools we have today focused on machine learning in general, such as Scikit Learn (Pedregosa; others, 2011), TensorFlow (ABADI *et al.*, 2015) and PyTorch (Paszke; others, 2019), it is possible to build a fully connected neural network, or dense NN, quite easily, as explained in subsection 3.1.2, to recognize the digits contained in these images. In this specific case, each pixel would represent a feature in the network and we would then have 784 features in the input layer, each with a value from 0 to 1 representing a shade of gray. Even so, we are talking about images with a very low resolution, compared to the standards of images present in our daily lives. A slightly larger image, say 100×100, would need a network with 10^4 inputs to do the same task. The number of parameters, of course, will depend on the size of the network, but it is possible that more complex data will require a more complex network to perform well. This number can be significantly larger than the number of inputs.

This problem can be solved using convolutional neural networks, where the logistic units of a network are locally connected. In other words, instead of connecting each logistic unit

[11]Most of the content of this subsection was extracted and/or adapted form Géron (2022). The reader can refer to this reference for more details.

Figure 43 – NIST database samples



Source: Extracted from LeCun *et al.* (1998).

to all units in the next layer, as shown in Figure 42, we impose a restriction so that each unit is connected to only a small number of other units. In addition to this characteristic, an even more distinctive feature of convolutional networks is the ability to work with data in matrix format. That is, instead of using 784 features for MNIST images, we can directly use a 28×28 matrix of values.

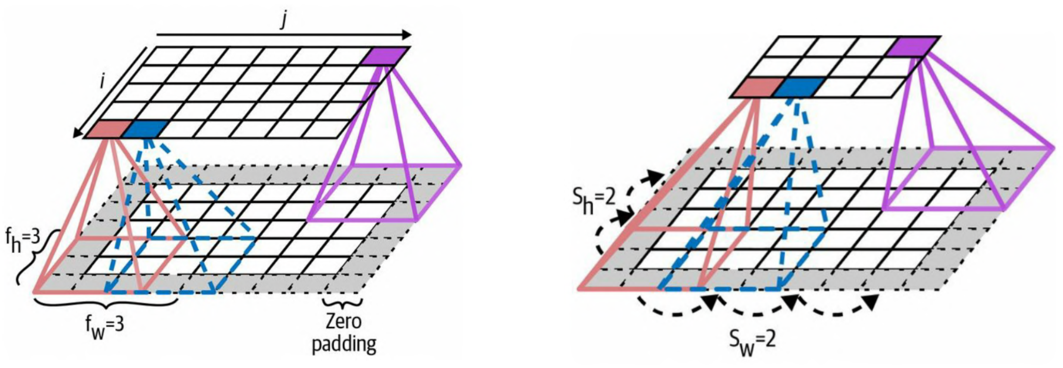
In this representation, it is easier to discuss the connections that elements in one layer make with those in the previous layer than vice versa. A visual representation of these connections is shown in Figure 44(a). Note that a unit located in a row i and column j of a given layer is connected to the units in the previous layer located between rows i to $i - 1 + f_h$ and columns $j - 1 + f_w$, where f_h and f_w are the height and width of what is known as the local receptive field[12]. The idea here is similar to that of a typical neural network. The logistic units in a layer that are contained within a local receptive field will connect to a single unit in the next layer through weights and activation functions. A *filter*, *convolution kernel* or just *kernel* is a small matrix of weights that is used to extract features from a local receptive field. It is common to think that the units in the next layer will learn to analyze specific elements from the

[12]A note to physicists: the concept of field used in any computational discussion in this work is purely a region in which each point has an associated value. I include these terms as they are used in the specific literature of machine learning in general.

previous layer. Additionally, it is usual, but not mandatory, to add units with a value of zero around the input units so that the next layer has the same height and width as the previous layer. This is known as *zero padding* or simply *padding*, which explains the -1 when describing the window of receptive fields. The term “convolution” comes from the convolution operation in mathematics (Smith; others, 1997), although the actual concept being slightly different.

Figure 44 – Connections between layers of a CNN

(a) (b)



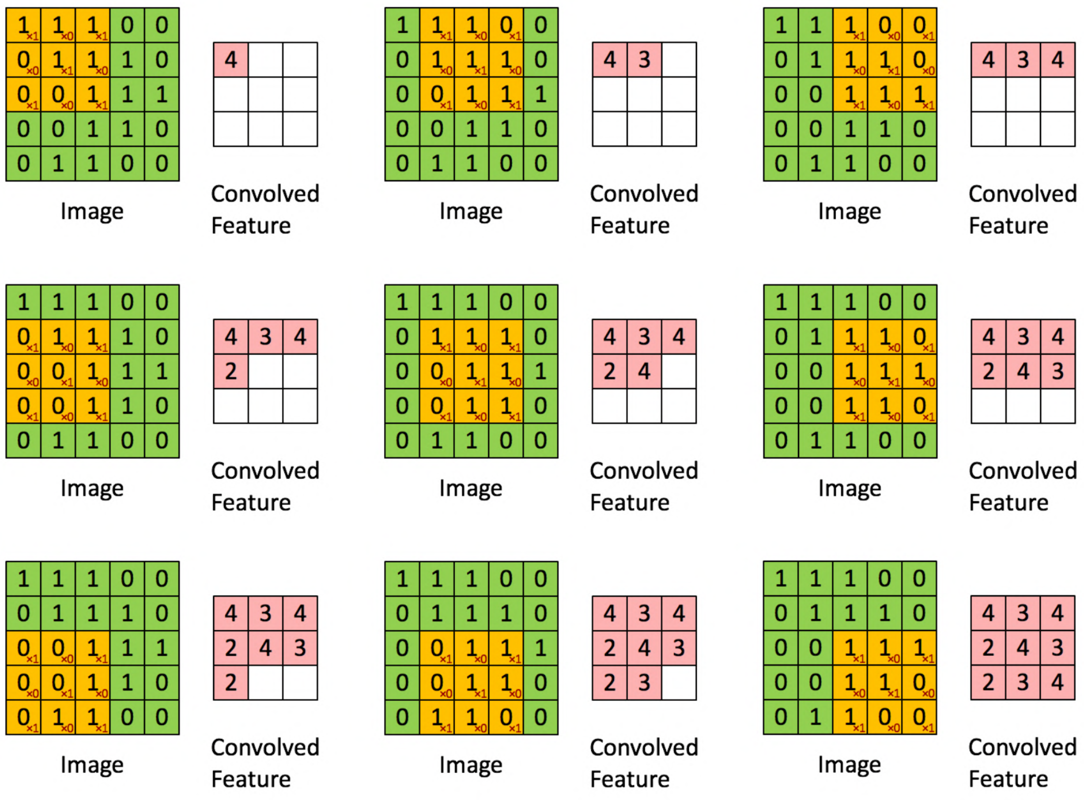
Source: Adapted from Géron (2022)

In case it is convenient to reduce the size of the next layer in relation to the previous layer, it is possible to add a separation between the windows of the fields, as shown in Figure 44(b). The horizontal and vertical spacing are referred as s_h and s_w, respectively, and are called *strides*. They may be equal or not. By this point of view, a logistic unit located in row i and column j of a layer connects with the units of the previous layer within the window of the kernel. This window goes in the range of rows from $i \times s_h$ to $i \times s_h - 1 + f_h$ and of columns from $j \times s_w$ to $j \times s_w - 1 + f_w$. This provides an expressive reduction in the computational complexity of the algorithm.

Naturally, we can observe patterns that repeat themselves in various areas of an image, such as edges, lines, curves, among others. This is interesting because the features that the network learns in a given area of the input image can be used in other parts of it. As an example, consider the way we represent the MNIST images. Each pixel has a value between 0 and 1 and so the weights of each logistic unit can also be represented by a small image that has the size of its associated receptive field. These are the already mentioned kernels.

A generic example is given by the Figure 45. It shows the process of creating these features through a given kernel based on an arbitrary image. A complete layer of logistic units that use the same kernel gives us a *feature map*, which highlights the areas in an image where that kernel has been activated the most.

Figure 45 – Formation of a new feature by convolution

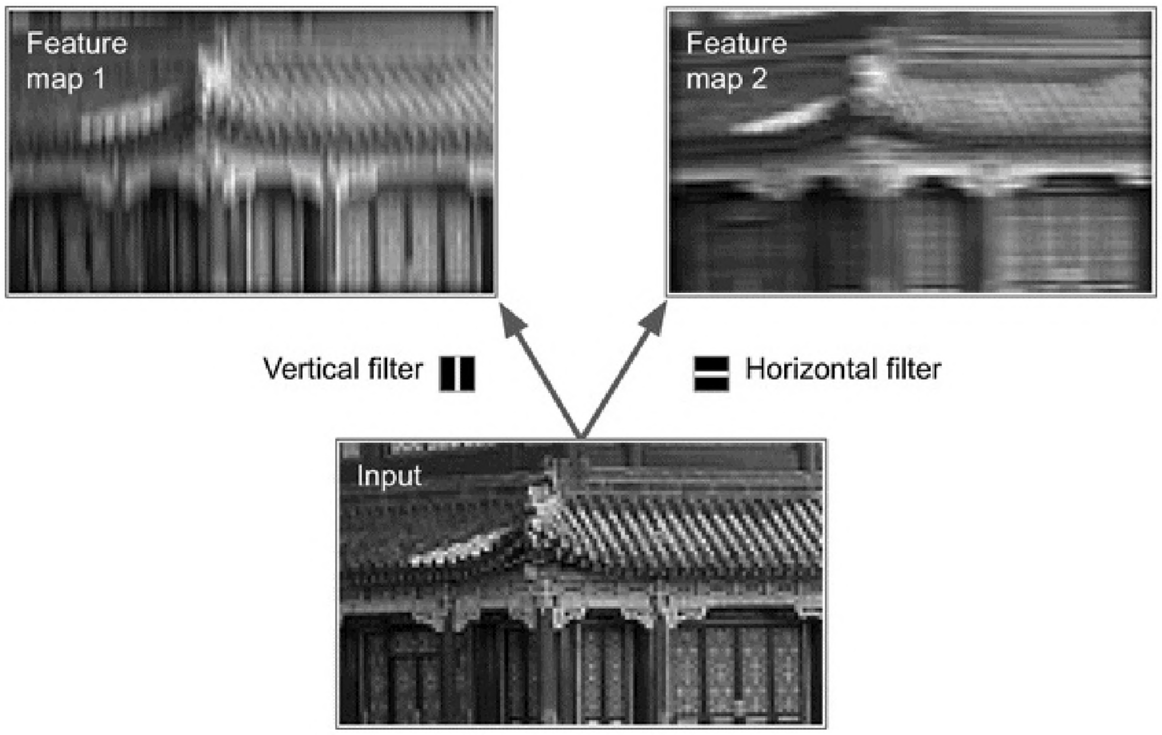


Source: Ng and others (2017)

As an example, consider Figure 46, where a filter 7×7 has been applied to the image. The kernel on the left has the central column with values equal to 1 (white) and the others equal to zero (black). The kernel on the left has the center line filled with 1's and the others with zeros. Through this, it is easy to notice the vertical and horizontal patterns for the left and right kernels, respectively. It is worth remembering that the best kernels will be automatically learned by the network, so that the upper layers can learn to combine them into even more complex patterns, and so on.

The images we usually deal with in our daily routine are in color, and this gives a whole new perspective in relation to CNNs. Briefly, this new perspective is due to the fact that pixels are made up of a combination of primary colors. A color *channel* is a matrix of the same size as the original image, but its values correspond only to the intensity of one of these primary colors. Some references call this matrix as a grayscale image. The color image is then generated by combining these channels (Gonzalez; Woods, 2018). For simplicity, the functioning of CNNs introduced so far took into account only 2D convolutional input and output layers, but their real

Figure 46 – Examples of feature maps



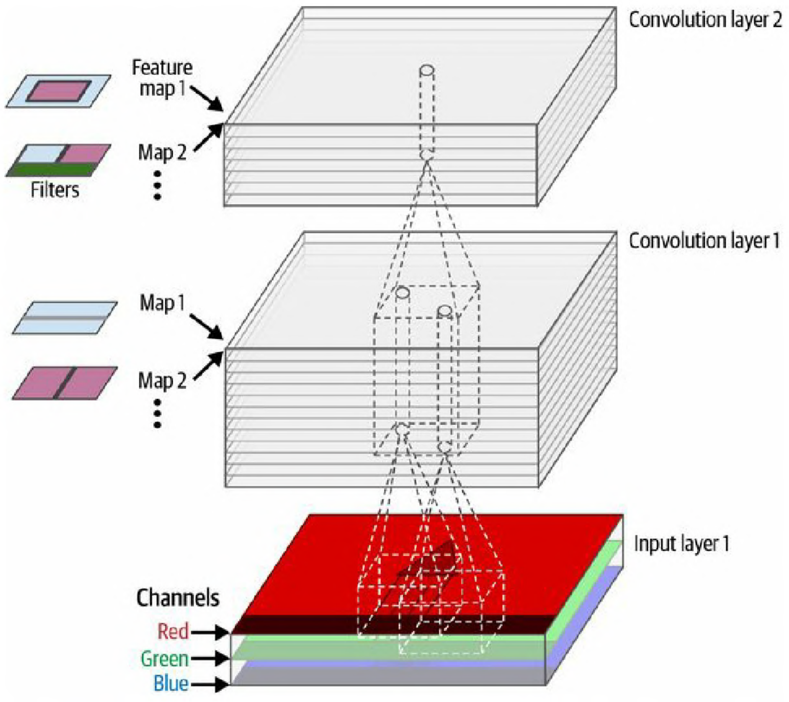
Source: Adapted from Géron (2022)

power comes from the fact that it is not only possible, but very common, to work with multiple kernels and build a feature map from each of these kernels. A common representation of images is based on RGB channels (*Red, Green, Blue*). From there, a more accurate representation of convolutional layers with their kernels for this example is given as in Figure 47.

We can use more channels besides RGB. In theory, we can use as many channels as we want. Interesting situations where more types of channels are explored are in multispectral and hyperspectral images, which essentially exploit the capture of image data in wavelengths beyond the visible range in the electromagnetic spectrum (Verhoeven, 2018). This type of image is particularly useful in astrophysics studies for several factors. Figure 78 shows our Sun seen in different wavelengths.

Another interesting example is given by Gonzalez and Woods (2018) and depicted in Figure 48, which shows an area of Washington D.C. In it, images *a* to *c* show channels in the (visible) red, green, and blue (RGB) bands, respectively. Image *d* is in the near-infrared (IR) range. Observing in this spectrum band is particularly useful for analyzing the biomass of the region. Thus, it is possible to combine RGB channels with the infrared channel to emphasize vegetation areas. Image *e* is a combination of the IR, green, and blue channels, while image *f* is a combination of the red, IR, and blue channels. The possibility of combining channels is

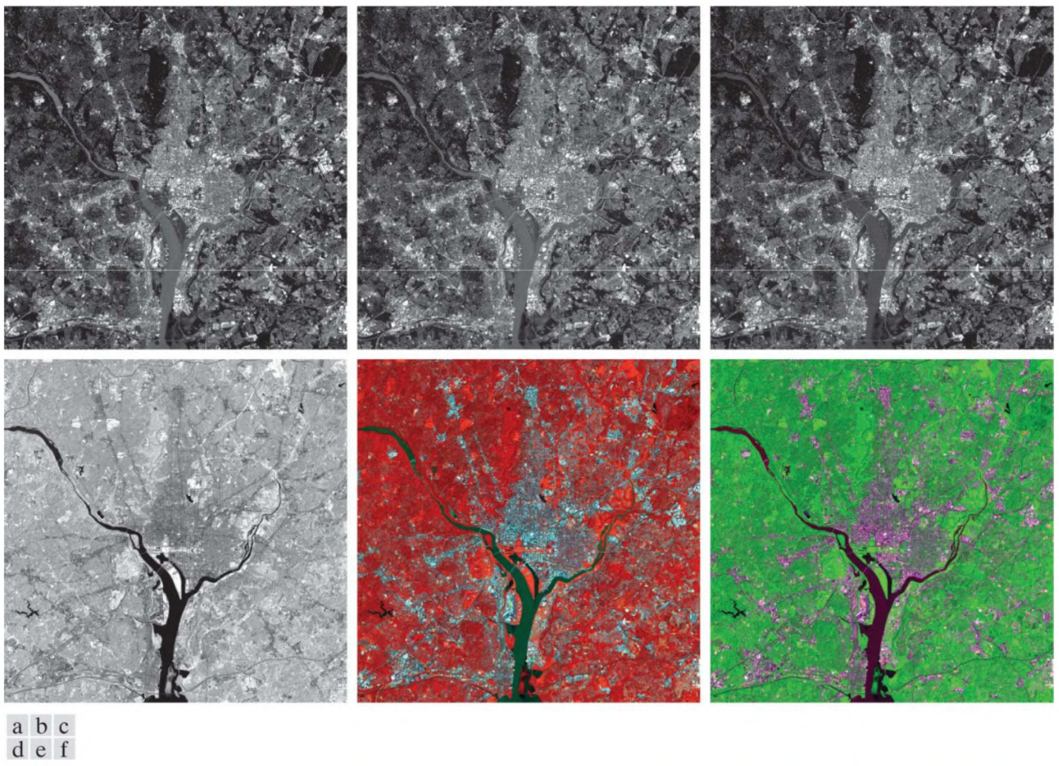
Figure 47 – Examples of convolutional layers with multiple kernels



Source: Adapted from Géron (2022)

promising in the context of temporal series imaging, as explained in Chapter 4.

Figure 48 – Example of multispectral images



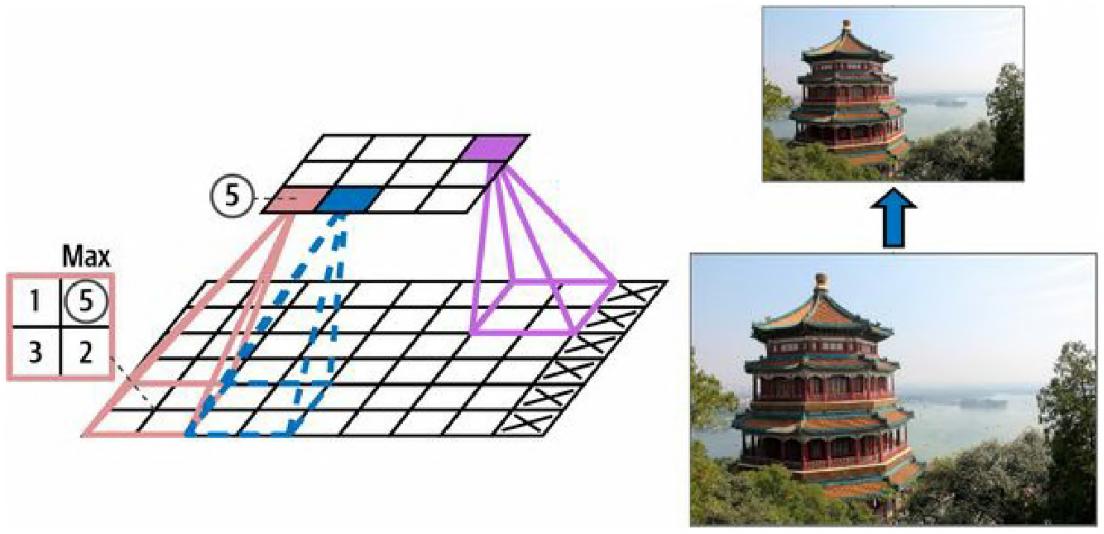
Note: The authors credit NASA for the multispectral images.
Source: Adapted from Gonzalez and Woods (2018).

As mentioned before, the fact that all the logistic units in a feature map share the same parameters significantly reduces the number of parameters in the model. The patterns that the network learns to recognize in a given region can be used in any other region. However, even with this improvement in performance with fewer parameters, CNNs still require a significant amount of computer memory, especially during training, where the backpropagation algorithm needs to access all the intermediate values in the forward propagation.

Exemplo 1 *A concrete example given by Aurlien Géron in "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" is as follows: a convolutional layer with 200 kernels of size* 5×5*,* $s_h = s_w = 1$*, and same padding will require* $(5\times5\times3+1)\times200 = 15,200$ *parameters to analyze an RGB image (three kernels) with a resolution of* 150×100*. The "+1" is due to the bias term. However, each of the 200 feature maps contains* 150×100 *logistic units, and each of these units needs to calculate a weighted sum of its* $5 \times 5 \times 3 = 75$ *inputs, resulting in a total of 225 million floating-point multiplications. If the feature maps are represented by 32-bit floats, then the output of a convolutional layer will occupy* $200 \times 150 \times 100 \times 32 = 96$ *million bits, which is approximately 12 MB of RAM. This is for just one instance. If we consider 100 of these instances, then the layer will use about 1.2 GB of RAM. To produce an output of the same size with a fully connected neural network, we would need* $200 \times 150 \times 100$ *logistic units, each connected to all* $150 \times 100 \times 3(+1)$ *inputs. This would result in approximately 135 billion (*10^9*) parameters.*

CNNs are much more optimized and ideal for this type of task compared to standard NNs. Nonetheless, the computational cost is still relatively high, especially for large databases. Fortunately, the computational burden of CNNs can be reduced somewhat through the use of *pooling layers*. They are responsible for sub-sampling the original image, effectively shrinking the input. This results in lower computational load, reduced memory usage, and fewer parameters. Having fewer parameters is desirable to limit the risk of overfitting. These layers follow the same principle as the convolutional layers. Each logistic unit in a pooling layer is connected to the outputs of the units within a receptive field of the previous layer. Here, we also have control over the size of this field, the strides, and the type of padding. The main difference is that the logistic units in the pooling layer do not have weights. They simply aggregate information from the previous layer through a predefined aggregation function, such as maximum or average value within the receptive field. An example of this can be seen in Figure 49.

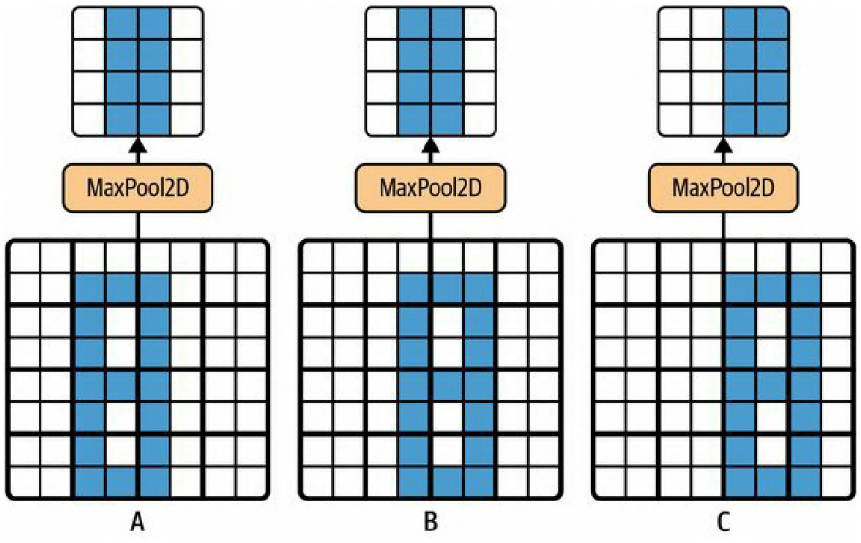
Figure 49 – Example of a pooling layer



Source: Adapted from Géron (2022)

Pooling layers still have the characteristic of introducing some level of invariance to small translations of elements in the image or image region being analyzed. In the example of Figure 50, where a 2×2 kernel was used with a stride[13] of 2, images B and C had their content shifted by 1 and 2 pixels compared to image A, respectively. The pooling layers of A and B are

Figure 50 – Example of invariance under small translations of the pooling layers



Source: Adapted from Géron (2022)

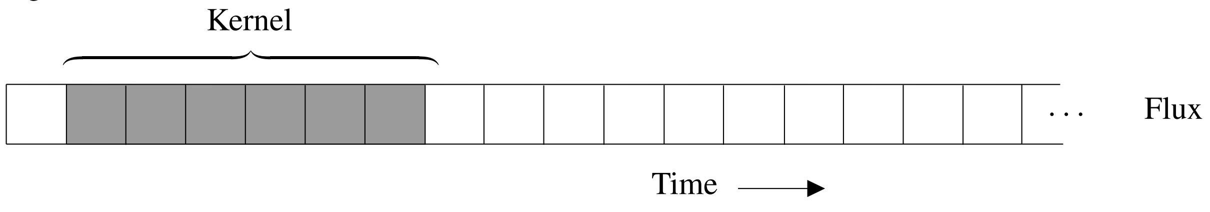
identical, but that of C is different. Note that, relative to the position of the content in A, the content of C shifted the most. In this case, the content shift in B is considered small and invariant. Similarly, a small degree of invariance can also be observed for rotation and scale variation. This

[13]When denoting only "stride equal to n" or simply "stride n", it is implicit that $s_h = s_w = n$.

is particularly useful in cases where the algorithm's prediction does not depend on these factors, such as in most classification tasks. A cat will still be recognized as a cat, even if it is shifted from a given reference, rotated, or scaled, for example. Despite all of this, a pooling layer has, by its nature, a destructive characteristic regarding input information. Therefore, it needs to be used wisely according to the project's requirements in order to achieve the desired objective.

When we talk about CNNs, it is usually implicit that we are referring to the processing of two-dimensional data such as images and videos in general. However, it is possible to apply all these CNN concepts to work with one-dimensional data (Kiranyaz *et al.*, 2021) and even three-dimensional data (QI *et al.*, 2016; Huang *et al.*, 2019; Bayu; Setyanto, 2022). In the case of 1D data, a straightforward example of application is in the use of time series, such as financial time series or the light curves we are working with. The operating principle is similar to what was explained earlier. There is a kernel, like the one shown in Figure 51, that scans the entire series, generating feature maps and so on.

Figure 51 – 1D Convolutional Kernel



Source: Self elaboration.

The advantage of this approach is the significant reduction in computational complexity: an image of size $N \times N$ with a kernel $K \times K$ will have a computational complexity of approximately $\sim O(N^2K^2)$, while a similar series, with dimensions N and K, will have a computational complexity of approximately $\sim O(NK)$. Thus, one of the proposals of this work, discussed in Chapter 5, is to evaluate whether the temporal series imaging method introduced in the following chapter is worth the computational cost it brings compared to using a 1D CNN.

3.3 Famous CNN Architectures

So far, what we have seen are basic concepts of CNNs. But with them, as we will see in Chapter 5, we can create models that are much better than the ones that rely only on the dense networks introduced in subsection 3.1.2. However, the attempt to achieve state of the art performance requires much more complex architectures, many of which uses techniques and concepts whose details goes far beyond the scope of this work. Still, we can have a superficial look at some of them. For the sake of space, only four models will be displayed not only for didactic purposes but because they were implemented and tested in this work. Each of them representing a benchmark at their time: the classic LeNet-5, AlexNet, ResNet and EfficientNet.

As mentioned earlier, LeCun *et al.* (1998) used the classic LeNet-5 for pattern recognition in images, specially handwritten digit recognition. Its architecture is described in Table 5. This is the arguably the most straightforward way of building a CNN "toy model" nowadays. We sure can change some of the values, make it larger or smaller, change some activation functions, and so on, but the idea is the same.

Table 5 – LeNet-5 architecture

Layer	Type	Maps	Size	Kernel size	Stride	Activation
In	Input	1	32x32	-	-	-
C1	Convolution	6	28x28	5x5	1	tanh
S2	Avg pooling	6	14x14	2x2	2	tanh
C3	Convolution	16	10x10	5x5	1	tanh
S4	Avg pooling	16	5x5	2x2	2	tanh
C5	Fully connected	120	1x1	5x5	1	tanh
F6	Fully connected	-	84	-	-	tanh
Out	Output	-	10	-	-	RBF

Note: "RBF" stands for Radial Basis Function (Broomhead; Lowe, 1988).
Source: Adapted from LeCun *et al.* (1998).

The decade of 2010 was marked by a significant improvement in the state of the art related to object detection and classification, mainly because competitions such as ILSVRC ImageNet challenge (RUSSAKOVSKY *et al.*, 2015). The competition, of course, consists on evaluating the candidates models' performance by measuring the error rate of them. The training and test images where extracted by online sources, standardized to the size (something in the range of 200×200 to 300×300) and manually labeled with one of the 1000 target objects. One thing they also considered was the *top-five* error rate, which means the number of test images for which the system's top five prediction did *not* include the correct answer. Over the course of only four years, between 2010 and 2014, the image classification error dropped from 28.2% to 6.7% (Howard *et al.*, 2018).

The AlexNet CNN architecture (Krizhevsky *et al.*, 2012), who won the 2012 ILSVRC challenge, is an example of how some modifications on the LeNet-5 can result in a very significant improvement on a model's performance, taking into account the state of the art at the time. As Table 6 shows, it is much larger and deeper. Its success, however, is not only due to the architecture itself, but also due to regularization techniques, such as dropout, data augmentation and *local response normalization*. With the exception of the data augmentation, which, in the context of this work, consists only on making a horizontal flipped copy of a light curve and putting them into the dataset, and dropout, which consists on “turnin off” (drop) some neurons of the network to avoid overfitting, the other techniques cited from hereon will not be discussed in this work, but the reader can refer to Géron (2022, p.392-399, 500-501) for more details.

Table 6 – AlexNet architecture

Layer	**Type**	**Maps**	**Size**	**Kernel size**	Stride	Padding	Activation
In	Input	3 (RGB)	227x227	-	-	-	-
C1	Convolution	96	55x55	11x11	4	valid	ReLu
S2	Max pooling	96	27x27	3x3	2	valid	-
C3	Convolution	256	27x27	5x5	1	same	ReLu
S4	Max pooling	256	13x13	3x3	2	valid	-
C5	Convolution	384	13x13	3x3	1	same	ReLu
C6	Convolution	384	13x13	3x3	1	same	ReLu
C7	Convolution	256	13x13	3x3	1	same	ReLu
S8	Max pooling	256	6x6	3x3	2	valid	-
F9	Fully connected	-	4096	-	-	-	ReLu
F10	Fully connected	-	4096	-	-	-	ReLu
Out	Fully connected	-	1000	-	-	-	Softmax

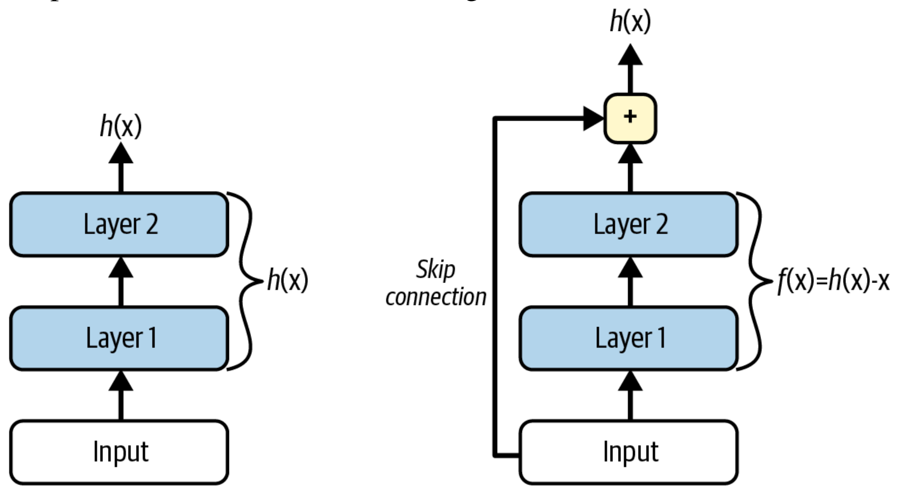
Source: Adapted from Krizhevsky *et al.* (2012).

Other architectures gained attention for their performance along the years on the ILSVRC challenges, such as GoogLeNet (Szegedy *et al.*, 2015), VGGNet (Simonyan; Zisserman, 2014), ResNet (He *et al.*, 2016), Xception (CHOLLET, 2017), SENet (Hu *et al.*, 2018). All of them inspired many other architecture variants, including variants of themselves. For instance, SENet not only uses and extends inception networks and ResNets, but also boosts their performance. ResNet, on the other hand, brings some remembrance from GoogLeNet, and so on.

ResNets serves as inspiration and basis for many modern architectures. With originally 152 layers, they confirmed a trend that was being seen in the middle of the last decade: CNNs in general were getting deeper and deeper, with fewer parameters. This introduced the usage of the term *extremely deep CNNs*. The availability to train such a network rely on something called *skip connections* or *shortcut connections*. This means that the signal that arrives into a layer is also added to the output of a layer above. Figure 52 compares a regular signal flow with a skip flow.

We saw on subsection 3.1.1 that the goal, when training, is to target a hypothesis function $h_\theta(\boldsymbol{x}^{(i)})$ with the minimum error possible. By adding a skip connection, we are adding

Figure 52 – Skip connection for residual learning



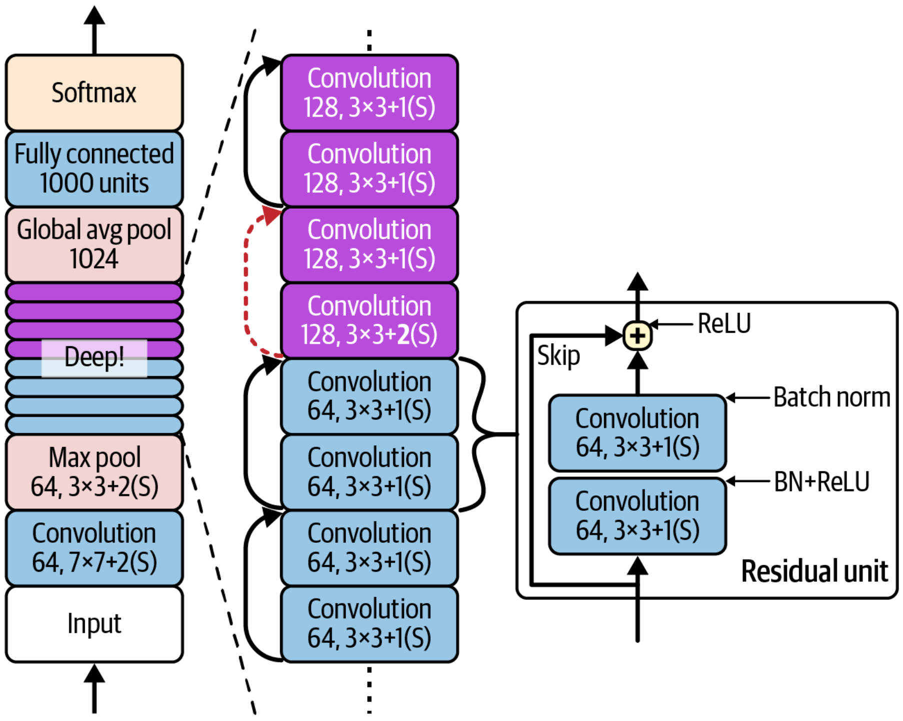
Source: Géron (2022, p.506)

the input $\boldsymbol{x}^{(i)}$ to the output of the network. This will make the network now target a new hypothesis function $f(\boldsymbol{x}^{(i)}) = h_\theta(\boldsymbol{x}^{(i)}) - \boldsymbol{x}^{(i)}$, instead of only $h_\theta(\boldsymbol{x}^{(i)})$. This is called *residual learning*, so then the name of the architecture.

With enough skip connections, the network can start making progress even if several layers have not started learning yet. This also helps to overcome the vanishing gradient problem (Hochreiter, 1998; Borawar; Kaur, 2023), since the signals can cross the whole network. We can organize the parts of a *deep residual network* by a stack of *residual units*, where each residual unit is a small neural network with a skip connection. Figure 53 shows a representation of the ResNet architecture. From it, we can see (red-dashed arrow) that some specific inputs cannot be added directly to the outputs of a residual unit, since the number of feature maps doubles its size and their height and width are halved and, therefore, the input and output do not have the same shape. This can be overcomed by adding a convolutional layer in the middle of the skip connection with the right specifications.

One of the most noteworthy architectures from recent years is the EfficientNet (Tan; Le, 2019). It is based on a technique called *compound scaling*, which by fixing a compound coefficient ϕ, the depth, width and resolution are scaled by α^ϕ, β^ϕ and γ^ϕ, respectively, with $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$ and $\alpha \geq 1, \beta \geq 1, \gamma \geq 1$. The authors shows that this approach is more effective than scaling each dimension independently. They explain that ϕ is a user-specified coefficient that controls how many more resources are available for model scaling. It is a logarithmic measure of the compute budget in the sense that if the compute budget doubles, then ϕ increases by 1. This gives the relation that the number of floating-point operations available for training is proportional to 2^ϕ. Also, α, β and γ indicates how to assign these extra resources to the network

Figure 53 – ResNet architecture



Source: Géron (2022, p.507)

width, depth and resolution, respectively.

The authors developed their baseline network by performing a multi-objective neural architecture search that optimizes both accuracy and FLOPS[14] (Tan *et al.*, 2019). At the end of their process, they had a computational efficient network which they called the *EfficientNet-B0 baseline*, described by Table 7. It uses mobile inverted bottleneck MBConv layers instead of regular convolutional layers (Sandler *et al.*, 2018; Tan *et al.*, 2019) and squeeze-and-excitation optimization (Hu *et al.*, 2018).

The compound scaling method proposed by them was applied at this point. Once the base network was created, they assumed the double of resources by fixing $\phi = 1$ and performed a small grid search for α, β, γ. They found that, for the EfficientNet-B0, the best values were $\alpha = 1.2$, $\beta = 1.1$ and $\gamma = 1.15$. Then, they fixed α, β, γ as constants and scaled up this baseline network to create larger architectures, named EfficientNet-B1 to EfficientNet-B7, for increasing values of ϕ.

More recently, the training bottlenecks of EfficientNets were analysed to improve the architectures even further (Tan; Le, 2021). By progressively increasing image size during training and dynamically adjusting regularization, the authors created a new and even more

[14]FLOPS, or FLoating point Operations Per Second, is a measure of computer performance. It measures the speed of a computer in operations per second, especially arithmetic operations involving floating-point numbers (Smith, 1999, p.526).

Table 7 – EfficientNet-B0 baseline network

Stage i	**Operator** $\hat{\mathcal{F}}_i$	**Resolution** $\hat{H}_i \times \hat{W}_i$	**#Channels** $\hat{C}_i$	**#Layers** $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Note: As the authors indicate, each row describes a stage i with $\hat{L}_i$ layers, with input resolution $\langle \hat{H}_i, \hat{W}_i \rangle$ and output channels $\hat{C}_i$. Their notations where kept.
Source: Tan and Le (2019).

powerful family of EficientNets, called EficientNetsV2. The detais of go far beyond the scope of the proposal of this section, but the reader may refer to their paper for more information.

3.4 Transfer Learning

Transfer learning is a technique that consists on the reuse of a pre-trained network from a previous problem on a new one (Dai *et al.*, 2007). The reason to do this is that *Deep Neural Network* (DNN) in general can be trained with little data, when we comparing with the amount of data required for the full training process. Since a pre-trained network will have partially adjusted wights on its neurons, the amount of computation required to achieve a decent performance with the new data might be reduced as well (Torrey; Shavlik, 2010).

The relevance of this topic for this work relies on the fact that there is arguably little exoplanet data if we take into account the dozens of thousands, or even millions of instances used to train models on other tasks in order to achieve state-of-the-art performance. Examples of such other applications are general image classification (Jiao; Zhao, 2019; He, 2020; OTTONI *et al.*, 2023), object detection (Masita *et al.*, 2020; WU *et al.*, 2020; KAUR; SINGH, 2023), medical imaging (Chakraborty; Mali, 2023; Li *et al.*, 2023; SAILUNAZ *et al.*, 2023) and so on.

There are several transfer learning techniques and approaches available (Torrey; Shavlik, 2010), but, in the context of deep learning, the most common one is using a model trained with a highly relateble dataset to the one we have and then finetune it with our data (Iman *et al.*, 2023). Although prone to catastrophic forgetting, i.e., the tendency of for knowledge of the peviously learned task(s) to be abruptly lost as information relevant to the current task is incorporated (Kirkpatrick *et al.*, 2017), it is very effective. The second most popular approach is freezing DNN layers of a pre-treined model and finetuning lateral fully connected layers (Iman *et al.*, 2023).

As mentioned in section 3.3, many powerful networks were trained with the ImageNet dataset for the ILSVRC challenge. These models can be used for both approaches described above, since they are easily accessible (Simon *et al.*, 2016). Considering that the training of such models from scratch can take days, even with powerful hardware such as clusters of GPUs/TPUs, downloading and using a public available pre-trained model can save a lot of time and resources if the transfer is successful.

3.5 Performance Metrics

All the algorithms used in this work serves for a specific purpose: classifying patterns in lightcurves as either *planet* or *non-planet*. The effectiveness of each algorithm, trained on the training data, is assessed by examining its performance on the test set data. To accomplish this, we assess the algorithm's performance using *performance measures*. This section provides a brief overview of some of these measures for binary classification.

When an algorithm performs classification on a given data, we can make four evaluations about the result. Taking our dataset as an example, if the label of a TCE is *planet* and the algorithm classifies it as such, then this result is a true positive (T_P). If the algorithm classifies it as *non-planet*, then this result would be a false negative (F_N). If the label of a TCE is *non-planet* and the algorithm classifies it in the same way, then this result would be a true negative (T_N). Finally, if this TCE is classified as *planet* by the algorithm, then the result would be a false positive (F_P). A false positive and false negative are, respectively, Type I and Type II errors in the context of statistical hypothesis testing (Shuttleworth; Wilson, 2008). These evaluations can be summarized in a confusion matrix, as describe in Table 8.

Table 8 – Generic confusion matrix

		Predicted	
		Negative	Positive
Original	Negative	T_N	F_P
	Positive	F_N	T_P

Source: Adapted from Géron (2022).

The most immediate performance measure in evaluating a classifier is its *accuracy*, which is the percentage of correct predictions achieved by the algorithm. Assuming the test was performed on a test set with m_t samples, then

$$\text{accuracy} = \frac{T_P + T_N}{m_t}. \tag{3.12}$$

In this particular case, we have that m_t is numerically equal to the sum of all the results, that is, $m_t = T_P + T_N + F_P + F_N$. The accuracy percentage of an algorithm should not be overestimated, especially in situations where the dataset is unbalanced. An unbalanced dataset is one where the number of samples from the classes considered by the model is significantly unequal. An example of this is our own dataset, where the *planet* class represents only 23% of the entire dataset[15]. This implies that a classifier algorithm programmed to only return negative results (*non-planet*), regardless of the light curve's characteristics, would be correct 77% of the time.

The concept of accuracy can be refined into a new metric when evaluating only the accuracy of positive predictions. This is referred to as *precision* and it is given by

$$\text{precision} = \frac{T_P}{T_P + F_P}. \tag{3.13}$$

In other words, precision tells us the portion of selected positives that are true. One metric that is always taken into account when discussing precision is the *recall* (also called sensitivity), given by

$$\text{recall} = \frac{T_P}{T_P + F_N}. \tag{3.14}$$

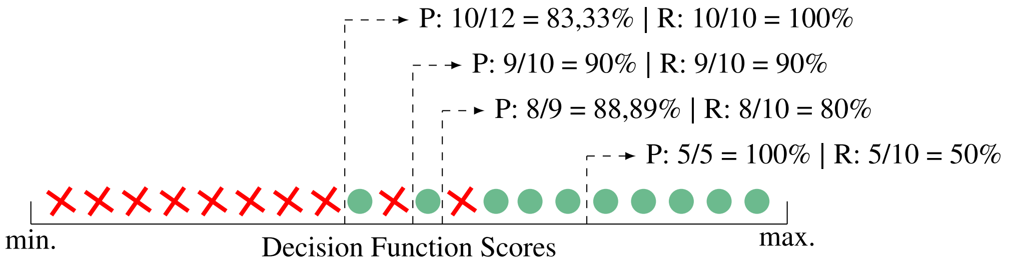
That is, the recall tells us the portion of true positives that were correctly identified relative to all the positives in the dataset. It is also referred to as sensitivity because it assesses the algorithm's effectiveness in successfully detecting positive results. A high value in both measures is important for the algorithm to perform well. However, increasing precision implies decreasing recall, and vice versa.

The algorithm determines whether an instance on the dataset belongs to the positive or negative class based on a score returned by a decision function. This decision is based on the decision boundaries discussed in subsection 3.1.1. If the value returned by the decision function is greater than an established limit, for example, then the instance is classified as positive. The choice of this limit directly affects the precision and recall values, as shown in Figure 54.

The trend of accuracy is to increase when the algorithm becomes more stringent with punctuation, but in some cases it may decrease, as is the case with the third threshold indicated

[15]The AFP and NTP labels, described in the Table 3, have been merged into a single *non-planet* class.

Figure 54 – Example of switching between Precision (P) and Recall (R) for different decision function score thresholds


P: 10/12 = 83,33% | R: 10/10 = 100%
P: 9/10 = 90% | R: 9/10 = 90%
P: 8/9 = 88,89% | R: 8/10 = 80%
P: 5/5 = 100% | R: 5/10 = 50%
min.
Decision Function Scores
max.

Source: Adapted from Géron (2022).

in the figure. In general, we have the freedom to choose this threshold in order to optimize the precision-recall trade-off that best suits the needs of the work. In the particular case of this study, a slightly higher recall over precision is preferred in order to reduce the loss of planets (especially in unexplored data). By choosing these values, we can plot precision against recall ($P \times R$) to evaluate the performance of models using these measures.

In order to compare two or more models on a task, it is often common to use one single metric called *F1 score*. It is given by the harmonic mean of precision and recall, so that

$$F_1 = 2 \times \frac{\texttt{precision} \times \texttt{recall}}{\texttt{precision} + \texttt{recall}}. \tag{3.15}$$

This metric is interesting, because the harmonic mean is more sensitive to low values. In other words, one can only have a high F_1 score if both precision and recall are high as well.

Another important metric is *specificity*. Just as recall can be understood as the algorithm's ability to correctly detect positives, specificity is the algorithm's ability to correctly detect negatives. It is defined by

$$\text{specificity} = \frac{T_N}{T_N + F_P}. \tag{3.16}$$

Thus, the specificity can also be called the true negative rate. With it, we can still derive the *False Positive Rate* (FPR) or *Fall-Out*, given by

$$\text{FPR} = 1 - \text{specificity} = \frac{F_P}{F_P + T_N}. \tag{3.17}$$

According to Figure 54, FPR increases as recall increases. A graph of recall, referred to as *True Positive Rate* (TPR) in this context, plotted against FPR can be generated to compare models in a similar way to precision-recall. The curve of this graph is known as the *Receiver Operating Characteristic* (ROC) curve.

All these metrics are strongly dependent on the threshold chosen, as illustrated in Figure 54, since it impacts directly a model's confusion matrix. There are two important measures that do not depend on a specific threshold: the *Area Under Curve* (AUC) from Precision vs Recall (PR) curves (PR-AUC) and from ROC curves (ROC-AUC). ROC curves analysis are the most common (He; Ma, 2013) and then it is customary to find in literature the term AUC implicitly related to the ROC curve.

Fernández *et al.* (2018, p.54) state that the ROC-AUC can be interpreted as the probability that the scores given by a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. In other words, ROC-AUC can be understand as a measure of a model's ability to distinguish between positive and negative cases. However, the authors argue that this metric can be unreliable under the presence of class rarity (FERNÁNDEZ *et al.*, 2018, p.55). This relates to the same accuracy problem exposed after equation 3.12: imagine how would be this problem, but with a dataset with only 0.1% of a positive class. In these cases with highly skewed data, PR curve analysis and PR-AUC are more recommended (Branco *et al.*, 2015).

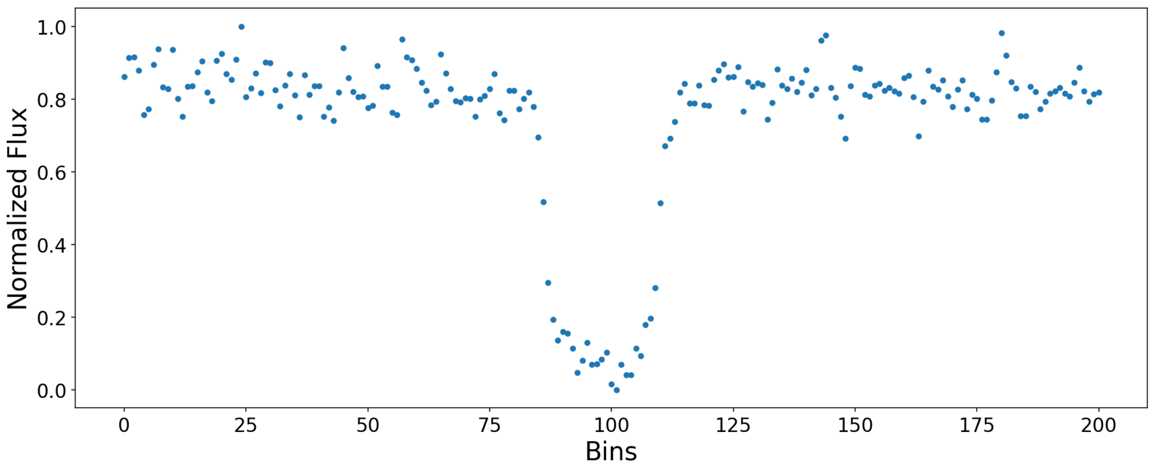
4 THE TIME SERIES IMAGING METHOD

Aiming for the rising success of deep learning, the work of Wang and Oates (2015) brought a new approach on working with time series using artificial intelligence algorithms. The technique introduced by the authors is based on the creation of three types of images through transformations made with the data of the processed time series. In the language used in section 3.2 and using the example from Figure 47, each generated image will be used as a channel that provides information about the analyzed time series. A fourth type of image was used in this work, which relies on the work of Eckmann *et al.* (1987). Let's now examine each of the types, along with examples already generated by the pipeline created in this work to transform the light curves treated as in section 2.3.

4.1 Angular Gramian Field

Consider a discrete time series $\boldsymbol{F} = \{f_1,\ f_2,\ \ldots,\ f_n\}$ of n real values. Let's take as an example an arbitrary local light curve already treated by the procedures of the section 2.3, given by Figure 55. In this case, $n = n_l = 201$ for local curves. The following procedure is analogous to global light curves, where $n = n_g = 2001$. Note that, due to the nature of the processing we did before, this light curve is already normalized so that its values are in the range of $[0,\ 1]$. This imaging technique requires an arbitrary time series to be rescaled so that its values fall within the range $[-1,\ 1]$ or $[0,\ 1]$. For completeness, we will refer to the rescaled time series as $\tilde{\boldsymbol{F}} = \left\{\tilde{f}_1, \tilde{f}_2, \ldots, \tilde{f}_n\right\}$.

Figure 55 – Arbitrary local light curve for imaging



Note: PC refers to *Planet Candidate*, that is, in this example we will be following the imaging of a light curve of a TCE that represents a planet.
Source: Self elaboration.

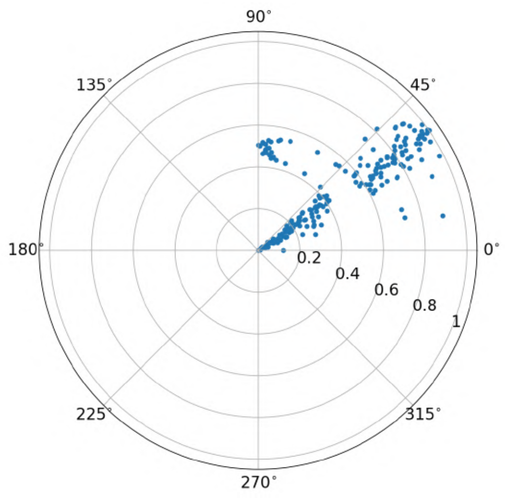
Thus, we can represent these values in polar coordinates following the equation 4.1:

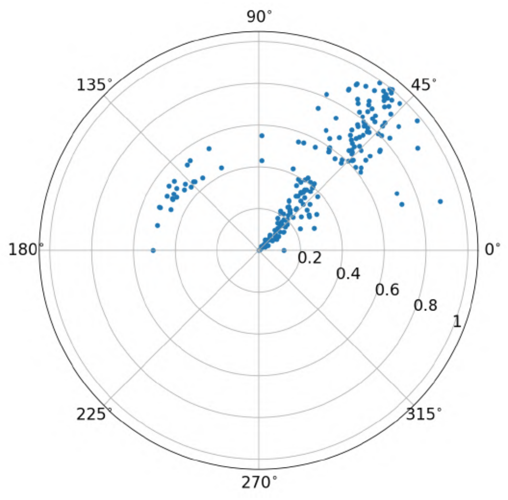
$$\begin{cases} \phi = \arccos\left(\tilde{f}_i\right), \quad -1 \leqslant \tilde{f}_i \leqslant 1, \quad \tilde{f}_i \in \tilde{F} \\ r = \frac{t_i}{N}, \quad t_i \in \mathbb{N} \end{cases}, \tag{4.1}$$

where t_i is the time associated with each measurement and N is a constant to regularize the interval of points in the polar coordinate system. Considering a unit radius in the polar plane and treating the curve through equation 2.3, r can be calculated by dividing the interval from 0 to 1 into n_l or n_g equal parts. For our example light curve, this transformation results in the graph shown in Figure 56. The interval $[-1, 1]$ was considered for comparison.

Figure 56 – Example of a light curve plotted in polar coordinates

(a) Normalization from 0 to 1 (b) Normalization from -1 to 1





Source: Self elaboration.

We can identify the temporal correlation between different points in the polar plane by leveraging trigonometric operations. Specifically, we can bild a matrix of values considering the sum and difference of the ϕ's just calculated. These matrices are referred to as the *Gramian Summation Angular Field* (GSAF) and the *Gramian Difference Angular Field* (GDAF), and they are defined as

$$\text{GSAF} = \left[\cos\left(\phi_i + \phi_j\right)\right] \tag{4.2}$$

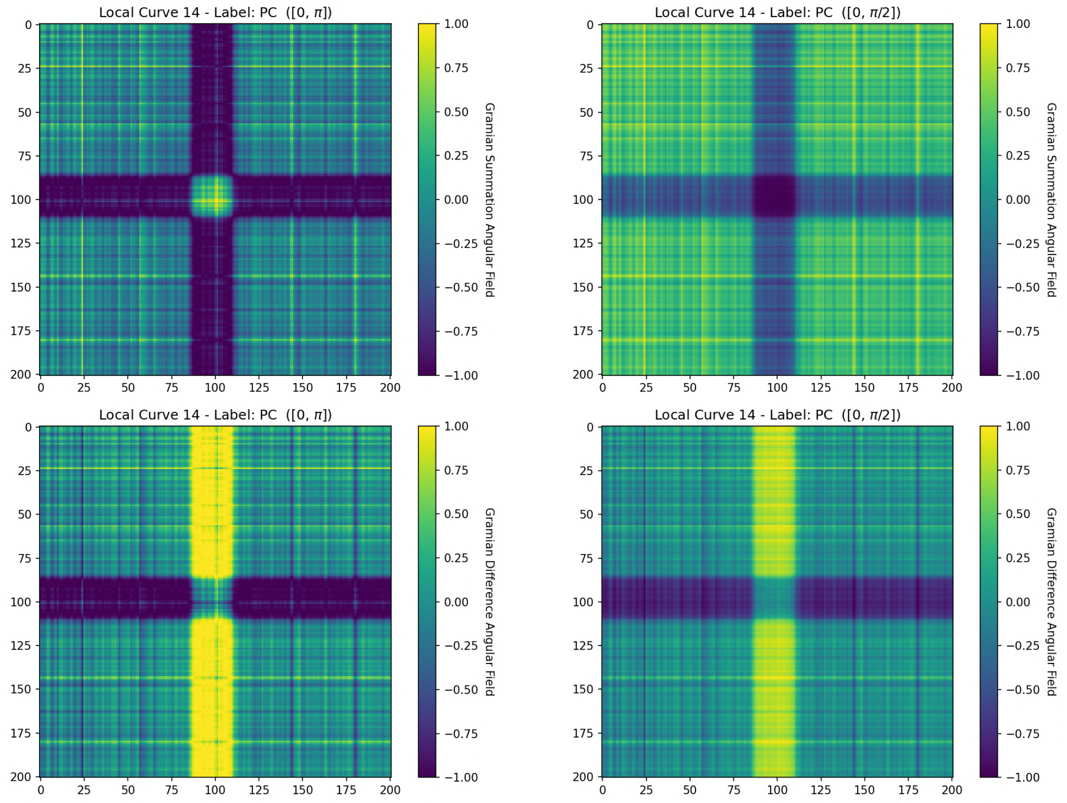
$$= \tilde{\boldsymbol{F}}' \cdot \tilde{\boldsymbol{F}} - \left(\sqrt{\mathbb{1} - \tilde{\boldsymbol{F}}^2}\right)' \cdot \sqrt{\mathbb{1} - \tilde{\boldsymbol{F}}^2}; \tag{4.3}$$

$$\text{GDAF} = \left[\text{sen}\left(\phi_i - \phi_j\right)\right] \tag{4.4}$$

$$= \left(\sqrt{\mathbb{1} - \tilde{\boldsymbol{F}}^2}\right)' \cdot \tilde{\boldsymbol{F}} - \tilde{\boldsymbol{F}}' \cdot \sqrt{\mathbb{1} - \tilde{\boldsymbol{F}}^2}, \tag{4.5}$$

where $\mathbb{1}$ is defined here as the unit vector $[1,\ 1,\ \ldots,\ 1]$. Examples of these fields generated for our example light curve can be seen in Figure 57.

Figure 57 – Examples of gramian angular fields



Note: The number associated of the light curve refers to the sequential number from the dataset.
Source: Self elaboration.

Wang and Oates (2015) comment that the Gramian Angular Fields (GAF) have their advantages. They preserve the temporal dependence of the series, as time increases as the position moves from the upper left corner to the lower right corner. The position in terms of i and j, such that the difference between i and j is equal to k, can be represented by $G_{(i,j||i-j|=k)}$, and it represents the relative correlation by superposition or difference of directions with respect to the time interval k. The main diagonal $G_{i,j}$ when $k = 0$ is the special case that contains the original angular values. Consequently, we can completely recover the light curve through it.

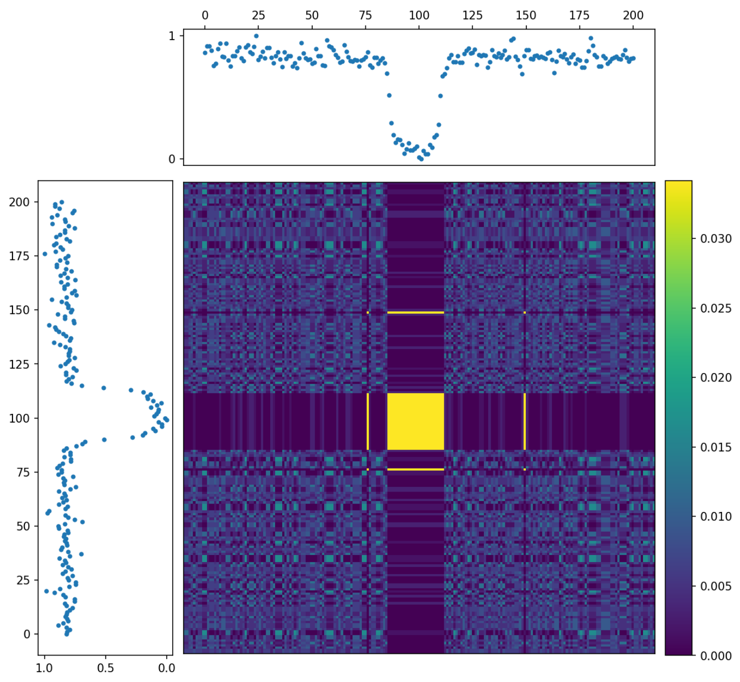
4.2 Markov Transition Field

Campanharo *et al.* (2011) use the technique of mapping a time series to a network in order to use network measures to determine properties of time series. Different concepts for this purpose are discussed, one of which is the transition of probabilities. Wang and Oates (2015) propose a similar approach to this work but extend the idea by sequentially representing the transition probabilities of Markov to preserve information in the time domain. These probabilities are derived from the theory of Markov chains (Gagniuc, 2017).

Wang and Oates (2015) introduce a state transition matrix as follows. Consider a time series $\boldsymbol{F} = \{f_1, f_2, \ldots, f_n\}$ of size n. We begin the transformation by aggregating each value of the time series into Q quantiles. Each of these quantiles is a state in the terminology of the Markov model. We then want to construct a state transition matrix, whose values are $W_{ij} = P(s_t = j | s_{t-1} = i)$ and represent the probability of transitioning from state i to state j. With Q quantiles, W is a $Q \times Q$ matrix.

The *Markov Transition Field* (MTF) follows a similar approach. It is a matrix M of size $n \times n$, where $M_{kl} = W_{q_k q_l}$, with W following the normalization $\sum_{q_l} W_{q_k q_l} = 1$. In practical terms, $W_{q_k q_l}$ is the count or frequency at which a point in quantile q_k transitions to q_l, where q_k is the quantile for the values f_k and q_l is the quantile for the values f_l[16]. Thus, M_{kl} represents the probability of a transition from the quantile of f_k values to the adjacent quantile of f_l values. In summary, the MTF shows how related two points in the time series are. An illustration of this is given in Figure 58, where the example light curve of this section was transformed to $Q = 8$ using the Python package `pyts` (Faouzi; Janati, 2020).

Figure 58 – MTF example for a light curve



Source: Self elaboration.

More examples of images generated by the transformations introduced above are shown in Figures 74, 75, and 76. Through them, we can, naturally, see that the MTFs do not

[16]Contrary to what the authors (and several other sources) do, I made the difference between the indices of M and W explicit because those of M refer to time and those of W refer to states.

depend on the normalization between $[-1,\ 1]$ or $[0,\ 1]$, as in the Gramian fields. The most striking characteristic of the light curves from TCEs known to be from planets is the sharp cross pattern generated by the decrease in light flux during planetary transit in all channels. This shows promise in the context of image classification, and the implications of it will be discussed in the next chapter.

4.3 Recurrence Plot

To exploit the possibilities of the present technique, we used one more type of time series imaging, the *Recurrence Plot* (RP). First introduced by Eckmann *et al.* (1987), RPs were used on diagnosis of assumptions of whether or not a time series was obtained from an autonomous dynamical system, *i.e.*, if the evolution equations do not contain the time explicitly. This assumptions are necessary to compute dynamical parameters from time series such as information dimension, entropy, Liapunov exponents, dimension spectrum, and so on, by an alternative approach. This is done, in general, by projecting a time series into a multidimensional space by embedding procedures and identifying time correlations, or recurrences, that are not directly apparent with a conventional one-dimensional representation.

The process to generate a recurrence plot, proposed by the authors, is given as follows. Consider a time series $\boldsymbol{X} = (x_1,\ \ldots,\ x_n)$. We choose an embedding dimension d and a time delay τ, so that we can extract a d-dimensional orbit, or trajectory, $\boldsymbol{u}$, where $\boldsymbol{u}_i = \left(u_i,\ u_{i+\tau},\ \ldots,\ u_{i+(d-1)\tau}\right)$, $\forall i \in \{1,\ \ldots,\ n-(d-1)\tau\}$. We then generate a $n \times n$ matrix $\boldsymbol{R}$ and insert a point in $\boldsymbol{R}_{i,j}$ if the point described by $\boldsymbol{u}_j$ is sufficiently close to $\boldsymbol{u}_i$. This closeness is given by a chosen $r(i)$ so that the d-dimensional ball of radius $r(i)$ centered at $\boldsymbol{u}_i$ in $\mathbb{R}^d$ contains a reasonable (but also arbitrary) number of other points $\boldsymbol{u}_j$ of the orbit. The filled matrix $\mathbf{R}$ is the recurrence plot.

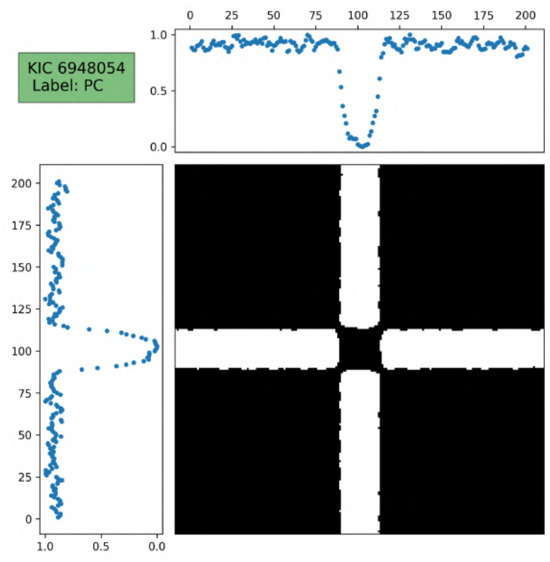
It is very common in literature to consider $\tau = 1$. Equation 1 of Thiel *et al.* (2004) gives the mathematical description of $\boldsymbol{R}_{i,j}$ for this particular case. Considering the general case where τ can be different of 1 and maintaining the consistency of the notation just presented, we have (ZBILUT *et al.*, 2002)

$$\boldsymbol{R}_{i,j} = \Theta\left(\varepsilon - ||\boldsymbol{u}_i - \boldsymbol{u}_j||\right), \qquad \forall i, j \in \{1,\ \ldots,\ n-(d-1)\tau\}, \tag{4.6}$$

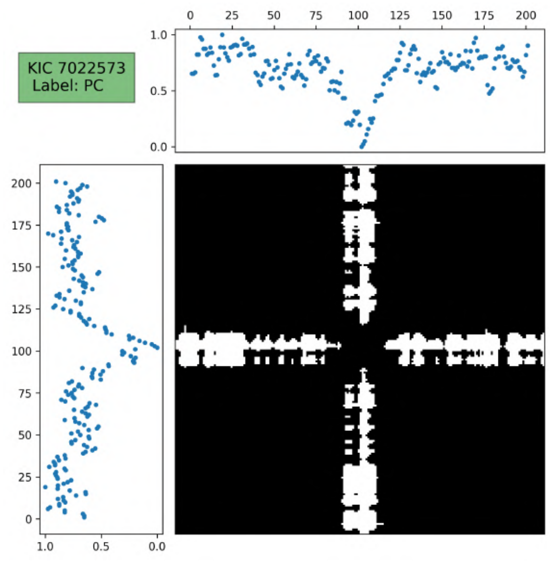
where $\Theta(\bullet)$ is the Heaviside function (*i.e.* $\Theta(\xi) = 0$ if $\xi < 0$, and $\Theta(\xi) = 1$ otherwise). In practical terms, there is a trade-off between the transformation complexity and the amount of information we can retrieve from RPs. Figure 59 shows some examples of this procedure applied on pre-processed light curves with the methods described in section 2.3. They were created by choosing $d = 2$ and $\varepsilon = 0.7$, which came to be a reasonable choice of parameters after a careful visual inspection on a representative sample of the full dataset.

Figure 59 – Examples of representative cases of recurrence plots

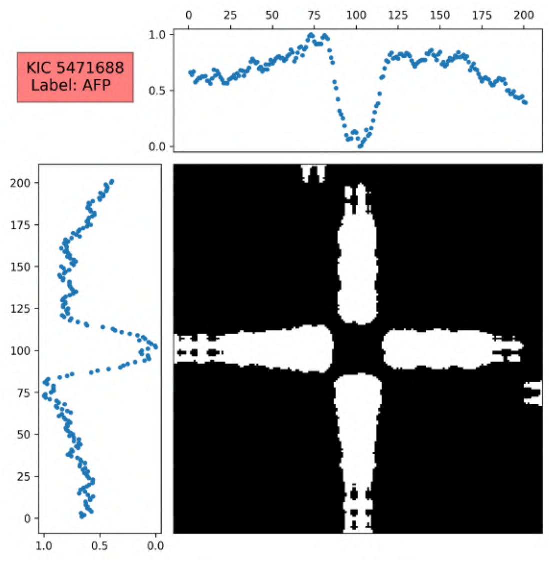
(a) Recurrence plot for KIC 6948054



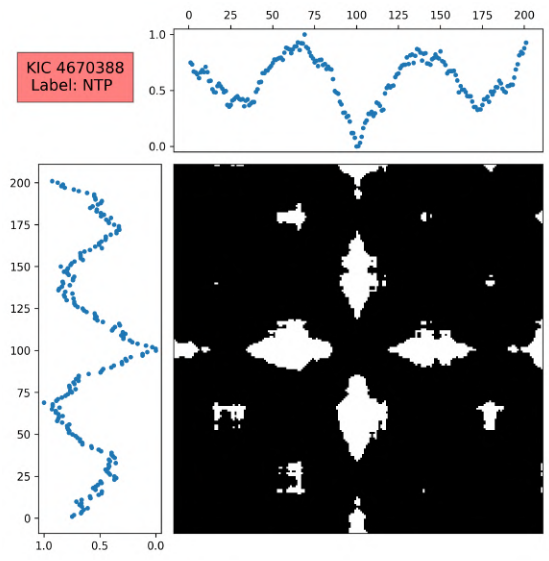
(b) Recurrence plot for KIC 7022573



(c) Recurrence plot for KIC 5471688



(d) Recurrence plot for KIC 4670388



Source: Self elaboration.

Once again, we see in Figure 59(a) the cross shape we saw with the other imaging methods for PCs. PCs with more noise in the light curve also maintain this trend, as we see in Figure 59(b). Although AFPs may have a transit-like drop in the light curve, it is very clear by Figure 59(c) that the resulting recurrence plot generates a somewhat different pattern in this case.

Finally, Figure 59(d) shows that a NTPs generates the most distinct RPs from that we see in the PCs cases.

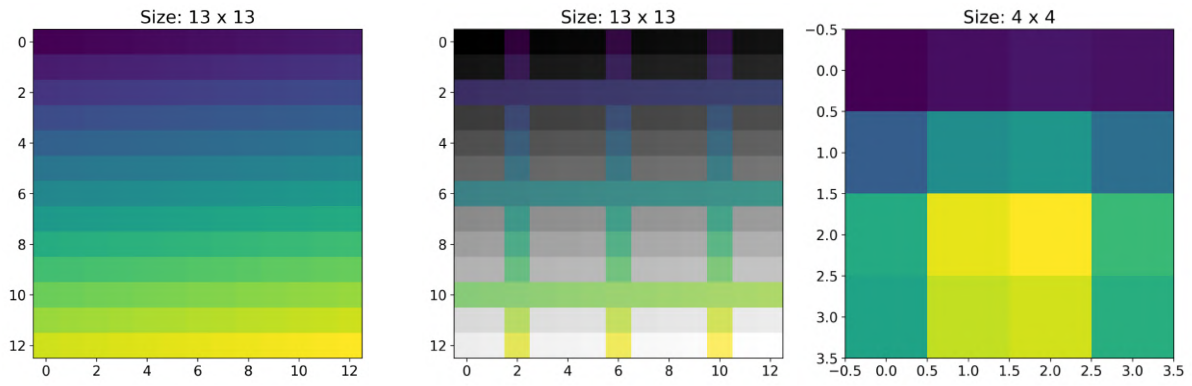
One of the observed advantages of RPs such the ones in Figure 59 in terms of machine learning in general is that they are *binary* recurrence plots, containing only 0's and 1's in their array. In comparison, a RP might not be binary if it is unthresholded (*i.e.* $\varepsilon = 0$) (THIEL *et al.*, 2004). Unthresholded RPs might contain more information about time series in general, but they are more difficult to quantify than binary RPs. With binary data, the patterns a ML or DL model have to learn are simpler because now they can focus more on the topology of the data over their magnitude of values. This simplicity may be helpful when the data is not very complex.

4.4 Reducing Data Size

With all these transformations, the size of the data set was increased dramatically, going from roughly 1 GB to over 1 TB. This was already expected, since we are essentially squaring the size of each light curve for *each* type of filter. For the global view light curves alone, each filter produced has a resolution of 2001×2001 pixels, which is even grater then a 2K resolution (2048×1080 pixels). This represents a very serious computational problem, specially for machine learning purposes. Wang and Oates (2015) overcame this issue by performing a Piecewise Aggregation Approximation (PAA) (Keogh; Pazzani, 2000) on their time series to smooth them while also preserving trends. As explained on section 2.3, it is a very similar process to the data binning already performed on the light curves to the limit balance between lightweight files and information loss.

Since reducing the light curves is no longer productive at this point, we reduced the size (or resolution) of each generated image. To do this, we introduced an additional step on the image generation pipeline, called here as *pre-pooling*, that was inspired on the process to create a pooling layer on CNNs, described on subsection 3.2.2. As in the pooling process, we divide the image data on windows and make a new image by taking the outputs of a desired aggregation function. Figure 60 shows a small example image to illustrate this. First, on Figure 60(a), we have a arbitrary image that we want to shrink. Then, on the gray areas of Figure 60(b), we select specific regions on the image whit a window of established height and width (3 for both, in this case). These windows are separated by a chosen value gap between them (1, in this case). If we are to maintain the CNN formalism, this gap refers to the stride and should not be counted merely as a gap between windows, but as the step the windows take from one another. The gap of 1 becomes a stride of 4 in this example. Finally, Figure 60(c) shows the resultant image formed by taking the mean of the gray areas in Figure 60(b).

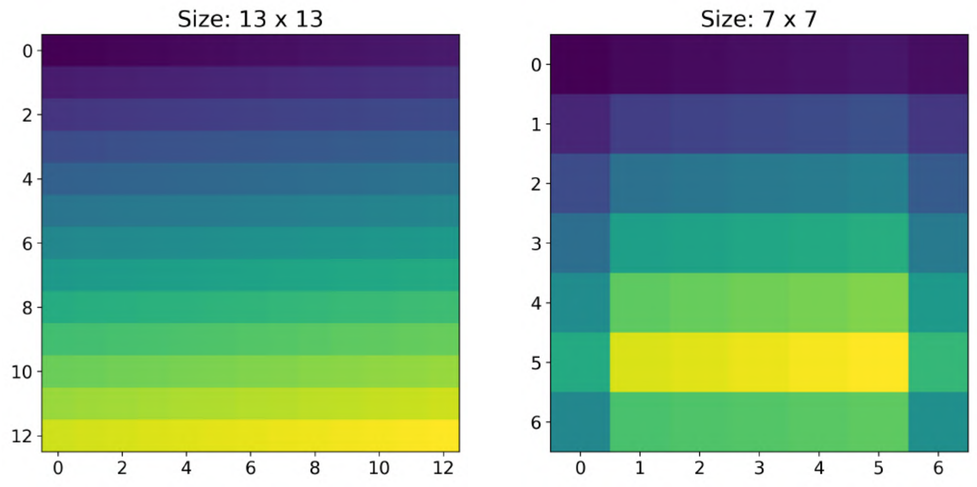
Figure 60 – Example of a pre-pooling process with stride 4
(a) Original image (b) Selected windows (c) Result image



Source: Self elaboration.

Note that we reduced the size of a image from 13×13 to 4×4. Using the same 3×3 window size and decreasing the gap to -1 (stride 2), we overlap the windows, similar as in Figure 45, and generate yet another image with a little higher resolution. Figure 61 shows that the new image size is now 7×7.

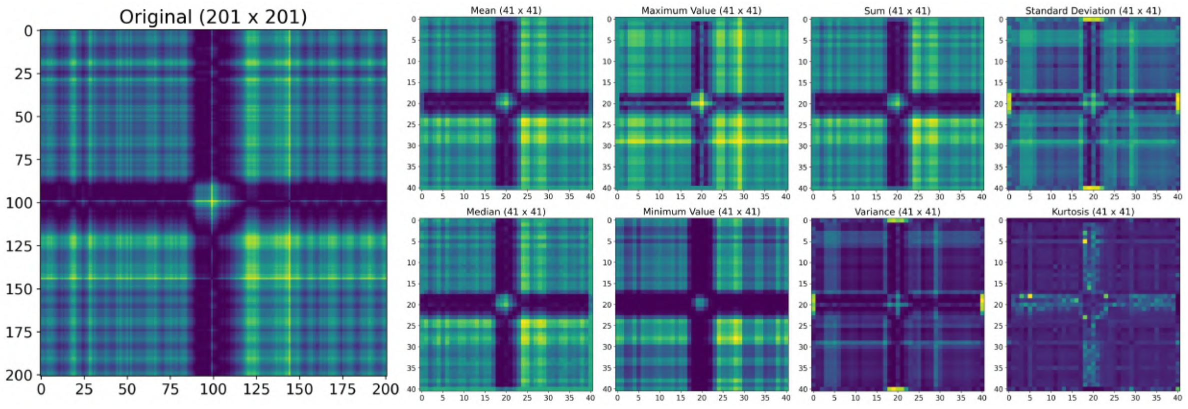
Figure 61 – Example of a pre-pooling process with stride 2
(a) Original image (b) Result image



Source: Self elaboration.

The aggregation function used in the previous example was the mean of the data contained on the windows, however we are free to chose any other appropriate function. The pre-pooling process was made using a 3×3 window and stride 2 on the images generated from the local light curves and using a 6×6 window and stride 2 on the global light curves. The output images are 41×41 and 401, respectively. Different aggregation functions were tested, as shown in Figure 62.

Figure 62 – Pre-pooling results for different aggregation functions



Source: Self elaboration.

Visual inspection on some dozens of samples of both classes of TCE was made to evaluate which of the tested aggregation function produced a better result, based on the the preservation of the trends on the images. The mean, maximum value and sum presented the best outputs and, among these three, the mean was arbitrarily chosen as a aggregation function for the whole pipeline. At the end of the process, the light curves and images were serialized, compressed and then saved as TFRecords files[17]. The new dataset, that had approximately 1 TB, ended up with ~ 80 GB.

[17]More detais in: https://www.tensorflow.org/tutorials/load_data/tfrecord

5 RESULTS AND DISCUSSIONS

The main types of models introduced in Chapter 3 where based on the concepts of Logistic Regression (LogistReg), Dense Neural Network (DenseNet), 1D Convolutional Neural Network (1D CNN) and 2D Convolutional Neural Network (2D CNN). In this chapter, we give a brief summary of the best models, compare their overall performance and then show an extra test that will gide the next steps of future work.

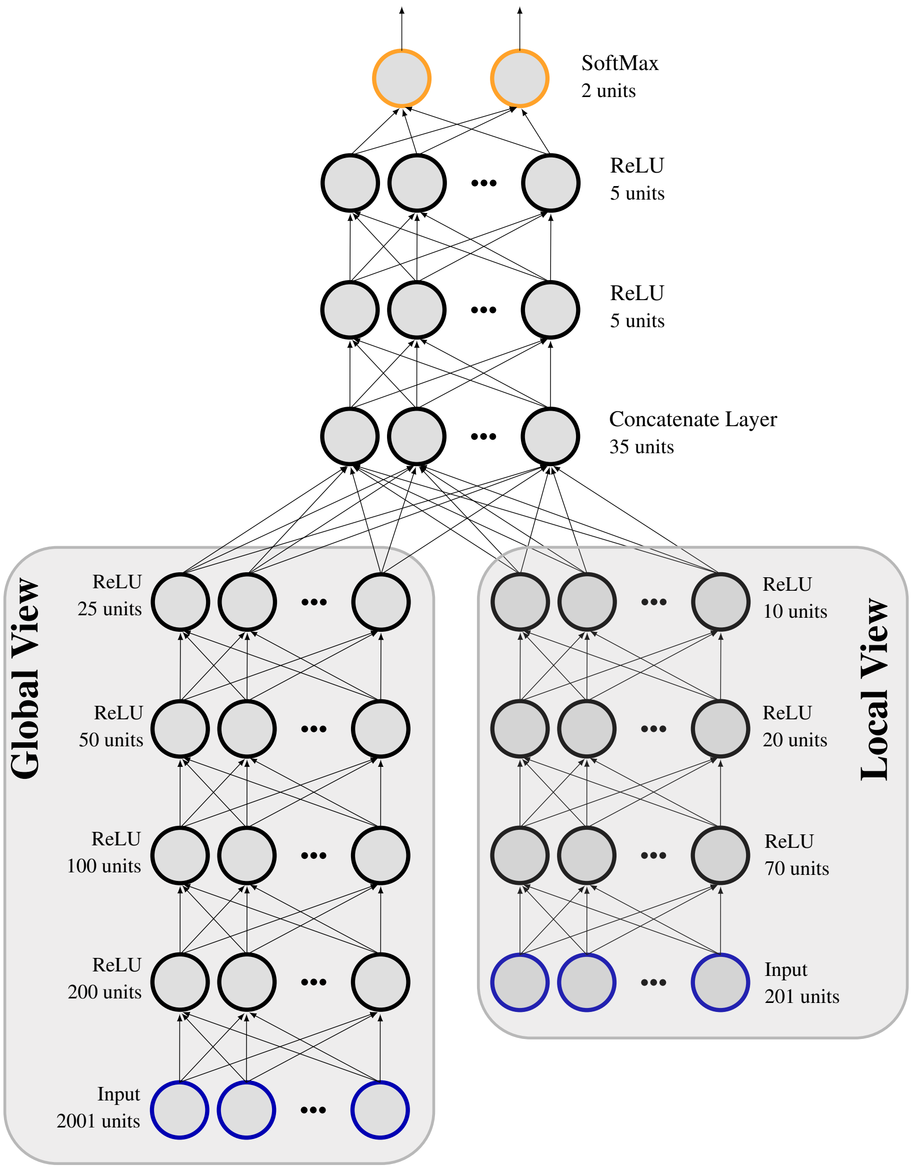
5.1 An Overview of the Types of Models Tested

Based on what was discussed in subsections 3.1.1 and 3.1.2, a LogistReg model can be given by a DenseNet, such as the one in Figure 42, but with no hidden layers. Since we have 2001 bins representing a light curve in its global view and 201 bins in its local view, our LogistReg model counted with 2202 inputs. In this context, both light curve views data were concatenated into a single array, so that the spatial disposition of the data were kept. This is the most simple tested model and, because of the nature of the logistic regression itself, makes the assumption that planets and non-planets can be separated by a linear surface in the input space. Although most of the models were implemented using scikit-learn (Pedregosa; others, 2011), the final version of this type of model was implemented, along with all the other models ahead, with TensorFlow (ABADI *et al.*, 2015).

A DenseNet, by the other hand, makes fewest assumptions about the input data and can generalize better more complex patterns. With a DenseNet, we have way more parameters to handle, so the search for an appropriate architecture and hyperparameter setting is crucial for a good performance. Many different models were created, trained, tested and compared among them. The one that showed the best performance has its architecture depicted in Figure 63. It has two branches: one to process the data from global view alone and other to process the local view alone. We used ReLU activation function and a 25% dropout on all the layers. We also used stochastic gradient descent as the optimizer function with a learning set to 0.01. These values, and the following ones, were found using KerasTuner (O'MALLEY *et al.*, 2019).

The third type of model tested was a 1D CNN. The level of complexity it can learn to distinguish is considerably higher than DenseNets in general, specially if we have spatially structured data, which is the case for the signals in our light curves. Since the data is originally one-dimensional, this kind of convolutional network is applicable. In the search for maximizing

Figure 63 – Architecture of the best dense neural network used in this work

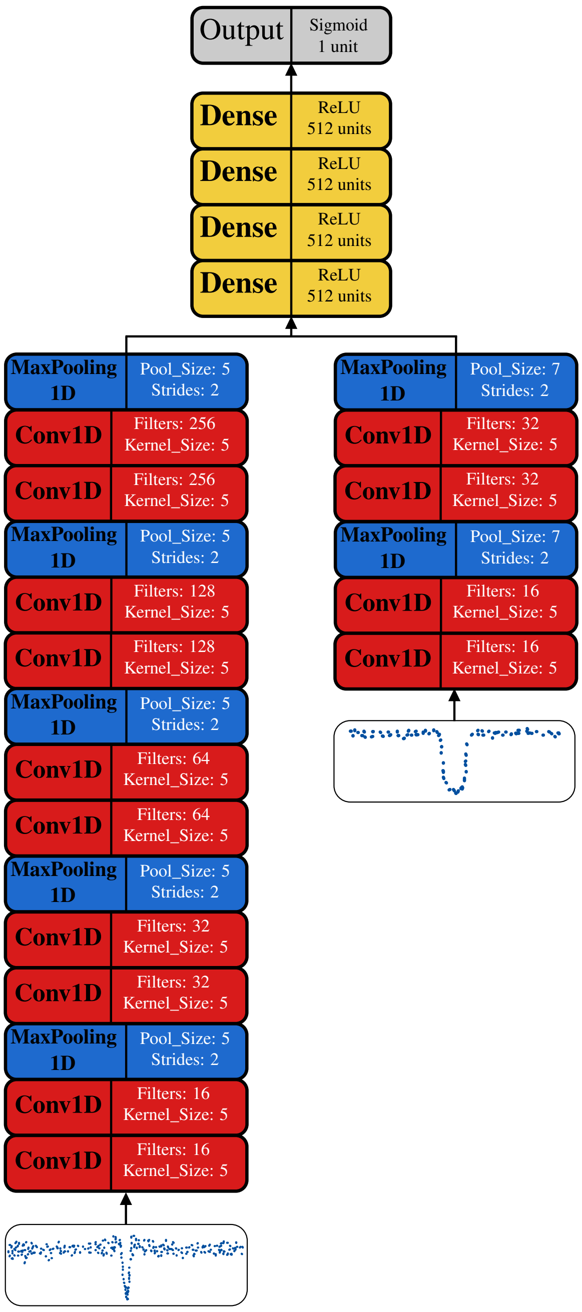


Source: Self Elaboration.

ROC-AUC in the validation dataset, the architecture found was the one depicted in Figure 64 and its best hyperparameters were α (learning rate) = 10^{-5}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$, which are inherent hyperparameters for the Adam optimization algorithm (Kingma; Ba, 2017).

Once again, the network is firstly divided in two segments, one for the global view and one for the local view. Both segments are independent 1D convolutional networks whose outputs are passed to a dense neural network with 4 layers using ReLU activation function without dropout and a 2 unit Sigmoid output layer. Interestingly, this is the exact same architecture and hyperparameters setting found by (Shallue; Vanderburg, 2018). This is not surprising, though, since the authors made an intensive architecture and hyperparameter search for the best model that received a 2001 and 201 global and local views sized light curves, respectively.

Figure 64 – Architecture of the best 1D convolutional network used in this work



Source: Self Elaboration.

The fourth and last type of model tested was a 2D CNN. With what we have seen in subsection 3.2.2, they overcome all the others models here presented in terms of complexity. They can be applied in a wide variety of tasks, being computer vision a very recurrent example. All this power and complexity, however, might come with just the opposite effect. Over-complex models might perform worse than simpler ones mainly because of overfitting. Indeed, many 2D models, among the thousands of tested ones, manifested this problem. The standard approach when creating any kind of model, 2D CNNs in particular, is to start from the simplest possible architecture and then start making little changes until finding one configuration that presents a good performance. This process was done several times until we find a promising model.

After that, we tried to apply transfer learning, as explained in section 3.4, to use a state-of-the-art pre-trained model with an a priori high performance. Many models are available online, in the Keras API[18], including those displayed in section 3.3. Some of these models are very large, so we selected one with a good balance between size and performance. The performance data available in the Keras API documentation refers to top-1 and top-5[19] accuracy on the ImageNet validation dataset. With this, we divided a model's accuracy by its size and created a simple metric we called *Accuracy-Memory Ratio* (AMR) for cost-benefit analysis. Naturally, AMR can be given in terms of top-1 and top-5 accuracy. A summary of this is displayed on Table 9. EfficientNetV2-B0 was chosen for being the model with top-1 accuracy grater than 75% with the best AMR.

Table 9 – Accuracy-Memory Ratio for some models available in Keras API

Model	Size (MB)	Top-1 Accuracy	Top-1 AMR (%/MB)
MobileNetV2	14.00	71.3%	5.10
MobileNet	16.00	70.4%	4.40
NASNetMobile	23.00	74.4%	3.23
EfficientNetV2B0	29.00	78.7%	2.71
EfficientNetB0	29.00	77.1%	2.66
EfficientNetB1	31.00	79.1%	2.55
EfficientNetV2B1	34.00	79.8%	2.35
DenseNet121	33.00	75.0%	2.27
EfficientNetB2	36.00	80.1%	2.23
EfficientNetV2B2	42.00	80.5%	1.92
EfficientNetB3	48.00	81.6%	1.70
EfficientNetV2B3	59.00	82.0%	1.39
DenseNet169	57.00	76.2%	1.34
EfficientNetB4	75.00	82.9%	1.11

Note: Only models with AMR ≥ 1 are shown.
Source: Self elaboration. Data retrieved from Keras API: https://keras.io/api/applications/

With the selected architecture, a transfer learning approach was made by using the weights for ImageNet dataset. To do this, the whole local views light curves dataset had to be re-generated with the procedures described in Chapters 2 and 4, since the images from ImageNet in EfficientNetV2-B0 architecture have a 224×224 size. Then, all the new local view light curves have 224 points instead of the previous 201. Therefore, all images generated from them are

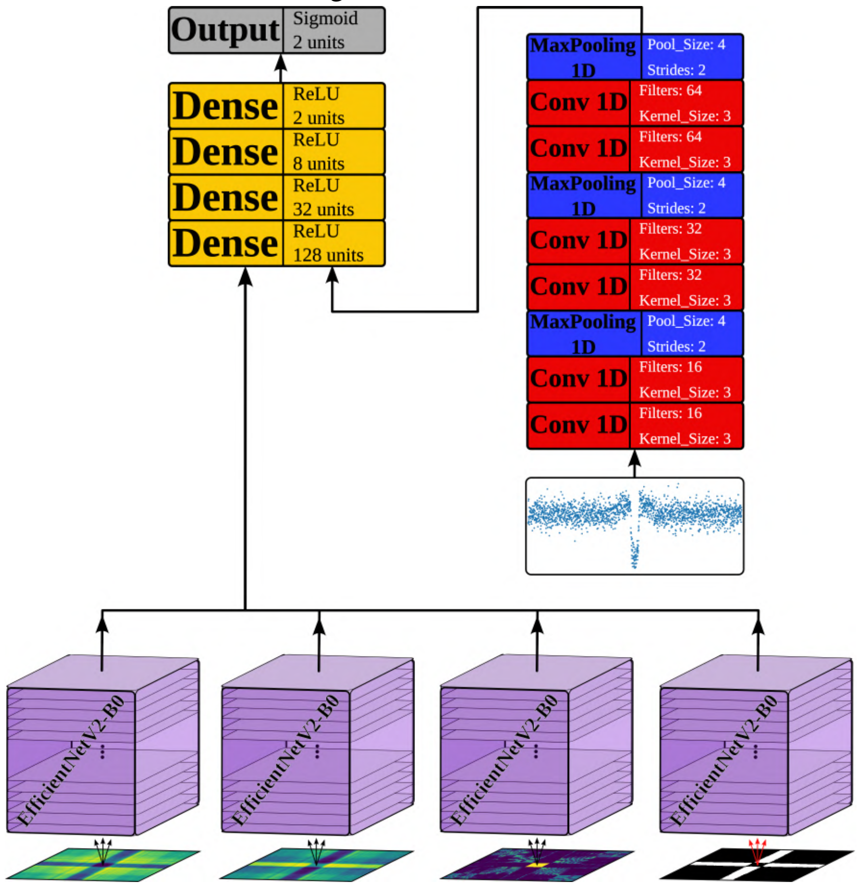
[18]See https://keras.io/api/applications/.

[19]Top-1 accuracy is the usual accuracy, explained in section 3.5, but taking into consideration that ImageNet is a multi-class dataset with 1000 different classes. The top-5 accuracy is a more lenient measure that gives the proportion of times a classifier's top 5 most likely predictions match the true label.

224×224 sized, except for the recurrence plots, which, because of the nature of their generation, ended up with a size of 223×223. The reason to deal only with local view light curves is because images generated for global view are large and demands impractical computational resources for our hardware. As a matter of fact, once this problem was observed, all 2D models tested after did not used the images generated from global light curves.

All the tested models with transfer learning performed poorly. This outcome was expected, even with a state-of-the-art model, since the ImageNet data is very unrelated with the ones generated by the imaging methods discussed in Chapter 4. Because of that, no harsh fine tuning was made for ImageNet transfer learning models. However, the EfficientNetV2-B0 architecture was not left aside, since we could use it to train the network from scratch with our data, without any pre-loaded weights. Among several dozens of trials, the model described by Figure 65 outperformed all the other 2D models when analyzing ROC-AUC.

Figure 65 – Architecture of the best large 2D convolutional network used in this work

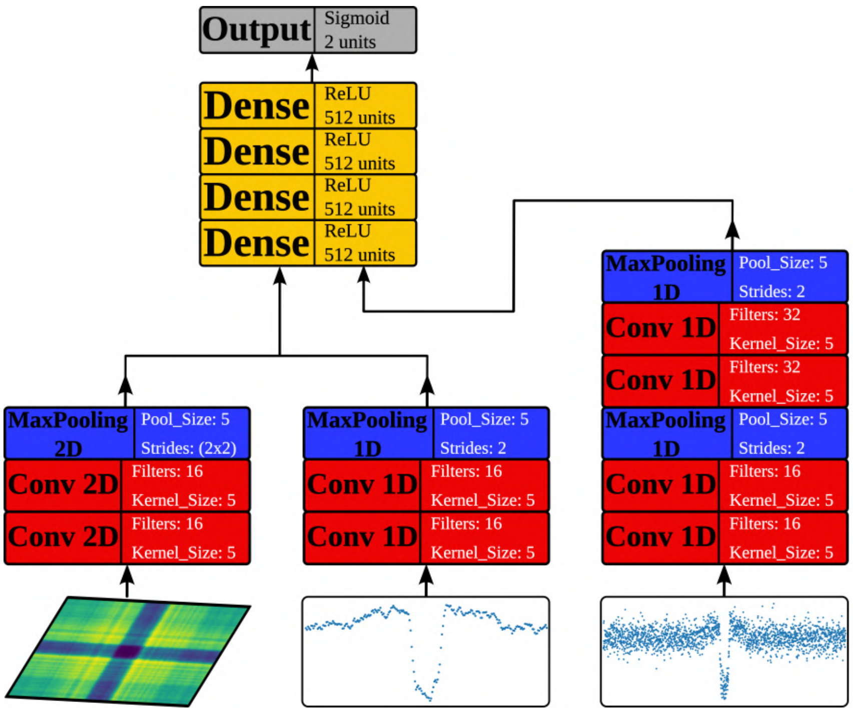


Source: Self elaboration.

This model has five segments. One of them is a 1D convolutional column that recieves the data from the global view light curves, very similar to the ones in Figure 64. The other four are independent EfficientNetV2-B0. Each 2D convolutional segment is dedicated to analyse the images generated from the local views. Notice the the local view light curve it self is not inputted into the network. All the information of the segments converges into a DenseNet with 20% dropout to finally arrive at the sigmoid output layer. With a batch size of 10 and 50 epochs, the training of this network lasted approximately 9 hours in a T4 cloud GPU.

Roughly speaking, our 2D models were designed and grouped by three categories: small (few convolutional layers), medium (using one famous architecture) and large (using two or more famous architectures). Most of the small models had good performance scores, while most of the large were poor due to overfitting. Smaller models were less harder and faster to fine tune, which explains part of their performance. As we see, the best model of all tested 2D CNN was one considered large. However, an interesting fact is that a small model is in a technical tie with this one. Figure 66 shows its architecture.

Figure 66 – Architecture of the best small 2D convolutional network used in this work



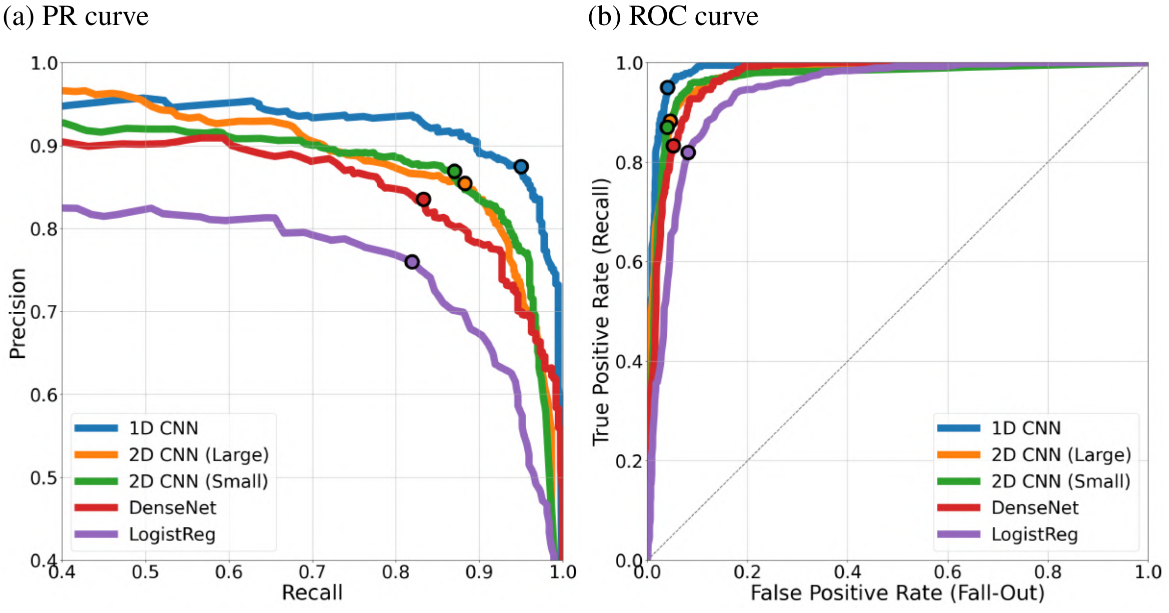
Source: Self elaboration.

Because of their sizes, we did not performed a harsh fine tune on ts their hyperparameters yet, so the performances displayed in the next section come mainly from raw parameter initialization. Still, we trained and evaluate their performances on both 201×2001 and 224×2001 local-global sized datasets. The best performance, in both small and large model, came from the later size configuration. We also noticed that all three types of tested 2D models so far (small, medium and large) had a moderate performance when combining images to form a 3-chanel or from a 4-chanel image, similar to what is exposed in Figure 47.

5.2 Best Models Comparison

Using the performance metrics introduced in section 3.5, we could evaluate the results of the best models in each model category, as explained in the previous section. Those who better performed on the test set were selected and again compared among themselves. Figure 67 and Table 10 shows the performance curves and metrics, respectively, of the best models.

Figure 67 – Performance curves for the best models



Note: The dots indicate where the models achieved the highest accuracy.
Source: Self elaboration.

Table 10 – Performance metrics for the best models

Model	PR-AUC	ROC-AUC	Accuracy	Precision	Recall
1D CNN	93.55%	98.60%	95.74%	87.47%	95.00%
2D CNN (Large)	91.70%	97.22%	93.78%	85.45%	88.25%
2D CNN (Small)	88.81%	97.04%	93.93%	86.90%	87.02%
DenseNet	87.98%	96.93%	92.07%	83.51%	83.29%
LogistReg	79.38%	93.96%	89.50%	76.00%	81.94%

Note: The table shows the highest possible accuracy value. The precision and recall relates to the threshold choice that maximizes accuracy.
Source: Self elaboration.

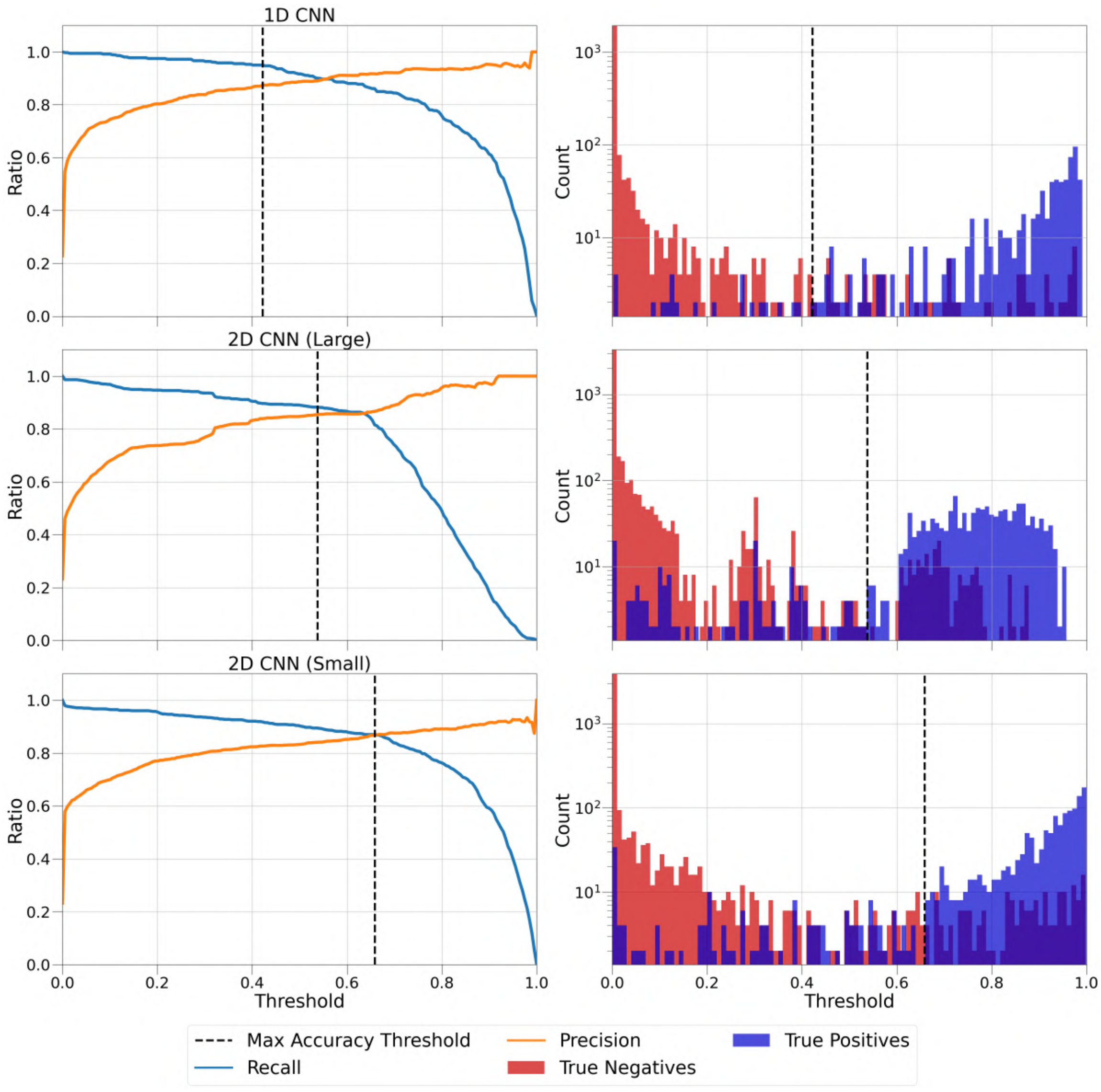
From these results, we can clearly see a higher overall performance of the 1D CNN over the other models. Once again, this was not a surprise since it has the same architecture and hyperparameters from the work of Shallue and Vanderburg (2018), who made a full hyperparameter tuning in regards of all aspects of the model. Even the local and global representations of the light curves themselves were fine tuned, having in mind equation 2.3. Still, the two best 2D models had a good performance. As discussed in section 3.5, ROC-AUC metric says that the large 2D CNN ranks plausible planet signals higher than false-positive signal 97.22% of the time in our test dataset. Both 2D models scored almost 94% accuracy, but the metric of most interest for the purposes of application of this work is perhaps recall. Since it gives the proportion of positives that were correctly identified, a model that has a good precision, but an even better recall is important. This being said, 1D CNN has a far advantage.

The left column of Figure 68 shows both precision and recall in terms of the threshold used for evaluation of the three overall best models. The right column shows the distribution of true positives and negatives also in terms of the threshold. With the plots on the column on the right, we can make an analysis similar to the one made with Figure 54. From this figure and Table 10, we can see that a lower threshold implies a higher recall. Also, we see that the *Maximum Accuracy Threshold* (MAT) is inversely proportional to the number of false positives.

An interesting observation regards the recall of the large 2D CNN. It presents two approximately linear regimes with negative slope. When this happens, the critical point of regime transition coincides with the precision curve, but this, in general, is not necessarily the case. At this point, the model abruptly starts becoming overly conservative in its predictions, but we can see why when looking at its distribution of predictions in the histogram on the right. At threshold ≈ 0.68, a great concentration of true positives is observed, which suggests that the model was able to find a strong pattern in the data. This high concentration of true positives is also observed in the smaller 2D CNN. This observation also suggests that the success of the 1D CNN is related to its capacity of rejecting false positives. By doing this, a smaller value to the threshold is viable for a higher performance, particularly recall.

Another observation worth taking is that all of those three models used a sigmoid activation function for their output layers. As equation 3.6 states, the output of a sigmoid function can be interpreted as the probability that the input sample belongs to the positive class. Since the thresholds are given in terms of these output values, the 1D CNN and, especially, the larger 2D CNN did not had any full (100%) confidence on any sample in the test dataset. The smaller 2D CNN, by the other hand, had the greatest counts on confidence when classifying true positives, at the cost of being more erroneous in the same region of confidence in its predictions.

Figure 68 – Threshold analysis for the best models



Note: The 2D CNNs were trained and tested with more data, because we had more freedom to augment the images datasets. The distribution of classes remained the same the for all three models.
Source: Self elaboration.

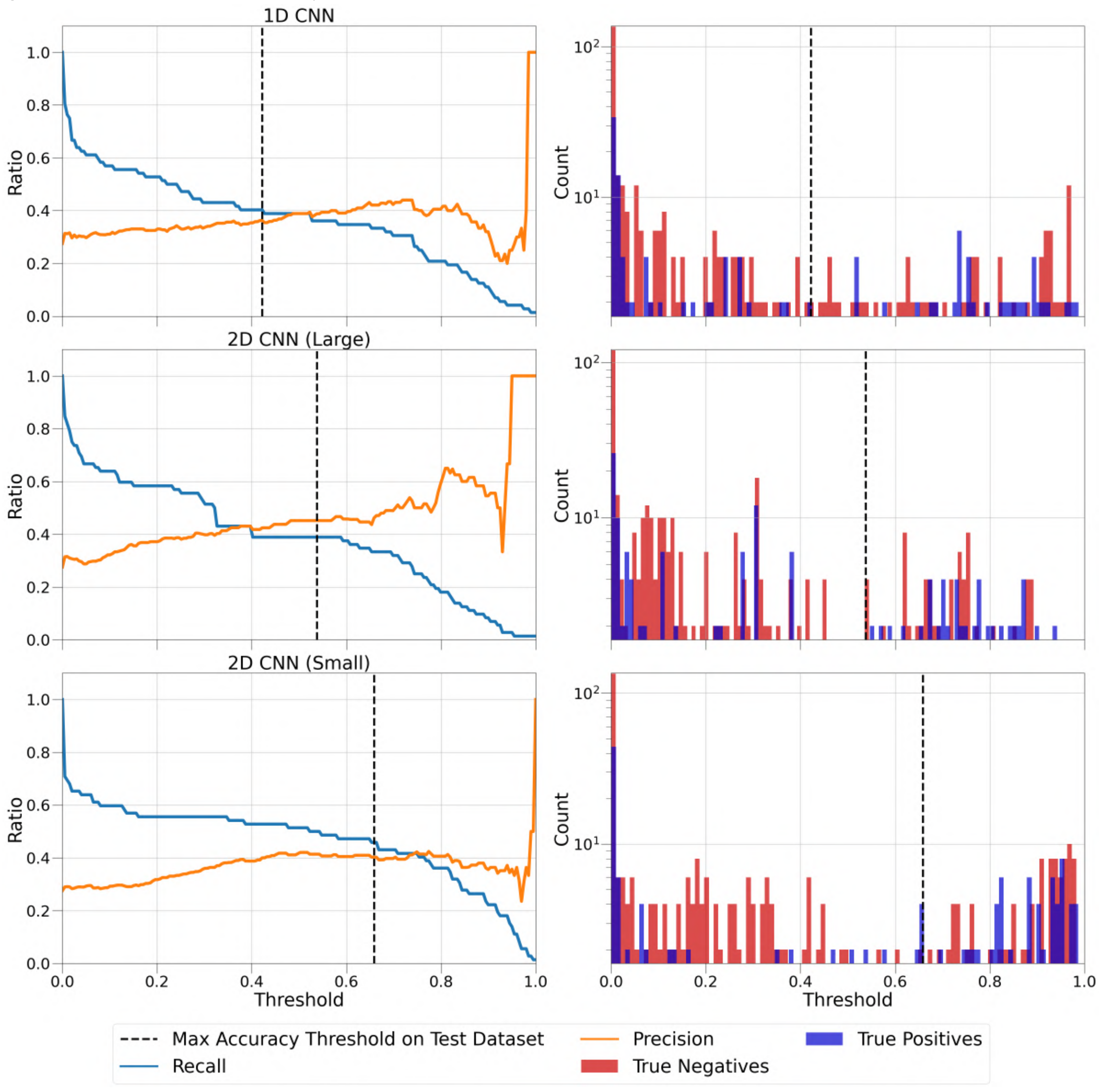
5.3 Test on Data from Other Space Missions

In order to measure the applicability of the models on other space missions data, we selected a sample of 522 TCEs from K2 and TESS missions together[20]. More specifically, 144 TCEs are confirmed planets from the K2 mission, 10 are AFP also from K2 mission and the rest 368 are AFP from TESS mission. Notice that the proportion of the positive class over the entire

[20]The K2 and TESS TCE data references are available in the *K2 Planets and Candidates* and *TESS Project Candidate* tables at https://exoplanetarchive.ipac.caltech.edu/.

new dataset is approximately the same as in the training dataset (≈ 27,59%). All the light curves treatment was made exactly as described in Chapters 2 and 4. We, then, inputed the formatted data into our models so they could make predictions. Figure 69 shows the performances of the models.

Figure 69 – Threshold analysis for the best models in the extra test dataset



Note: The thresholds indicated on the plots are those correspondent to the model's maximum accuracy (MAT) in the original test dataset.
Source: Self elaboration.

As we can see, the models' performances are substantially worse than in the test made before. Tables 11, 12 and 13 shows, in numbers, these results. Considering equations 3.12, 3.13 and 3.14 and table 8, by setting the models' thresholds as the same as the MATs from the test before, the large 2D CNN had an accuracy of 70.50%, the highest of all three models. However, the small 2D CNN achieved a recall of 44.44%, being the model that most retrieved true planets. All the metrics extracted from the confusion matrices of this test are presented in table 14.

Table 11 – Confusion matrix for the 1D CNN model on the extra test dataset

		Predicted	
	1D CNN	Negative	Positive
Original	Negative	276	102
	Positive	88	56

Source: Self elaboration.

Table 12: Confusion matrix for the small 2D CNN model on the extra test dataset

		Predicted	
	2D Small	Negative	Positive
Original	Negative	282	96
	Positive	80	64

Source: Self elaboration.

Table 13 – Confusion matrix for the large 2D CNN model on the extra test dataset

		Predicted	
	2D Large	Negative	Positive
Original	Negative	312	66
	Positive	88	56

Source: Self elaboration.

Table 14 – Performance metrics for the complementary test

Model	**Accuracy (%)**	**Precision (%)**	**Recall (%)**
1D CNN	63.60	35.44	38.89
2D CNN (small)	66.28	40.00	44.44
2D CNN (large)	70.50	45.90	38.89

Source: Self elaboration.

Based on the results above, we can infer that the models work well on Kepler Mission data, but not as well on data from other space missions, K2 and TESS in this case. Although the physical process of transit is the same, systematic trends and noise inherent in the missions might influence some decisions made by the networks. Notice that the 1D CNN model, which was the best on the first test, performed worst in all aspects. This suggests that the model might have given a lot of importance to the training trends at the cost low generality on other datasets. In practical terms, the most accurate network (the larger 2D CNN) was able to filter 82.54% of the true negatives at the cost of $61, 11\%$ of the planets being lost. With this, an astronomer would find a true planet $45, 90\%$ of the time, if following the predictions of the model. Although not the best scenario, this represents a welcoming improvement in an initial finding rate, considering that the whole dataset had only $\approx 27, 59\%$ of planets.

An important observation is that all the representations of the TCEs from this test were inspect by eye, one by one. We could see, in fact, that all samples were coherent with the theory given in subsection 2.1.3 for the positive class. Also, we saw that the AFP class data was, in some cases, very similar to transit-like signals, which could present a difficult challenge for the models.

A natural step for improving the models performance in this case would be to train them with data not only from the Kepler mission data, but also with K2 and TESS data as well. This is expected to make the models more general while working with future telescopes data, such PLATO. Another desirable aspect the models must improve is the sensitivity on lower SNR and MES data. NASA Ames Research Center made a whole dataset with Kepler mission data with injected signals to mimic planetary transits, EBs, and other kinds of signal with many values of MES, including those that are lower than the Kepler's pipeline threshold (Christiansen, 2017). We did not made any tests on these data yet, but Shallue and Vanderburg (2018) already show that their 1D CNN (using our nomenclature to represent their model tested here) is not sensitive enough to weak secondary eclipses to classify many of these simulated signals as false positives. It will be interesting to see if the 2D CNN presented in here will have a better performance on these data in future works.

6 CONCLUSIONS AND PERSPECTIVES

In this work, we revised the main methods of exoplanet detection and gave a glance at some technicalities of the astrophysical data processing and evaluation. In summary, we considered space telescopes that measure photometric data in a given region of the sky. With this data, astrophysicists can search for periodic signals and study their properties to evaluate if it is given by a planetary transit or not. The evaluation of this signal candidate is made with a series of rigorous and time-consuming tests. With the ever-growing number of data and space missions launched, manual evaluation is becoming unpractical.

We also revised the basics concepts behind machine learning and, more specifically, deep learning algorithms. In subsection 2.2.3, we saw some of the most well known machine models to automatically classify the supra mentioned astrophysical data into planet or not. Although having their unique specific characteristics, all of them share the same overall goal, that is to reduce the manual load of analysis once required from the astrophysical community while making reliable evaluations.

With all this, we were able to pursue an approach given by Shallue and Vanderburg (2018) on how to build a deep learning algorithm to automatically classify signals candidates. Although not being the best model nowadays, it is simple enough so we could review and briefly explain the concepts of interest from both areas of exoplanetology and artificial intelligence and how they can be related. Also, it served as a good model for comparison with the data treatment proposed in this work: the time series imaging.

The time series imaging technique was proposed in the deep learning and computer vision context by Wang and Oates (2015). As we saw with Eckmann *et al.* (1987), in section 4.3, the concept it self of transforming a time series into an image is not new. Wang and Oates (2015) showed that time series imaging was a valid and relevant method to produce state-of-the-art models at the time of their proposal. In addition to carrying out a didactic review on the subjects discussed above, one other objective of this work was to perform a conceptual trial to see if the time series imaging method could provide an improvement on a deep learning algorithm that works with exoplanet detection.

To pursue this objective, we reproduced the algorithm from Shallue and Vanderburg (2018) (the 1D CNN model) and compared with the best ones we built (the small and large 2D CNN models). The models used part of the data from Kepler mission available on the *Autovetter Planet Candidate Catalog for Q1-Q17 DR24* (Catanzarite, 2015), were 80% of it was dedicated for training, 10% for validation and 10% for testing. Given the nature of the method proposed, an immediate observation is that the 2D models are way heavier than the one we took as reference. This had a direct impact on the computational cost for the models' training and, specially, hyperparameters fine tuning. Even so, our results so far, presented in Figure 67 and Table 10, shows that we can build a decent classifier based on the imaged data from Kepler mission. By what was discussed in section 3.5, the ROC-AUC metric says that the large 2D CNN ranks plausible planet signals higher than false-positive signal 97.22% of the time in our test dataset. Both 2D models scored almost 94% accuracy. Due to computational limitations, we did not had time to search and fine-tune an even better model, so we did not achieved a performance on the test set as good as the 1D CNN model proposed by Shallue and Vanderburg (2018) (ROC-AUC = 98.60% and accuracy = 95.74%).

However, the aspect of most concern is, perhaps, the capacity of the models to generalize their performance on other datasets, specially the ones generated with data from other space missions, such as K2, TESS and the coming PLATO. A complementary test was made using data from K2 and TESS missions. Surprisingly, Figure 69 and Table 14 showed that both our models performed better and could generalise more on new data. Even though the metrics themselves are no match for the state-of-the-art ExoMiner's metrics (Vallenari *et al.*, 2022), this result suggests that the time series imaging method might, indeed, be helpful on the task of exoplanet search.

As discussed in sections 5.2 and 5.3, one of the greatest problems of all the models, including the 1D CNN models from Shallue and Vanderburg (2018), lacks on the robust capacity of efficiently reject false positives. Even though we demonstrated that phase-folded flux alone is a very strong information to describe transiting planets, it is not completely sufficient by what was discussed in subsection 2.2.3. There, with the help of Table 2, we commented that a trend in recent literature is to include more and more astrophysical information as inputs for the models (Ansdell; others, 2018; Armstrong *et al.*, 2021; Valizadegan *et al.*, 2022). By mimicking the rigorous analysis made by an astrophysicist on all the parameters given by a data validation summary report generated from Kepler/TESS pipeline (Twicken; others, 2018), such as the one in Figure 25, ExoMiner could validate 301 new planets and enhance the state-of-the-art of deep learning usage on exoplanet detection.

The time series imaging has shown to be a method that produces large models and, therefore, the computational demand for them is high. Because of this downside, we still could not fully explore the potential of the method. However, since both 2D models tested here could generalize better on new data in comparison with 1D data, we think it is still worth to search for better models configurations. Besides phase-folded light curves, they should also try to follow the example of ExoMiner and take into consideration aspects such as the ones presented in Table 2. There is space to train the eventual upgraded models with data not only from Kepler, but also from K2, TESS and even with the injected/artificial data (Christiansen, 2017), as discussed in section 5.3. Besides, by what was discussed in subsections 2.1.1 and 2.1.2, we still can evaluate if time series imaging serves of any help on other exoplanet detection methods, such as radial velocity and gravitational microlensing.

Even if future tests on the time series imaging technique shows that it is not the way, the usage of deep learning in exoplanet detection and in astrophysics in general is here to stay. Considering technological novelties from this decade, such as the PLATO mission, we will have way more data than Kepler was ever able to provide and that we are still digesting. With this in mind, the spread of the knowledge on automatic astrophysical data processing with artificial intelligence has become essential. With this work, we contribute to this spread and also opened paths for new models we will develop in the future.

BIBLIOGRAPHY

ABADI, M. *et al.* TensorFlow: Large-scale machine learning on heterogeneous systems. 2015. Software available from tensorflow.org. Available at: https://www.tensorflow.org/.

Andres, A.; Otero, A.; Amavilah, V. Using deep learning neural networks to predict the knowledge economy index for developing and emerging economies. **Expert Systems with Applications**, Elsevier, v. 184, p. 115514, 2021.

Ansdell, M.; others. Scientific domain knowledge improves exoplanet transit classification with deep learning. **The Astrophysical Journal Letters**, The American Astronomical Society, v. 869, n. 1, p. L7, dec 2018. Available at: https://dx.doi.org/10.3847/2041-8213/aaf23b.

Armstrong, D. J.; Gamper, J.; Damoulas, T. Exoplanet validation with machine learning: 50 new validated Kepler planets. **Monthly Notices of the Royal Astronomical Society**, v. 504, n. 4, p. 5327–5344, 08 2021. ISSN 0035-8711. Available at: https://doi.org/10.1093/mnras/staa2498.

ASTROPY COLLABORATION *et al.* **The Astropy Project**: Building an Open-science Project and Status of the v2.0 Core Package. The Astrophysical Journal, v. 156, n. 3, p. 123, 09 2018.

Balaji, T.; Annavarapu, C.; Bablani, A. Machine learning algorithms for social media analysis: A survey. **Computer Science Review**, Elsevier, v. 40, p. 100395, 2021.

Barclay, T. Target pixel files. NASA, 05 2023. Available at: https://heasarc.gsfc.nasa.gov/docs/tess/Target-Pixel-File-Tutorial.html#.

Batalha, N. M.; others. **Pre-spectroscopic false-positive elimination of Kepler planet candidates**. The Astrophysical Journal, American Astronomical Society, v. 713, n. 2, p. L103–L108, 03 2010.

Bayu, W.; Setyanto, A. 3d cnn for micro expression detection. *In*: IEEE. **2022 5th International Conference on Information and Communications Technology (ICOIACT)**. [S.l.], 2022. p. 397–401.

BEAUGE, C.; NESVORNỲ, D. Emerging trends in a period–radius distribution of close-in planets. **The Astrophysical Journal**, IOP Publishing, v. 763, n. 1, p. 12, 2012.

Bengio, Y.; Courville, A.; Vincent, P. Representation learning: A review and new perspectives. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 35, n. 8, p. 1798–1828, 2013.

Bengio, Y.; others. Learning deep architectures for ai. **Foundations and trends® in Machine Learning**, Now Publishers, Inc., v. 2, n. 1, p. 1–127, 2009.

Boisnard, L.; Auvergne, M. Corot in brief. *In*: **The CoRoT Mission Pre-Launch Status-Stellar Seismology and Planet Finding**. [S.l.: s.n.], 2006. v. 1306, p. 19.

Borawar, L.; Kaur, R. Resnet: Solving vanishing gradient in deep networks. *In*: SPRINGER. **Proceedings of International Conference on Recent Trends in Computing: ICRTC 2022**. [S.l.], 2023. p. 235–247.

Borucki, W. J. Kepler mission: development and overview. **Reports on Progress in Physics**, IOP Publishing, v. 79, n. 3, p. 036901, feb 2016. Available at: https://dx.doi.org/10.1088/0034-4885/79/3/036901.

Borucki, W. J.; others. Kepler-4b: A hot neptune-like planet of a g0 star near main-sequence turnoff*. **The Astrophysical Journal Letters**, The American Astronomical Society, v. 713, n. 2, p. L126, mar 2010. Available at: https://dx.doi.org/10.1088/2041-8205/713/2/L126.

Borucki, W. J. *et al.* **Kepler Planet-Detection Mission**: Introduction and first results. Science, v. 327, n. 5968, p. 977–980, 2010.

Borucki, W. J.; others. Characteristics of planetary candidates observed by kepler. ii. analysis of the first four months of data. **The Astrophysical Journal**, The American Astronomical Society, v. 736, n. 1, p. 19, jun 2011. Available at: https://dx.doi.org/10.1088/0004-637X/736/1/19.

BOTTOU, L. *et al.* Stochastic gradient learning in neural networks. **Proceedings of Neuro-Nımes**, Nimes, v. 91, n. 8, p. 12, 1991.

Branco, P.; Torgo, L.; Ribeiro, R. A survey of predictive modelling under imbalanced distributions. 2015.

Breiman, L. **Random forests**. Machine learning, Springer, v. 45, n. 1, p. 5–32, 2001.

Brennan, P. Discoveries dashboard. **NASA**, NASA, 4 2023. Available at: https://exoplanets.nasa.gov/discovery/discoveries-dashboard/. Accessed on: 17 apr. 2023.

Broomhead, D.; Lowe, D. Radial basis functions, multi-variable functional interpolation and adaptive networks. **Royal signals and radar establishment Malvern (United Kingdom)**, v. 25, n. 3, p. 1–8, 1988.

Bryson, S. T. *et al.* **Selecting pixels for Kepler downlink**. *In*: Software and Cyberinfrastructure for Astronomy. [S.l.]: SPIE, 2010. v. 7740, p. 526 – 537.

Cabrera, J.; Csizmadia, S.; Lehmann, H. *et al.* The planetary system to kic 11442793: a compact analogue to the solar system. **The Astrophysical Journal**, IOP Publishing, v. 781, n. 1, p. 18, 2013.

CAMPANHARO, A. S.; SIRER, M. I.; MALMGREN, R. D.; RAMOS, F. M.; AMARAL, L. A. N. Duality between time series and networks. **PloS one**, Public Library of Science San Francisco, USA, v. 6, n. 8, p. e23378, 2011.

Campbell, B.; Walker, G. A. H.; Yang, S. **A Search for Substellar Companions to Solar-type Stars**. The Astrophysical Journal, v. 331, p. 902, Aug. 1988.

Catanzarite, J. **Autovetter Planet Candidate Catalog for Q1-Q17 Data Release 24**. Astronomy & Astrophysics, 2015.

Chakraborty, S.; Mali, K. An overview of biomedical image analysis from the deep learning perspective. **Research Anthology on Improving Medical Imaging Techniques for Analysis and Intervention**, IGI Global, p. 43–59, 2023.

Chatsiou, K.; Mikhaylov, S. Deep learning for political science. **arXiv preprint arXiv:2005.06540**, 2020.

CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2017. p. 1251–1258.

Christiansen, J. L. Planet detection metrics: Pixel-level transit injection tests of pipeline detection efficiency for data release 25. **Kepler Science Document KSCI-19110-001**, p. 18, 2017.

Claesen, M.; Moor, B. D. Hyperparameter search in machine learning. 2015.

Cleve, J. E. V.; Caldwell, D. A. **Kepler Instrument Handbook**. 2016. 1 p. Kepler Science Document KSCI-19033-002.

Cleve, J. E. V. *et al.* **That's How We Roll**: The NASA K2 mission science products and their performance metrics. Publications of the Astronomical Society of the Pacific, IOP Publishing, v. 128, n. 965, p. 075002, 06 2016.

Coughlin, J.; others. **Humans Need Not Apply**: Robotization of Kepler Planet Candidate Vetting. *In*: American Astronomical Society Meeting Abstracts #225. [S.l.: s.n.], 2015. (American Astronomical Society Meeting Abstracts, v. 225), p. 202.05.

Coughlin, J. L. *et al.* Planetary candidates observed by kepler. vii. the first fully uniform catalog based on the entire 48-month data set (q1–q17 dr24). **The Astrophysical Journal Supplement Series**, IOP Publishing, v. 224, n. 1, p. 12, 2016.

Dai, W.; Yang, Q.; Xue, G.; Yu, Y. Boosting for transfer learning. *In*: **Proceedings of the 24th international conference on Machine learning**. [S.l.: s.n.], 2007. p. 193–200.

Dattilo, A. *et al.* Identifying exoplanets with deep learning. ii. two new super-earths uncovered by a neural network in k2 data. **The Astronomical Journal**, The American Astronomical Society, v. 157, n. 5, p. 169, apr 2019. Available at: https://dx.doi.org/10.3847/1538-3881/ab0e12.

Daubechies, I. Orthonormal bases of compactly supported wavelets. **Communications on pure and applied mathematics**, Wiley Online Library, v. 41, n. 7, p. 909–996, 1988.

de Beurs, Z. L.; Vanderburg, A.; Shallue, C. J. *et al.* Identifying exoplanets with deep learning. iv. removing stellar activity signals from radial velocity measurements using neural networks. **The Astronomical Journal**, IOP Publishing, v. 164, n. 2, p. 49, 2022.

Deepchecks, A. What is data binning. 07 2023. Available at: https://deepchecks.com/glossary/data-binning/.

Deng, L.; Yu, D.; others. Deep learning: methods and applications. **Foundations and trends® in signal processing**, Now Publishers, Inc., v. 7, n. 3–4, p. 197–387, 2014.

DÍAZ, R. F.; ALMENARA, J. M.; SANTERNE, A.; MOUTOU, C.; LETHUILLIER, A.; DELEUIL, M. pastis: Bayesian extrasolar planet validation–i. general framework, models, and performance. **Monthly Notices of the Royal Astronomical Society**, Oxford University Press, v. 441, n. 2, p. 983–1004, 2014.

Dimiduk, D.; Holm, E.; Niezgoda, S. Perspectives on the impact of machine learning, deep learning, and artificial intelligence on materials, processes, and structures engineering. **Integrating Materials and Manufacturing Innovation**, Springer, v. 7, p. 157–172, 2018.

Eastman, J.; Siverd, R.; Gaudi, B. S. Achieving better than 1 minute accuracy in the heliocentric and barycentric julian dates. **Publications of the Astronomical Society of the Pacific**, IOP Publishing, v. 122, n. 894, p. 935, 2010.

Eckmann, J.; Kamphorst, S. O.; Ruelle, D. Recurrence plots of dynamical systems. **Europhysics Letters**, v. 4, n. 9, p. 973, nov 1987. Available at: https://dx.doi.org/10.1209/0295-5075/4/9/004.

Eggenberger, A.; Mayor, M.; Naef, D.; Pepe, F.; Queloz, D.; Santos, N.; Udry, S.; Lovis, C. The coralie survey for southern extrasolar planets-xiv. hd 142022 b: a long-period planetary companion in a wide binary. **Astronomy & Astrophysics**, EDP Sciences, v. 447, n. 3, p. 1159–1163, 2006.

Etangs, A. L. D.; Lissauer, J. J. The iau working definition of an exoplanet. **New Astronomy Reviews**, Elsevier, p. 101641, 2022.

Fan, C.; Sun, Y.; Zhao, Y.; Song, M.; Wang, J. Deep learning-based feature engineering methods for improved building energy prediction. **Applied energy**, Elsevier, v. 240, p. 35–45, 2019.

Faouzi, J.; Janati, H. pyts: A python package for time series classification. **Journal of Machine Learning Research**, v. 21, n. 46, p. 1–6, 2020. Available at: http://jmlr.org/papers/v21/19-763.html.

FERNÁNDEZ, A.; GARCíA, S.; GALAR, M.; PRATI, R. C.; KRAWCZYK, B.; HERRERA, F. **Learning from Imbalanced Data Sets**. 1st ed.. ed. [S.l.]: Springer International Publishing, 2018. ISBN 978-3-319-98073-7,978-3-319-98074-4.

Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. **Biological cybernetics**, Springer, v. 36, n. 4, p. 193–202, 1980.

Gagniuc, P. **Markov chains: from theory to implementation and experimentation**. [S.l.]: John Wiley & Sons, 2017.

Gilliland, R. L. *et al.* **Initial characteristics of Kepler short cadence data**. The Astrophysical Journal Letters, IOP Publishing, v. 713, n. 2, p. L160, 2010.

Gonzalez, G.; Brownlee, D.; Ward, P. **The Galactic Habitable Zone**: Galactic chemical evolution. Icarus, v. 152, n. 1, p. 185–200, 2001. ISSN 0019-1035. Available at: https://www.sciencedirect.com/science/article/pii/S0019103501966175.

Gonzalez, R.; Woods, R. **Digital Image Processing**. 4. ed. [S.l.]: Pearson, 2018.

Gregory, P.; Loredo, T. **A New Method for the Detection of a Periodic Signal of Unknown Shape and Period**. The Astrophysical Journal, v. 398, p. 146, 10 1992.

Greicius, T. **Overlooked Treasure:** The First Evidence of Exoplanets. NASA, 2017. Available at: https://www.nasa.gov/feature/jpl/overlooked-treasure-the-first-evidence-of-exoplanets. Accessed on: 29 nov. 2021.

Grubbs, F. E. **Procedures for Detecting Outlying Observations in Samples**. Technometrics, Taylor & Francis, v. 11, n. 1, p. 1–21, 1969.

Géron, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**. 3. ed. [S.l.]: O'Reilly Media, Inc., 2022.

Hafiz, A.; Bhat, G. A survey of deep learning techniques for medical diagnosis. **Information and Communication Technology for Sustainable Development: Proceedings of ICT4SD 2018**, Springer, p. 161–170, 2020.

HARTMAN, J. D.; BAKOS, G. A. Vartools: a program for analyzing astronomical time-series data. 2016.

He, H.; Ma, Y. **Imbalanced Learning: Foundations, Algorithms, and Applications**. [S.l.]: Wiley-IEEE Press, 2013. ISBN 9781118074626,9781118646106.

He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778.

He, Z. Deep learning in image classification: A survey report. *In*: **2020 2nd International Conference on Information Technology and Computer Application (ITCA)**. [S.l.: s.n.], 2020. p. 174–177.

Hellier, C. Wasp planets. **WASP Planets**, 06 2019. Available at: https://wasp-planets.net/about/. Accessed on: 29 nov. 2021.

Hippke, M.; Angerhausen, D. **Photometry's bright future**: detecting solar system analogs with future space telescopes. The Astrophysical Journal, American Astronomical Society, v. 810, n. 1, p. 29, 08 2015.

Hippke, M.; David, T.; Mulders, G.; Heller, R. Wōtan: Comprehensive time-series detrending in python. **The Astronomical Journal**, American Astronomical Society, v. 158, n. 4, p. 143, 09 2019. ISSN 1538-3881. Available at: http://dx.doi.org/10.3847/1538-3881/ab3984.

Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. **International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems**, World Scientific, v. 6, n. 02, p. 107–116, 1998.

Holman, M.; Fabrycky, D.; Ragozzine, D.; Ford, E.; Steffen, J.; Welsh, W.; Lissauer, J.; Latham, D.; Marcy, G.; Walkowicz, L.; others. Kepler-9: a system of multiple planets transiting a sun-like star, confirmed by timing variations. **science**, American Association for the Advancement of Science, v. 330, n. 6000, p. 51–54, 2010.

Howard, A.; Park, E.; Kan, W. Imagenet object localization challenge. Kaggle, 2018. Available at: https://kaggle.com/competitions/imagenet-object-localization-challenge.

Howard, A. W. Observed properties of extrasolar planets. **Science**, American Association for the Advancement of Science, v. 340, n. 6132, p. 572–576, 2013.

Howell, S. B. *et al.* The k2 mission: Characterization and early results. **Publications of the Astronomical Society of the Pacific**, University of Chicago Press, v. 126, n. 938, p. 398, apr 2014.

Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2018. p. 7132–7141.

Huang, R.; Dong, H.; Yin, G.; Fu, Q. Ensembling 3d cnn framework for video recognition. *In*: IEEE. **2019 International Joint Conference on Neural Networks (IJCNN)**. [S.l.], 2019. p. 1–7.

Iman, M.; Arabnia, H.; Rasheed, K. A review of deep transfer learning and recent advancements. **Technologies**, MDPI, v. 11, n. 2, p. 40, 2023.

Jenkins, J.; McCauliff, S.; Burke, C.; Seader, S.; Twicken, J.; Klaus, T.; Sanderfer, D.; Srivastava, A.; Haas, M. Auto-vetting transiting planet candidates identified by the kepler pipeline. **Proceedings of the International Astronomical Union**, Cambridge University Press, v. 8, n. S293, p. 94–99, 2012.

Jenkins, J. M. The impact of solar-like variability on the detectability of transiting terrestrial planets. **The Astrophysical Journal**, v. 575, n. 1, p. 493, aug 2002. Available at: https://dx.doi.org/10.1086/341136.

Jenkins, J. M.; Caldwell, D. A.; Borucki, W. J. Some tests to establish confidence in planets discovered by transit photometry. **The Astrophysical Journal**, v. 564, n. 1, p. 495, jan 2002. Available at: https://dx.doi.org/10.1086/324143.

Jenkins, J. M.; others. Discovery and rossiter–mclaughlin effect of exoplanet kepler-8b*. **The Astrophysical Journal**, The American Astronomical Society, v. 724, n. 2, p. 1108, nov 2010. Available at: https://dx.doi.org/10.1088/0004-637X/724/2/1108.

Jenkins, J. M. *et al.* **Initial characteristics of Kepler long cadence data for detecting transiting planets**. The Astrophysical Journal Letters, IOP Publishing, v. 713, n. 2, p. L120, 2010.

Jenkins, J. M. *et al.* **Overview of the Kepler science processing pipeline**. The Astrophysical Journal, American Astronomical Society, v. 713, n. 2, p. L87–L91, 03 2010.

Jenkins, J. M. *et al.* **Transiting planet search in the Kepler pipeline**. *In*: Software and Cyberinfrastructure for Astronomy. [S.l.: s.n.], 2010. v. 7740, p. 77400D.

Jiao, L.; Zhao, J. A survey on the new generation of deep learning in image processing. **IEEE Access**, v. 7, p. 172231–172263, 2019.

Johnson, D. H. **Signal-to-noise ratio**. Scholarpedia, v. 1, n. 12, p. 2088, 2006.

KAUR, R.; SINGH, S. A comprehensive review of object detection with deep learning. **Digital Signal Processing**, v. 132, p. 103812, 2023. ISSN 1051-2004. Available at: https://www.sciencedirect.com/science/article/pii/S1051200422004298.

Ke, G. *et al.* **LightGBM**: A highly efficient gradient boosting decision tree. *In*: Advances in Neural Information Processing Systems. [S.l.]: Curran Associates, Inc., 2017. v. 30.

Keogh, E. J.; Pazzani, M. J. Scaling up dynamic time warping for datamining applications. *In*: **Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.: s.n.], 2000. p. 285–289.

Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. 2017.

Kiranyaz, S.; Avci, O.; Abdeljaber, O.; Ince, T.; Gabbouj, M.; Inman, D. 1d convolutional neural networks and applications: A survey. **Mechanical systems and signal processing**, Elsevier, v. 151, p. 107398, 2021.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; others. Overcoming catastrophic forgetting in neural networks. **Proceedings of the national academy of sciences**, National Acad Sciences, v. 114, n. 13, p. 3521–3526, 2017.

Kottursamy, K. A review on finding efficient approach to detect customer emotion analysis using deep learning analysis. **Journal of Trends in Computer Science and Smart Technology**, v. 3, n. 2, p. 95–113, 2021.

KovÁcs, G.; Zucker, S.; Mazeh, T. **A box-fitting algorithm in the search for periodic transits**. Astronomy & Astrophysics, EDP Sciences, v. 391, n. 1, p. 369–377, 07 2002. ISSN 1432-0746.

Krizhevsky, A.; Sutskever, I.; Hinton, G. Imagenet classification with deep convolutional neural networks. *In*: PEREIRA, F.; BURGES, C.; BOTTOU, L.; WEINBERGER, K. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2012. v. 25. Available at: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Kunimoto, M.; Matthews, J. M.; Rowe, J. F.; Hoffman, K. Lifting transit signals from the kepler noise floor. i. discovery of a warm neptune. **The Astronomical Journal**, The American Astronomical Society, v. 155, n. 1, p. 43, jan 2018. Available at: https://dx.doi.org/10.3847/1538-3881/aaa005.

LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. **nature**, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, Ieee, v. 86, n. 11, p. 2278–2324, 1998.

LeCun, Y.; Cortes, C.; Burges, C. Mnist handwritten digit database. 2009. Available at: http://yann.lecun.com/exdb/mnist/.

Lefkowitz, M. Professor's perceptron paved the way for ai – 60 years too soon. 09 2019. Available at: https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon. Accessed on: 15 may. 2023.

Li, X.; Li, M.; Yan, P.; Li, G.; Jiang, Y.; Luo, H.; Yin, S. Deep learning attention mechanism in medical image analysis: Basics and beyonds. **International Journal of Network Dynamics and Intelligence**, p. 93–116, 2023.

Lightkurve Collaboration *et al.* **Lightkurve**: Kepler and TESS time series analysis in Python. 2018. Astrophysics Source Code Library.

Lin, Y.; Zhang, W. Towards a novel interface design framework: function–behavior–state paradigm. **International journal of human-computer studies**, Elsevier, v. 61, n. 3, p. 259–297, 2004.

Lissauer, J. J.; FABRYCKY, D. C.; FORD, E. B.; BORUCKI, W. J.; FRESSIN, F.; MARCY, G. W.; OROSZ, J. A.; ROWE, J. F.; TORRES, G.; WELSH, W. F. *et al.* A closely packed system of low-mass, low-density planets transiting kepler-11. **Nature**, Nature Publishing Group UK London, v. 470, n. 7332, p. 53–58, 2011.

Lissauer, J. J.; ROWE, J. F.; JONTOF-HUTTER, D.; FABRYCKY, D. C.; FORD, E. B.; RAGOZZINE, D.; STEFFEN, J. H.; NIZAM, K. M. Updated catalog of kepler planet candidates: Focus on accuracy and orbital periods. 2023.

Liu, T.; Siegel, E.; Shen, D. Deep learning and medical image analysis for covid-19 diagnosis and prediction. **Annual Review of Biomedical Engineering**, Annual Reviews, v. 24, p. 179–201, 2022.

Lomb, N. **Least-squares frequency analysis of unequally spaced data**. Astrophysics and space science, Springer, v. 39, n. 2, p. 447–462, 1976.

Long, W.; Lu, Z.; Cui, L. Deep learning-based feature engineering for stock price movement prediction. **Knowledge-Based Systems**, Elsevier, v. 164, p. 163–173, 2019.

Luo, Y.; Jiang, G.; Yu, T.; Liu, Y.; Vo, L.; Ding, H.; Su, Y.; Qian, W.; Zhao, H.; Peng, J. Ecnet is an evolutionary context-integrated deep learning framework for protein engineering. **Nature communications**, Nature Publishing Group UK London, v. 12, n. 1, p. 5743, 2021.

Mackay, C. D. Charge-coupled devices in astronomy. **Annual review of astronomy and astrophysics**, Annual Reviews 4139 El Camino Way, PO Box 10139, Palo Alto, CA 94303-0139, USA, v. 24, n. 1, p. 255–283, 1986.

Maliar, L.; Maliar, S.; Winant, P. Deep learning for solving dynamic economic models. **Journal of Monetary Economics**, Elsevier, v. 122, p. 76–101, 2021.

Mallama, A. The spherical bolometric albedo of planet mercury. **arXiv preprint arXiv:1703.02670**, 2017.

Masita, K.; Hasan, A.; Shongwe, T. Deep learning in object detection: a review. *In*: **2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)**. [S.l.: s.n.], 2020. p. 1–11.

Mayor, M.; Queloz, D. A Jupiter-mass companion to a solar-type star. **Nature**, v. 378, n. 6555, p. 355–359, Nov. 1995.

McCauliff, S. D. *et al.* **Automatic classification of Kepler planetary transit candidates**. The Astrophysical Journal, American Astronomical Society, v. 806, n. 1, p. 6, 06 2015.

McCulloch, W.; Pitts, W. **A logical calculus of the ideas immanent in nervous activity**. The bulletin of mathematical biophysics, Springer, v. 5, n. 4, p. 115–133, 1943.

Mitchell, T. **Machine Learning**. [S.l.]: McGraw-Hill, 1997. (McGraw-Hill International Editions). ISBN 9780071154673.

Morris, R. L. *et al.* **Kepler Data Processing Handbook**: Photometric analysis. Kepler Science Document KSCI-19081-003, p. 6, 2020.

Morton, T. D. An efficient automated validation procedure for exoplanet transit candidates. **The Astrophysical Journal**, The American Astronomical Society, v. 761, n. 1, p. 6, nov 2012. Available at: https://dx.doi.org/10.1088/0004-637X/761/1/6.

Morton, T. D.; Bryson, S. T.; Coughlin, J. L.; Rowe, J. F.; Ravichandran, G.; Petigura, E. A.; Haas, M. R.; Batalha, N. M. False positive probabilities for all kepler objects of interest: 1284 newly validated planets and 428 likely false positives. **The Astrophysical Journal**, The American Astronomical Society, v. 822, n. 2, p. 86, may 2016. Available at: https://dx.doi.org/10.3847/0004-637X/822/2/86.

Morton, T. D.; Johnson, J. A. On the low false positive probabilities of kepler planet candidates. **The Astrophysical Journal**, The American Astronomical Society, v. 738, n. 2, p. 170, aug 2011. Available at: https://dx.doi.org/10.1088/0004-637X/738/2/170.

Mu, R.; Zeng, X. A review of deep learning research. **KSII Transactions on Internet and Information Systems (TIIS)**, Korean Society for Internet Information, v. 13, n. 4, p. 1738–1764, 2019.

NASA. Kepler space telescope. 10 2018. Available at: https://exoplanets.nasa.gov/keplerscience/. Accessed on: 17 apr. 2023.

NASA. Kepler's legacy: discoveries and more. 10 2018. Available at: https://exoplanets.nasa.gov/keplerscience/.

NASA. Transiting exoplanets survey satellite (tess) - exoplanet exploration: Planets beyond our solar system. 02 2019. Available at: https://exoplanets.nasa.gov/tess/. Accessed on: 17 apr. 2023.

NASA. Exoplanet and candidate statitics. 2023. Available at: https://exoplanetarchive.ipac.caltech.edu/docs/counts_detail.html.

Ng, A.; others. Unsupervised feature learning and deep learning tutorial. 2017. Available at: http://ufldl.stanford.edu/tutorial/supervised/FeatureExtractionUsingConvolution/.

Nieto, L.; Díaz, R. Exoplannet: A deep learning algorithm to detect and identify planetary signals in radial velocity data. **arXiv preprint arXiv:2303.09335**, 2023.

NYAWA, S.; TCHUENTE, D.; FOSSO-WAMBA, S. Covid-19 vaccine hesitancy: a social media analysis using deep learning. **Annals of Operations Research**, Springer, p. 1–39, 2022.

O'MALLEY, T.; BURSZTEIN, E.; LONG, J.; CHOLLET, F.; JIN, H.; INVERNIZZI, L. *et al.* Kerastuner. 2019.

OTTONI, A. L. C.; SOUZA, A. M.; NOVO, M. S. Automated hyperparameter tuning for crack image classification with deep learning. **Soft Computing**, Springer, p. 1–20, 2023.

Pascual-Granado, J.; Garrido, R.; Suárez, J. **MIARMA**: A minimal-loss information method for filling gaps in time series-application to corot light curves. Astronomy & Astrophysics, EDP Sciences, v. 575, p. A78, 2015.

Paszke, A.; others. Pytorch: An imperative style, high-performance deep learning library. 2019.

Pedregosa, F.; others. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

Perryman, M. **The exoplanet handbook**. [S.l.]: Cambridge university press, 2018.

Poleski, R.; Yee, J. C. Modeling microlensing events with mulensmodel. **Astronomy and computing**, Elsevier, v. 26, p. 35–49, 2019.

Prša, A.; Zwitter, T. **A computational guide to physics of eclipsing binaries**. i. demonstrations and perspectives. The Astrophysical Journal, IOP Publishing, v. 628, n. 1, p. 426, 2005.

QI, C. R.; SU, H.; NIESSNER, M.; DAI, A.; YAN, M.; GUIBAS, L. J. Volumetric and multi-view cnns for object classification on 3d data. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 5648–5656.

Quemy, A. **Data Pipeline Selection and Optimization**. *In*: DOLAP. [S.l.: s.n.], 2019.

Rao, S.; Mahabal, A.; Rao, N.; Raghavendra, C. Nigraha: Machine-learning-based pipeline to identify and evaluate planet candidates from tess. **Monthly Notices of the Royal Astronomical Society**, Oxford University Press, v. 502, n. 2, p. 2845–2858, 2021.

Rauer, H.; Heras, A. M. Space missions for exoplanet science: Plato. **Handbook of Exoplanets**, p. 86, 2018.

Ricker, G. R. *et al.* Transiting Exoplanet Survey Satellite. **Journal of Astronomical Telescopes, Instruments, and Systems**, SPIE, v. 1, n. 1, p. 014003, 2014. Available at: https://doi.org/10.1117/1.JATIS.1.1.014003.

Roberts, A. Data binning challenges in production: How to bin to win. 03 2023. Available at: https://arize.com/blog-course/data-binning-production/.

ROJAS, R.; ROJAS, R. The backpropagation algorithm. **Neural networks: a systematic introduction**, Springer, p. 149–182, 1996.

RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C.; FEI-FEI, L. ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision (IJCV)**, v. 115, n. 3, p. 211–252, 2015.

SAILUNAZ, K.; ÖZYER, T.; ROKNE, J.; ALHAJJ, R. A survey of machine learning-based methods for covid-19 medical image analysis. **Medical & Biological Engineering & Computing**, Springer, p. 1–41, 2023.

Samuel, A. **Some studies in machine learning using the game of checkers**. IBM Journal of research and development, IBM, v. 3, n. 3, p. 210–229, 1959.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L. Mobilenetv2: Inverted residuals and linear bottlenecks. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2018. p. 4510–4520.

Savitzky, A.; Golay, M. J. **Smoothing and differentiation of data by simplified least squares procedures**. Analytical chemistry, ACS Publications, v. 36, n. 8, p. 1627–1639, 1964.

Scargle, J. **Studies in astronomical time series analysis**. ii-statistical aspects of spectral analysis of unevenly spaced data. The Astrophysical Journal, v. 263, p. 835–853, 1982.

SCHMIDHUBER, J. **Deep Learning in Neural Networks**: An overview. CoRR, abs/1404.7828, 2014.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. **Neural networks**, Elsevier, v. 61, p. 85–117, 2015.

Seager, S. **Exoplanets**. [S.l.]: University of Arizona Press, 2010. ISBN 9780816529452.

Shallue, C. J.; Vanderburg, A. Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90. **The Astronomical Journal**, American Astronomical Society, v. 155, n. 2, p. 94, 01 2018. Available at: https://doi.org/10.3847/1538-3881/aa9e09.

Shuttleworth, M.; Wilson, L. Type i error and type ii error - experimental errors in research. 11 2008. Available at: https://explorable.com/type-i-error.

Simon, M.; Rodner, E.; Denzler, J. Imagenet pre-trained models with batch normalization. 2016.

Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

Smith, S.; others. **The scientist and engineer's guide to digital signal processing**. [S.l.]: California Technical Pub. San Diego, 1997. 246–252 p.

Smith, S. W. **The Scientist and Engineer's Guide to Digital Signal Processing**. 2. ed. [S.l.]: California technical Publishing, 1999. ISBN 0-9660176-4-1,0-9660176-6-8,0-9660176-7-6,9780966017649.

Suganyadevi, S.; Seethalakshmi, V.; Balasamy, K. A review on deep learning in medical image analysis. **International Journal of Multimedia Information Retrieval**, Springer, v. 11, n. 1, p. 19–38, 2022.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. *In*: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015.

Tamuz, O.; Ségransan, D.; Udry, S.; Mayor, M.; Eggenberger, A.; Naef, D.; Pepe, F.; Queloz, D.; Santos, N.; Demory, B.-O. *et al.* The coralie survey for southern extra-solar planets-xv. discovery of two eccentric planets orbiting hd 4113 and hd 156846. **Astronomy & Astrophysics**, EDP Sciences, v. 480, n. 3, p. L33–L36, 2008.

Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; Le, Q. Mnasnet: Platform-aware neural architecture search for mobile. *In*: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. [S.l.: s.n.], 2019. p. 2820–2828.

Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. *In*: PMLR. **International conference on machine learning**. [S.l.], 2019. p. 6105–6114.

Tan, M.; Le, Q. V. Efficientnetv2: Smaller models and faster training. 2021.

Tenenbaum, P.; others. Detection of potential transit signals in the first three quarters of kepler mission data. **The Astrophysical Journal Supplement Series**, The American Astronomical Society, v. 199, n. 1, p. 24, mar 2012. Available at: https://dx.doi.org/10.1088/0067-0049/199/1/24.

THIEL, M.; ROMANO, M. C.; KURTHS, J. How much information is contained in a recurrence plot? **Physics Letters A**, v. 330, n. 5, p. 343–349, 2004. ISSN 0375-9601. Available at: https://www.sciencedirect.com/science/article/pii/S0375960104009922.

Thompson, S. E. *et al.* **Kepler Archive Manual**. [S.l.], 2016.

Thompson, S. E.; others. Planetary candidates observed by kepler. viii. a fully automated catalog with measured completeness and reliability based on data release 25. **The Astrophysical Journal Supplement Series**, The American Astronomical Society, v. 235, n. 2, p. 38, apr 2018. Available at: https://dx.doi.org/10.3847/1538-4365/aab4f9.

Torres, G.; others. Modeling kepler transit light curves as false positives: Rejection of blend scenarios for kepler-9, and validation of kepler-9 d, a super-earth-size planet in a multiple system. **The Astrophysical Journal**, The American Astronomical Society, v. 727, n. 1, p. 24, dec 2010. Available at: https://dx.doi.org/10.1088/0004-637X/727/1/24.

Torrey, L.; Shavlik, J. Transfer learning. *In*: **Handbook of research on machine learning applications and trends: algorithms, methods, and techniques**. [S.l.]: IGI global, 2010. p. 242–264.

Twicken, J.; Caldwell, D.; Jenkins, J.; Vanderspek, R.; Tenenbaum, P.; Smith, J.; Wohler, B.; Rose, M.; Ting, E.; Vanderspek, R.; others. Tess science data products description document: Exp-tess-arc-icd-0014 rev f. 2020.

Twicken, J. D. *et al.* **Presearch data conditioning in the Kepler Science Operations Center pipeline**. *In*: Software and Cyberinfrastructure for Astronomy. [S.l.]: SPIE, 2010. v. 7740, p. 673 – 684.

Twicken, J. D.; others. Kepler data validation i—architecture, diagnostic tests, and data products for vetting transiting planet candidates. **Publications of the Astronomical Society of the Pacific**, The Astronomical Society of the Pacific, v. 130, n. 988, p. 064502, apr 2018. Available at: https://dx.doi.org/10.1088/1538-3873/aab694.

Udry, S.; Mayor, M.; Clausen, J. *et al.* The coralie survey for southern extra-solar planets-x. a hot jupiter orbiting hd 73256. **Astronomy & Astrophysics**, EDP Sciences, v. 407, n. 2, p. 679–684, 2003.

UZSOY, A. S. M.; ROGERS, L. A.; PRICE, E. M. Radius and mass distribution of ultra-short-period planets. **The Astrophysical Journal**, IOP Publishing, v. 919, n. 1, p. 26, 2021.

Valizadegan, H.; Martinho, M. J. S.; Wilkens, L. S.; Jenkins, J. M.; Smith, J. C.; Caldwell, D. A.; Twicken, J. D.; Gerum, P. C. L.; Walia, N.; Hausknecht, K.; Lubin, N. Y.; Bryson, S. T.; Oza, N. C. Exominer: A highly accurate and explainable deep learning classifier that validates 301 new exoplanets. **The Astrophysical Journal**, The American Astronomical Society, v. 926, n. 2, p. 120, feb 2022. Available at: https://dx.doi.org/10.3847/1538-4357/ac4399.

Vallenari, A.; Brown, A.; Prusti, T. Gaia data release 3. summary of the content and survey properties. **Astronomy & Astrophysics**, EDP Sciences, 2022.

Vanderburg, A.; Johnson, J. A. A technique for extracting highly precise photometry for the two-WheeledKeplerMission. **Publications of the Astronomical Society of the Pacific**, IOP Publishing, v. 126, n. 944, p. 948–958, 10 2014. Available at: https://doi.org/10.1086/678764.

Verhoeven, G. Multispectral and hyperspectral imaging. *In*: ____. **The Encyclopedia of Archaeological Sciences**. John Wiley & Sons, Ltd, 2018. p. 1–4. ISBN 9781119188230. Available at: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119188230.saseas0395.

Wang, Z.; Oates, T. Imaging time-series to improve classification and imputation. **arXiv preprint arXiv:1506.00327**, 2015.

Wolszczan, A.; Frail, D. A. **A planetary system around the millisecond pulsar PSR1257 + 12**. Nature, v. 355, n. 6356, p. 145–147, Jan. 1992.

WU, X.; SAHOO, D.; HOI, S. C. Recent advances in deep learning for object detection. **Neurocomputing**, v. 396, p. 39–64, 2020. ISSN 0925-2312. Available at: https://www.sciencedirect.com/science/article/pii/S0925231220301430.

Yu, L. *et al.* Identifying exoplanets with deep learning. iii. automated triage and vetting of tess candidates. **The Astronomical Journal**, The American Astronomical Society, v. 158, n. 1, p. 25, jun 2019. Available at: https://dx.doi.org/10.3847/1538-3881/ab21d6.

ZBILUT, J. P.; THOMASSON, N.; WEBBER, C. L. Recurrence quantification analysis as a tool for nonlinear exploration of nonstationary cardiac signals. **Medical Engineering & Physics**, v. 24, n. 1, p. 53–60, 2002. ISSN 1350-4533. Non-linear Processing of Biomechanical Signals. Available at: https://www.sciencedirect.com/science/article/pii/S1350453301001126.

Zechmeister, M.; KÜrster, M. **The generalised Lomb-Scargle periodogram**. Astronomy & Astrophysics, EDP Sciences, v. 496, n. 2, p. 577–584, 01 2009. ISSN 1432-0746.

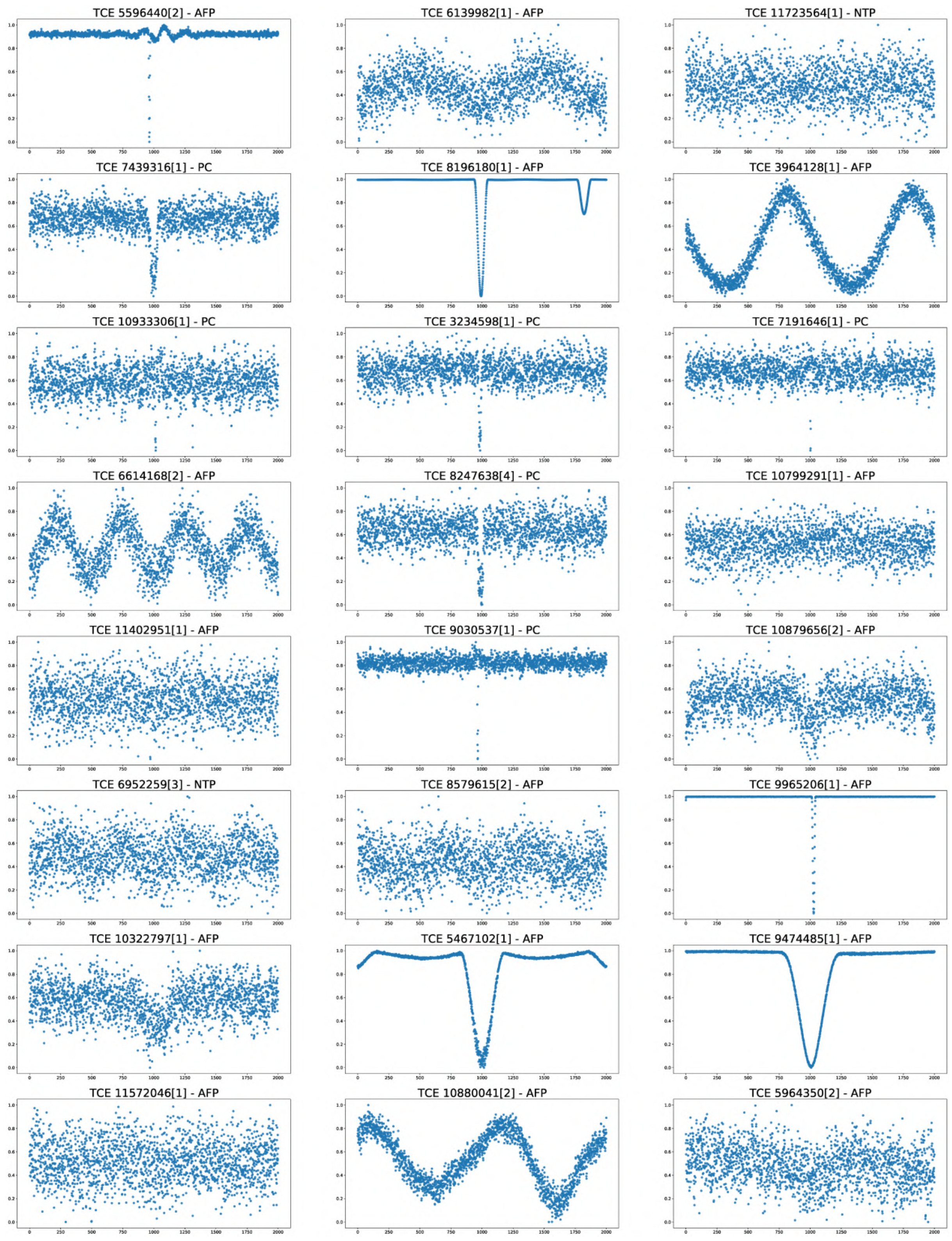
Zhang, W.; Li, H.; Li, Y.; Liu, H.; Chen, Y.; Ding, X. Application of deep learning algorithms in geotechnical engineering: a short critical review. **Artificial Intelligence Review**, Springer, p. 1–41, 2021.

Zhang, W.; Lin, Y.; Sinha, N. On the function-behavior-structure model for design. **Proceedings of the Canadian Engineering Education Association (CEEA)**, 2005.

Zhang, W.; Yang, G.; Lin, Y.; Ji, C.; Gupta, M. On definition of deep learning. *In*: IEEE. **2018 World automation congress (WAC)**. [S.l.], 2018. p. 1–5.

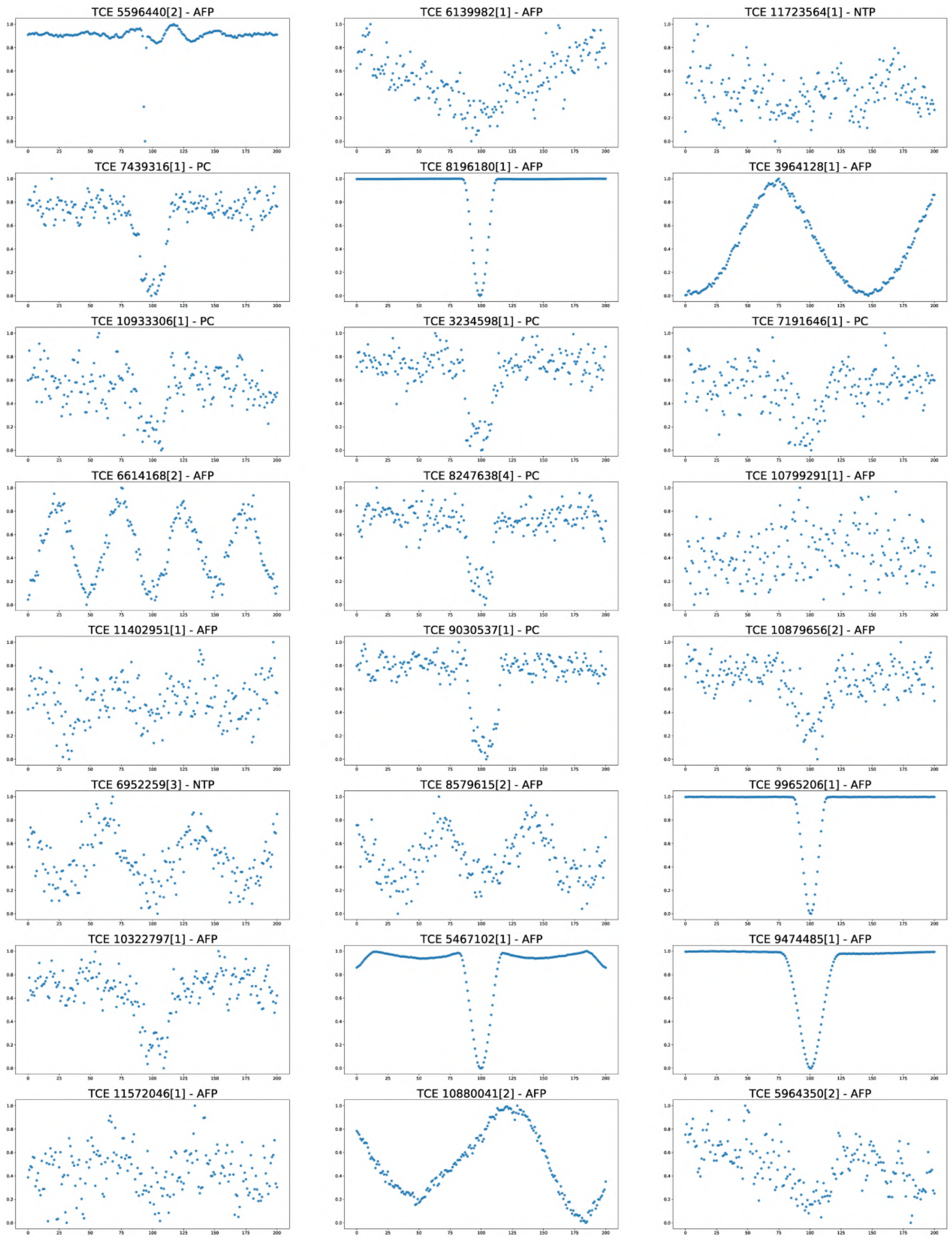
APPENDIX A – COMPLEMENTARY FIGURES

Figure 70 – Examples of global curves generated by the developed pipeline



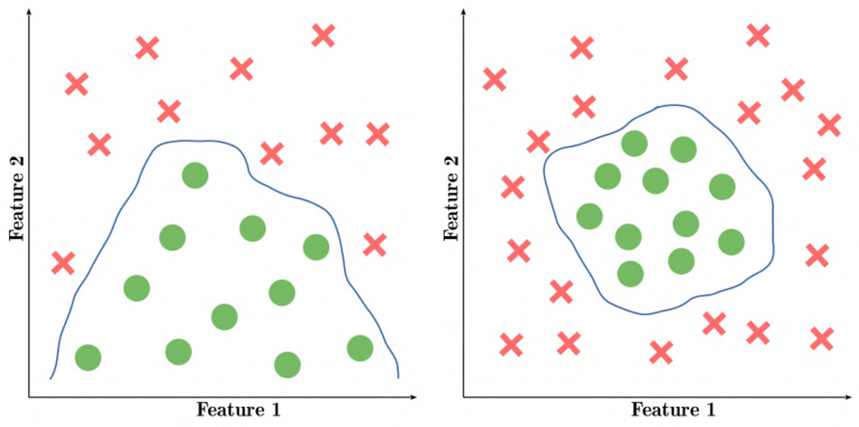
Source: Self elaboration.

Figure 71 – Examples of local curves generated by the developed pipeline



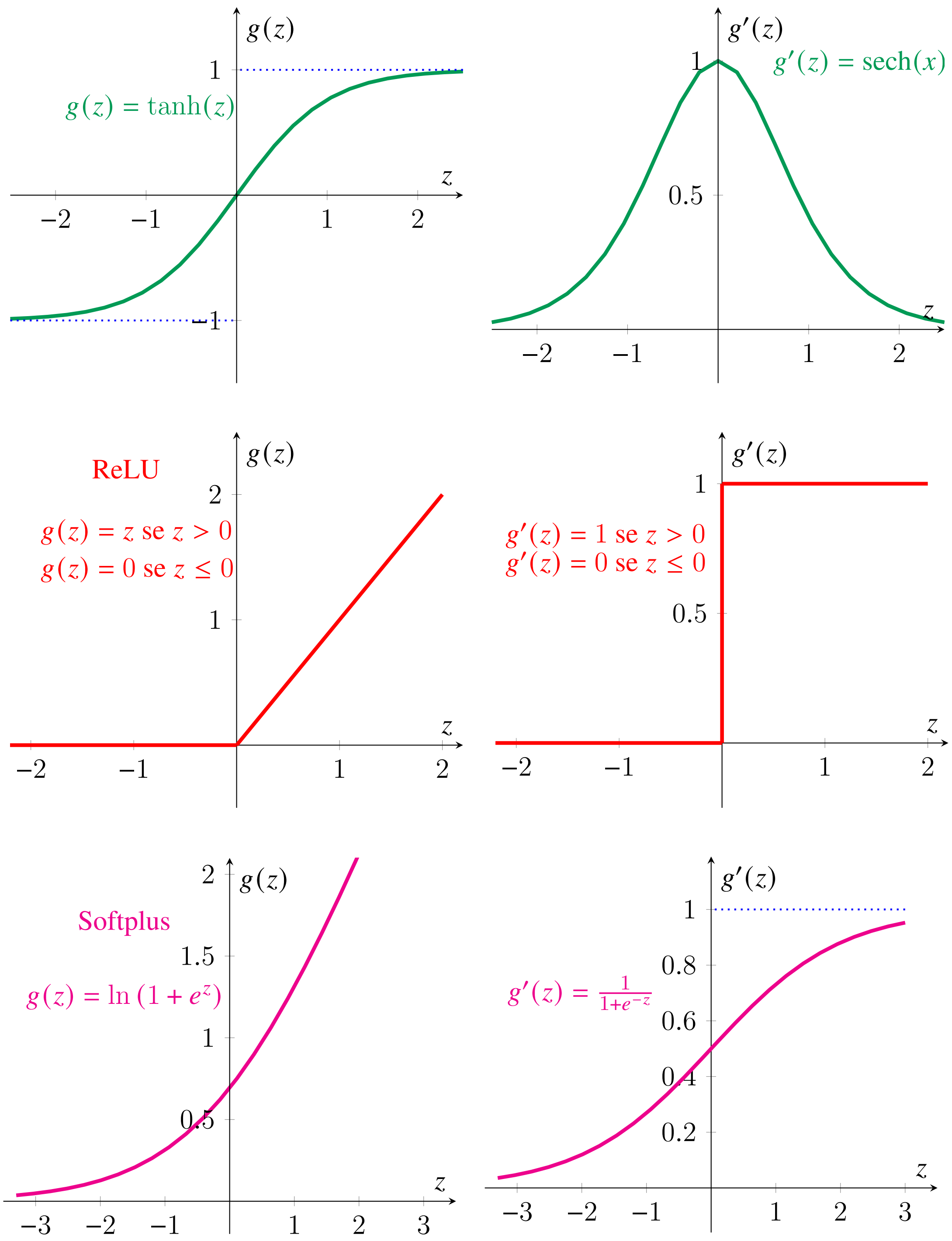
Source: Self elaboration.

Figure 72 – Helper examples of decision boundaries for classifying TCEs



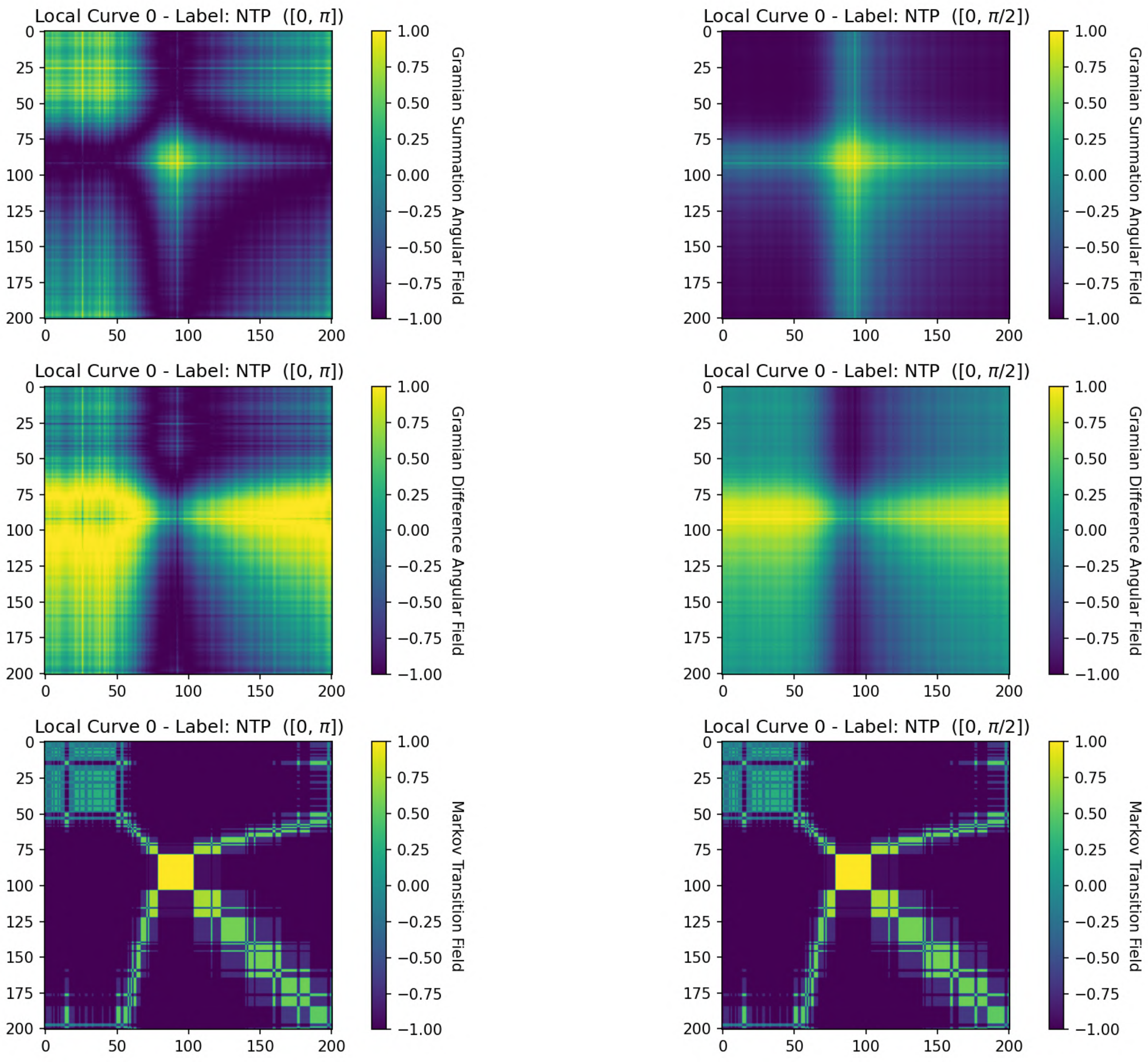
Source: Self elaboration.

Figure 73 – Other activation functions and their derivatives



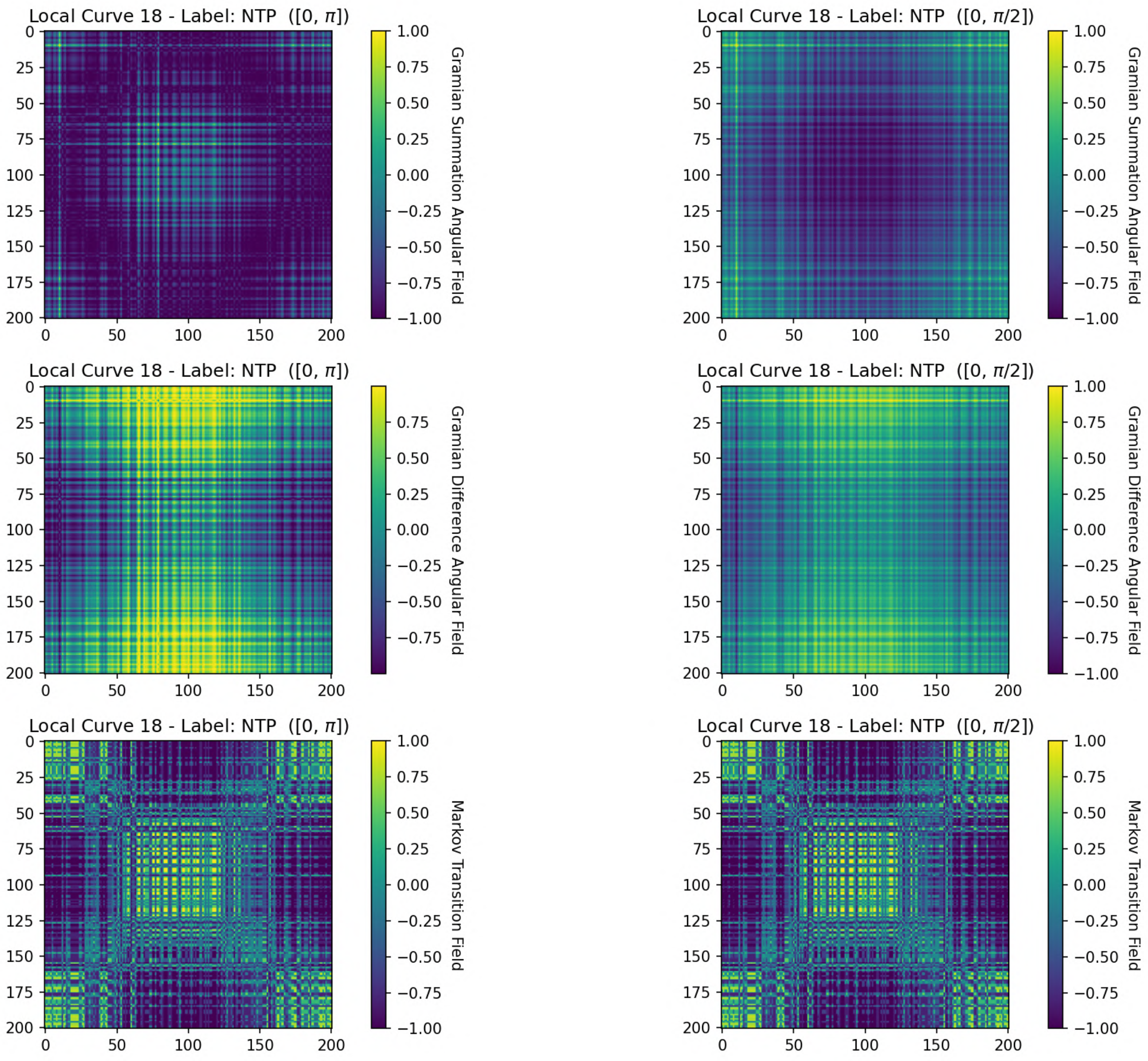
Source: Self elaboration.

Figure 74 – Images generated from a light curve for an NTP TCE (LC0)



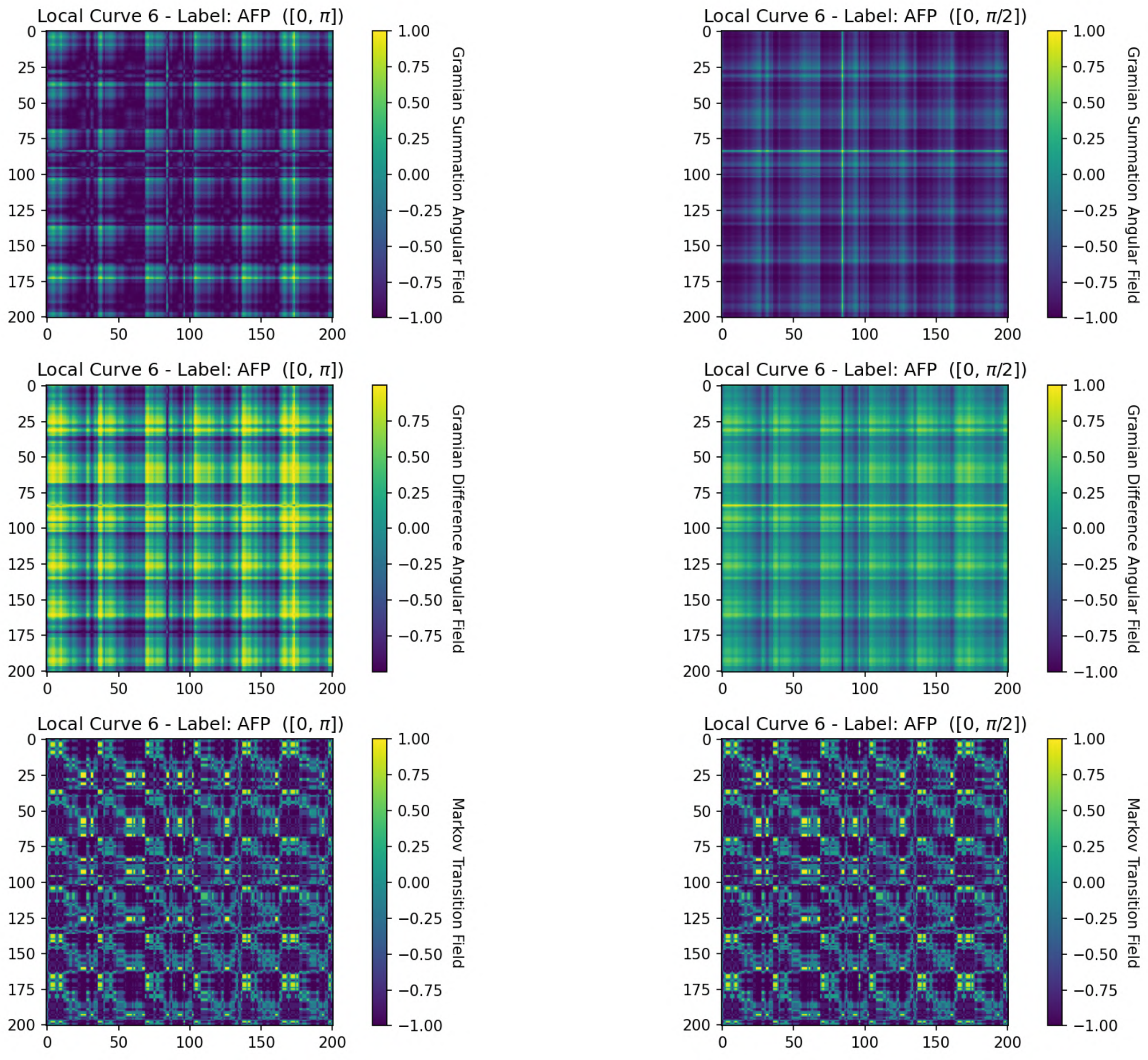
Source: Self elaboration.

Figure 75 – Images generated from a light curve for an NTP TCE (LC18)



Source: Self elaboration.

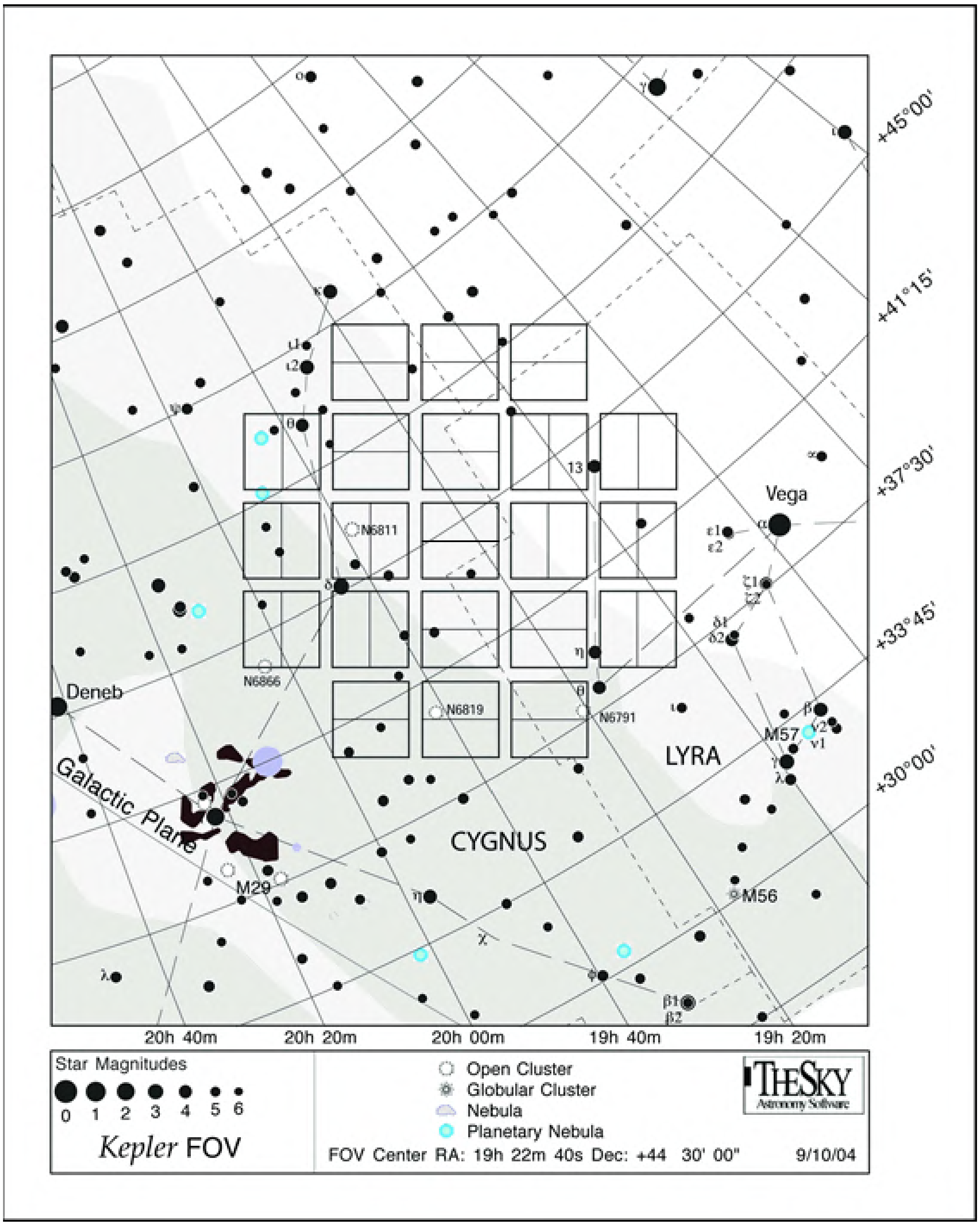
Figure 76 – Images generated from a light curve for an AFP TCE (LC6)



Source: Self elaboration.

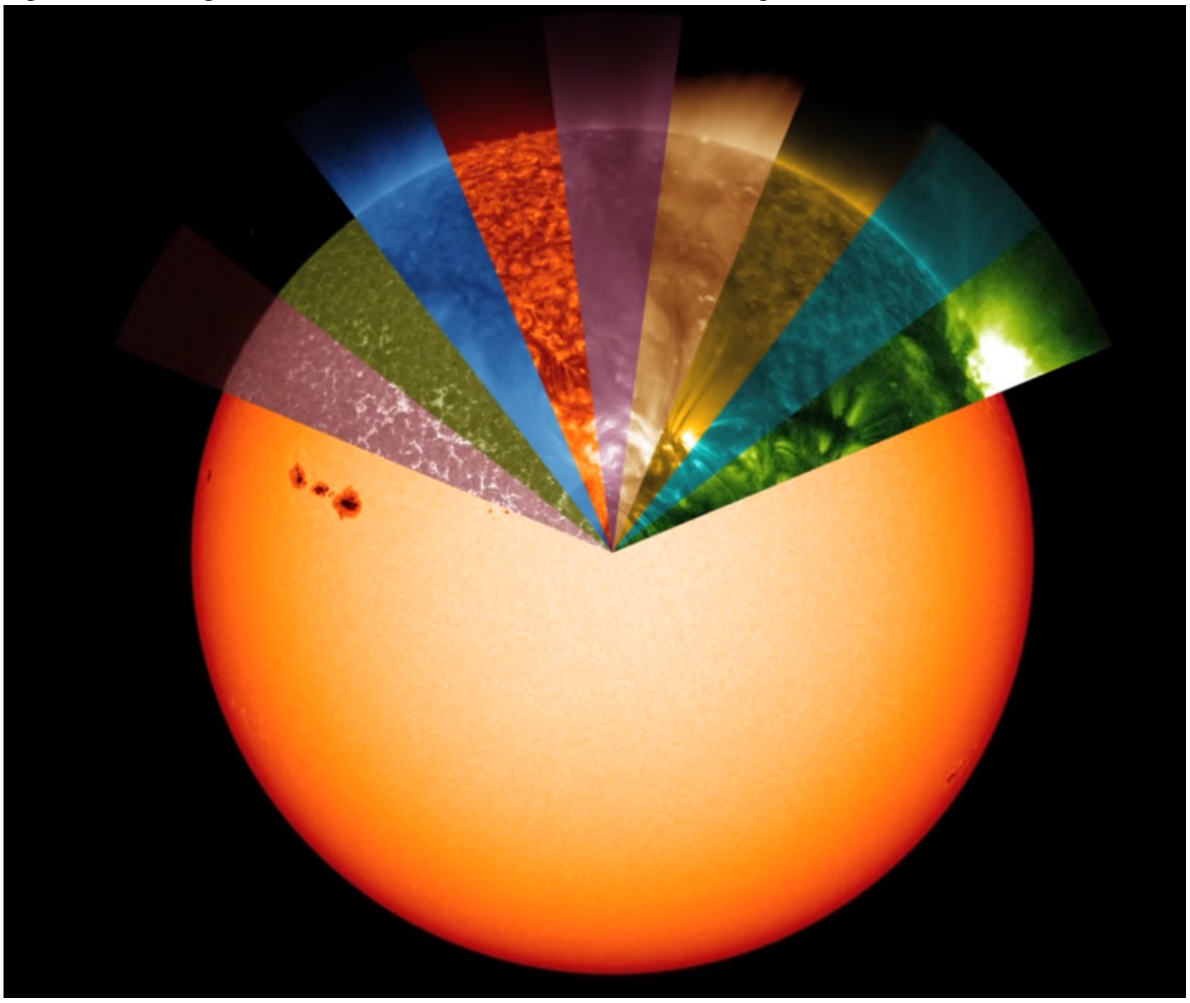
ANNEX A – AUXILIARY FIGURES

Figure 77 – Illustration of Kepler’s field of view



Source: MAST, available at https://archive.stsci.edu. Accessed on 30 April 2023.

Figure 78 – Image of the Sun observed in different wavelengths



Source: NASA Solar Dynamics Observatory, available at https://youtu.be/Sr9Aih_IlCs. Accessed on 29 May 2023.