Universidad Nacional del Sur
Department of Computer Science
and Engineering

Master's Computer Science
Final Project

Supervised Machine Learning applied to Astronomy
Use of Artificial Neural Networks for the detection and classification of galaxies.

Ezequiel Giménez

Director: Dr. Alejando Javier García
Co: Lic. Federico Martín Schmidt

Abstract

In this paper we explore the use of a supervised machine learning model for the detection and classification of galaxies in astronomical images. Specifically, we adopt a Convolutional Artificial Neural Network model where we take an architecture based on separable convolution layers per channel. On top of this, we present a method for extracting and proposing regions of an image that potentially contain galaxies. Finally, and adding all these components in a processing flow, we build an application capable of analyzing a region of the celestial sphere in search of galaxies and their morphological classification.

Introduction

In this project we propose to use a special type of Artificial Neural Networks for its application in the field of astronomy. The main objective of this thesis is to have an application to be able to recognize different types of galaxies in specific regions of the sky.

Understanding the why and how of our origin is one of humanity's greatest unknowns. In order to be able to answer and decipher the nuance that makes up his answer, we must delve into the universe we inhabit. A vitally important part of this is to inquire about the origin of the galaxies that compose it. These come in various shapes and colors; from spirals with many arms to vast elliptical galaxies. Understanding their dispersion, location, and the types of galaxies based on their shape, size, and color is essential to solving the puzzle we are asking ourselves. Figure 1 shows the classification created by the American astronomer Edwin Powell Hubble in 1926, which is widely used by astronomers today. This classification is known as *the morphological classification of galaxies*.

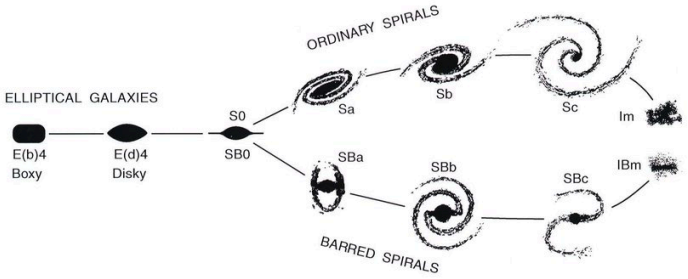


Figure 1: Edwin Hubble's classification of galaxies.

Historically, the process of classifying galaxies was done manually by a specialist person. Although this process was highly reliable, it has currently been hampered by the large amount of information collected. For example, the Sloan Digital Sky Survey (SDSS) has published approximately 800,000 images of astronomical objects in its latest publications. In turn, in the coming years, other observations such as the Wide-Field Infrared Survey Telescope (WFIRST; Spergel et al. 2013) and Euclid (Amiaux et al. 2012) will produce unprecedented amounts of images. It is precisely here where it is observed that manual analysis by a person

is impracticable. Taking this aspect into account, we can appreciate an opportunity to use some artificial intelligence techniques.

(*Artificial Neural Networks* RNA) are Machine *Learning* that has gained popularity in recent years due to its great versatility. Finding uses in various fields of study where they provide an efficient mechanism for the detection of non-linear patterns in their data samples. Thanks to this, the range of applications extends from weather forecasts, detection of medical pathologies and even to understanding the behavior of users on social networks. This flexibility is achieved due to the individual contribution of its atomic components, called artificial neurons. The ANNs are organized in a layered architecture which allows the learning of these patterns.

Although various types of RNA have been developed, we will pay particular attention to *Convolutional Neural Networks* (CNN), which have shown great potential in processing two-dimensional data, such as images. This type of RNA makes it possible to preserve the spatial and size relationships between information points and, in this way, to recognize characteristics that make up the image in different hierarchical degrees from a semantic point of view. Its name comes from an operation performed on one stage of its neuron architecture, where a filter runs through the image in different regions in an iterative process. This filter, in each run, collects the relevant information of the highest level of the section and thus decreases the amount of input data for the next stage of the convolutional network. In this way, a model with a remarkable ability to detect patterns in an image is obtained, which allows, among other applications, to classify the objects that are part of the image, in our case galaxies.

In this thesis a processing flow is described as shown in Figure 2. In this proposal, a user selects an area of interest in the sky and then seeks to identify objects that are potentially galaxies. Next, a classification and segmentation of the galaxy that is located there will be carried out. Once classified, the objects will be properly displayed for the appropriate use by the user.

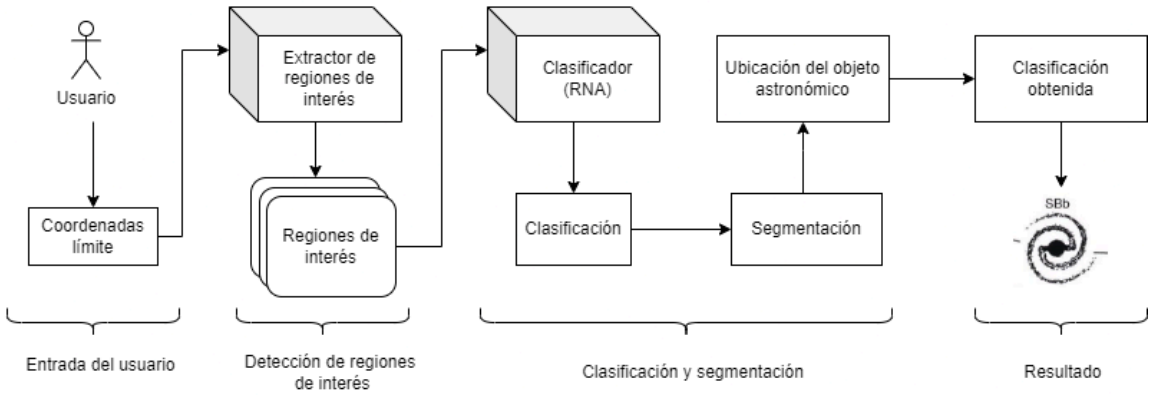


Figure 2. Processing flow with the different stages proposed.

In the first part of this report we will discuss concepts from the field of astronomy that are relevant to our work. Then, we will give a theoretical framework that is necessary to understand our processing flow proposal and its different stages. Next, we will explain this flow in detail, dividing it into stages following Figure 2. Then, we will dedicate a section to presenting a web application that allows the proposed flow to be fully functional. Finally, we will present conclusions of the study carried out.

Astronomy

In this section we will introduce various topics in the field of astronomy that are central to this project. The objective is to introduce the reader to the area so that they can have a better understanding of the proposed flow and its practical utility. We will begin by detailing the morphological classification of the galaxies as they are the focus of the project.

Morphological classification

Galaxies are fundamental pieces of the Universe, they are scattered throughout space, many being found alone and others in groups or even orbiting each other. Each one of them is a conglomeration not only of stars, but of dust, interstellar gases and dark matter, united by gravitational forces that maintain its structure. It is estimated that there are about 170 billion galaxies in the observable universe (ie, within a circle centered on the Solar System of radius 4.40 x 10^{26}m or 46.5 billion light-years).

Each one has a unique shape, which reflects the physical processes that occur within [1], such as the formation of new stars. In addition, its morphology can reveal the history of its evolution, from the forms it took throughout its existence to the mergers it became involved in with other galaxies. After numerous observations made over the years, it has been detected that its shape can vary from simple structures to complex formations with arms or even rings. Usually the characteristics that identify it are easily distinguishable and can be detected by visual inspection.

The Hubble Sequence

In 1926, the American astronomer Edwin Hubble proposed a system of morphological classification of galaxies based on features observed from Earth, called *the Hubble Sequence*. This system divides galaxies into 4 classifications: elliptical, lenticular, spiral, and irregular. Although it has been modified several times and can be very simple, it has managed to cover the different shapes of galaxies correctly and that is why it is currently one of the most accepted galaxy classification systems by the scientific community.

As we can see in Figure 3, the sequence takes the form of a tuning fork and we have divisions of the aforementioned classifications, beginning from left to right with elliptical galaxies and ending with spiral galaxies divided into two: with and without central bar. Initially, this sequence tried to describe the evolutionary process of a galaxy during its life cycle, but today it is known that this is not true, but it does not mean that the classifications are not valid.

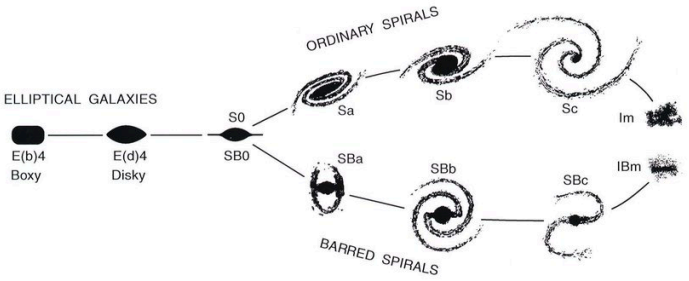


Figure 3. Diagram of the Hubble Sequence.

It should be noted that for various reasons many galaxies do not fit naturally into the Hubble Sequence, or cannot be easily identified as belonging to a class by simple visual inspection. According to this, in this project we have taken 4 groups of galaxies; on the one hand the *spiral* and *elliptical*, which are generally easy to discern visually, and on the other hand the *merging* and *edge-*.

Spirals Spiral

galaxies consist of a flattened disk with thin threads emerging from the center, forming a spiral structure. These threads or *arms* vary according to the orientation in which the galaxy rotates and how loose or close they are to the center of the galaxy. Inside it is a central bulge that has a higher concentration of stars and it is thought that a large part of the galaxies have a black hole inside. Its core is usually made up of stars older than its arms, and this is why a reddish hue can be seen in it. This bulb can also contain a bar that runs through it, and in this case, the arms start from its ends. In thesequence , those galaxies that contain the bar correspond to the types SBa, SBb and SBc and those that do not Sa, Sb and Sc. Both types of spiral galaxies are equally common and can contain several arms, although the most usual is to find galaxies between 2 and 4 arms. An example can be seen in Figure 4.

Figure 4. The galaxy UGC 12158 is an example of a centrally barred spiral galaxy.

Ellipticals

Figure 5 shows an image of an elliptical galaxy. Elliptical galaxies have a nearly uniform and smooth distribution of stars, resembling an ellipse with light intensity strongest in the center and fading to the sides. They range in size from dwarf ellipticals with billions of stars to gigantic ellipticals with trillions of stars. The stars contained in elliptical galaxies are usually older than those found in spiral galaxies. [2].

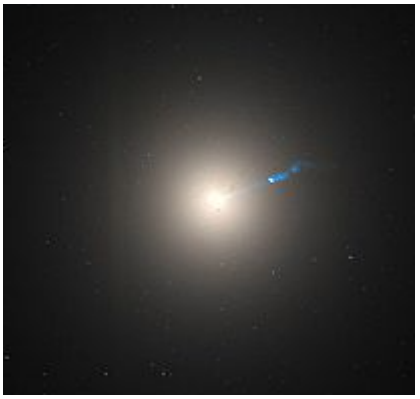


Figure 5. The galaxy M87 is a giant elliptical galaxy.

Merging

Although galaxies tend to be far apart, it often happens that they collide with each other, merging and forming larger galaxies. In this collision process, the galaxies are mixed and their stars do not collide due to the vast empty space that separates them, but it creates an opportunity for the creation of new stars, supernovae and even black holes. The shape of merging galaxies is distorted due to

the gravitational force that they carry changing their morphology, it has even been theorized that spiral galaxies become elliptical after having merged. When observing galaxies during the merger process (which can last millions of years) it is difficult to assign it one of the previous classifications and therefore it is assigned a merger *categorization*. Their detection and classification becomes relevant since measuring their occurrence assists in understanding the evolution of galaxies. Usually merging galaxies find a part of themselves stretched and loose, pulled and distorted by the gravitational pull of its counterpart. In Figure 6 we can see two spiral galaxies merging.

Figure 6. The gravitational interaction of two galaxies merging into the pair of spiral galaxies Arp 240 can be seen

.

The galaxies in the previous figures show their figure perpendicular to the field of view from where the observation was made, however, its forehead can be seen tilted in such a way that only its edge or edge can be seen. Although knowing the morphology of the galaxies that are found in this way requires a more detailed analysis, they allow obtaining other types of relevant information about the galaxies, such as studying the vertical distribution of the stars and gases that compose it, as well as know the size of the central bulb, and that is why identifying them becomes relevant. In Figure 7 we can see a galaxy of this category.

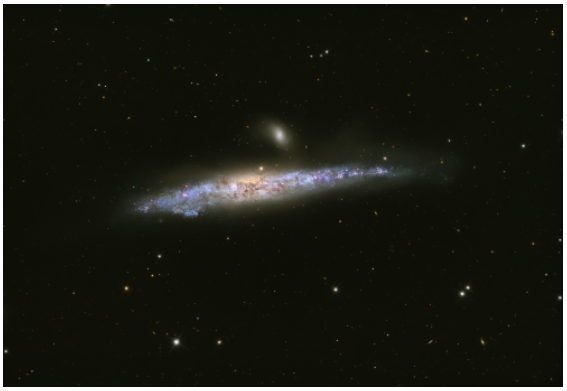


Figure 7. Galaxy NGC 4631 which is edge-on from our perspective.

Automation of the process

Due to the large number of images currently collected and the low availability of specialists in the area, visually classifying the morphology of galaxies through manual inspection is inefficient. Another approach to classifying galaxies on a large scale is to use crowdsourcing (open collaboration to the public or citizen science), in which any hobbyist can assist in the classification task by answering a series of simple questions that try to obtain a description of the Galaxy. One of the studies that has done this is Galaxy Zoo, and it demonstrated the potential of this method by making the answers converge to a common result thanks to the participation of thousands of people. However, it is important to note that the amount of data collected is growing rapidly and methods like crowdsourcing only increase the availability of people answering questions but the speed of sorting itself remains the same. This is why it is vital to apply Machine Learning techniques to speed up the process for large numbers of images.

Equatorial

coordinates Equatorial coordinates form a system that allows an object to be located on the celestial sphere with respect to the celestial equator and the vernal equinox. These coordinates are called *declination* (DEC or δ) and *right ascension* (AR or α) and are equivalent to geographic latitude and longitude respectively. DEC measures the angle formed from the celestial equator towards the poles of the celestial sphere, it can be measured in degrees or minutes of arc. AR is the angle of the object to be identified from the intersection line of the ecliptic plane

with the celestial equatorial plane starting from the Aries point, and it is measured in hours, minutes and seconds.

In order to identify an object, a pair (α, δ) is presented, but it is important to note that this coordinate system is tied to the orientation of the Earth in space, which changes every 26,000 years (due to the precession of the equinoxes). . This is why the pair of coordinates is accompanied by an epoch which sets the orientation of the Earth. The standards for this time are J2000 and B1950, although currently only the first is used. In Figure 8 you can see an indicative scheme of the equatorial coordinates.

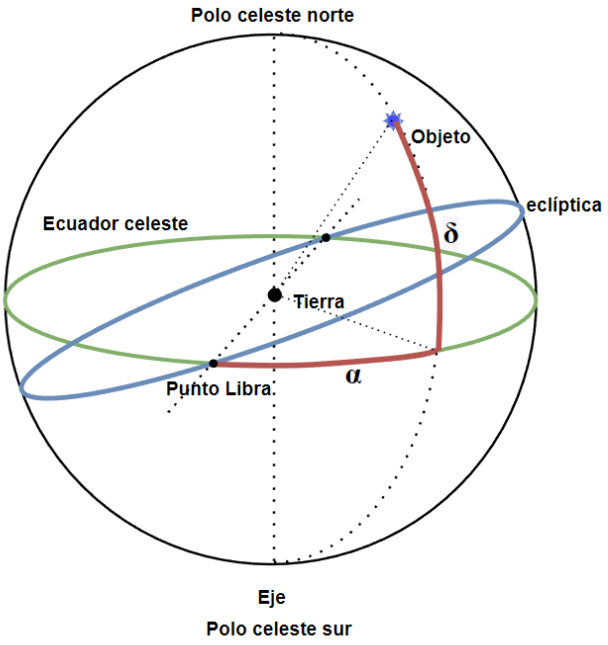


Figure 8. Illustration of the equatorial coordinate system. The observer is at the center of the sphere.

Preliminary Concepts - Theoretical Framework

Vision

Computer *Vision* is a field within Artificial Intelligence focused on the study of methods that allow computers to understand and obtain relevant information from a digital image in order to make decisions. based on it. Computer vision has received particular attention in recent years due to its wide variety of applications, ranging from robotics, where its use lies in understanding the environment where robots operate, to biology and medicine, assisting in the diagnosis of diseases. As a result of this, studies have been motivated to develop techniques or models so that a computer can carry out these tasks. In broad terms, research has been divided into different areas, where *image classification* and *object recognition*. The first covers the task of understanding the objects that make up an image in order to assign a label that identifies them. On the other hand, *object recognition* combines the classification task with the task of recognizing and locating objects within the image. Due to its vital importance in the development of this thesis, we will give a detailed analysis of one of the models created from the study of these tasks that has shown great performance.

The development of these models has its difficulties, since the way in which an image is represented in a computer does not take into account the information about the interpretation attributable to it and leaves a wide margin of correlation between its representation and its interpretation (*semantic gap*)and it is here where the models must unfold to fulfill their goal. By way of example, this complexity can be appreciated by noting that, given the representation of an object in the computer (for example, an object in an image can be represented with a set of numerical values corresponding to the pixels in the image) and a model capable of analyzing these values and classifying the image, it could change all its values due to a slight variation in its environment (such as a change in the camera angle) and make the model no longer effective. Above this, the object to be classified could be covered by another, be deformed or vary the lighting it receives and this would modify its entire representation in the computer but its semantics would be maintained. Ideally, the model should remain robust to these situations and continue to obtain the same result regardless of possible variations in the image.

innumerable Intuitively, an approach for model design could be to describe the fundamental characteristics of a class by means of rules, and that their fulfillment determines the membership of an object to it, but this approach is not very feasible in practice since we would have to writerules and would fail to generalize correctly to other classes. Consequently, alternatives that scale better have been investigated, managing to find a model development technique that, instead of being based on static rules, takes a data *-driven*. In general, this approach consists of, from an image dataset together with their respective classifications, **training** a model that is capable of learning the patterns or characteristics that make up the object, this is where *machine learning*. The detection of these features in an image should be done automatically and following a hierarchical order, that is, start by detecting low-level (edges or color spots) and group them in order to recognize increasingly higher-level features. (a nose, eyes). As an example, Figure 9 shows feature extraction in a *deep learning*. Finally, this trained model should be **evaluated** in order to measure its ability to classify new images that have not been previously seen.

Figure 9. Illustration representing the extraction of features from an image[1].

Convolutional Neural Networks

Within the area of *machine learning,* one of the most notable advances in recent times for image classification were Convolutional Neural Networks (CNN from here on), which are capable of extracting increasingly abstract features from the image. image content. To do this, an NCR does not need to pre-analyze the data to derive features such as texture and shapes, but instead learns to extract the attributes from the input image pixels.

[1] MIT 6.S191

RNCs are based on conventional Artificial Neural Networks (ANNs), which receive an input (a single vector) and transform it as it progressively traverses the layers of the network. Each layer is built by a set of neurons, each connected to all the neurons of the preceding layer. Each neuron within a layer works independently and does not share any connection or information with the others. The last layer of neurons is called the output layer of the network and when the model performs classification tasks, its output represents the membership probability of the input tensor to each class it is trying to assign.

Although it would be possible to apply ANNs to the image classification task, this approach would not scale to large images. As an example, if we have an image 16x16 pixels wide and high along with 3 color channels (red, green and blue), the ANN will have in its first hidden layer 16*16*3 = 768 weights. As much as this amount seems acceptable, if we consider an image of a more common size, such as an image of 200x200 pixels, we will have 120,000 weights, which significantly increases the computational cost of the training (note that we are only considering a single layer , but we could have several hidden layers.

With this in mind, NCRs consider the fact that their input consists of images (three-dimensional tensors) and build their architecture in a way more suitable for this. In particular, unlike ANNs, their layers have neurons organized in 3 dimensions: width, height and depth that correspond to the color channels.The neurons in these layers instead of being connected to all the other neurons of the previous layer, are connected only to a subset of them, this set will be the *receptive field* of a neuron (equivalent to the size of the filter that we will describe shortly) and from this arises the main advantage of this model. And above this, the last layer of the classifying RNCs has a 1x1xC format where C is the number of classes to detect. This is because for that stage of the architecture, the complete image is reduced to a vector of probabilities of belonging to each one of the classes. In Figure 10 the organization of neurons of a RNC can be appreciated in a simple way.

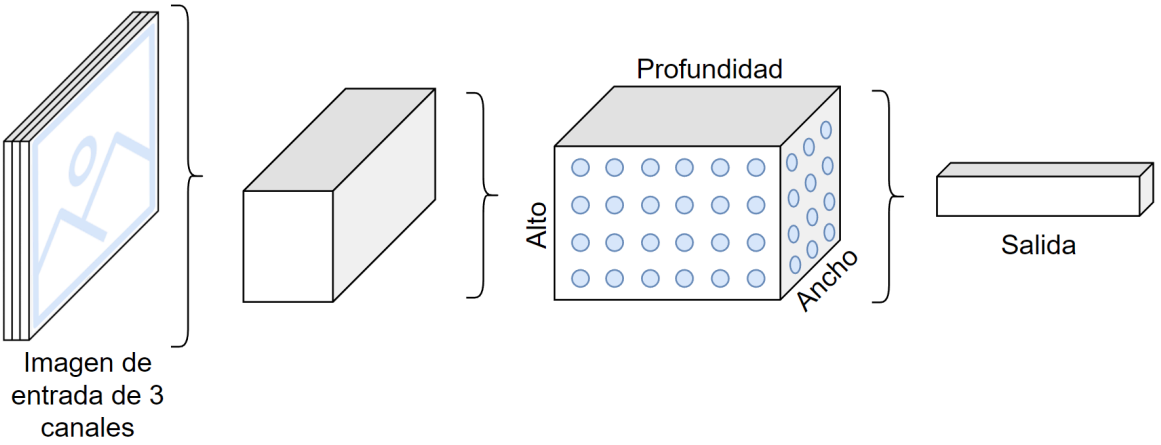


Figure 10. High-level illustration of a Convolutional Neural Network architecture where neurons are organized in a three-dimensional way.

Convolution Operation

Although RNCs have a complex architecture, they are broadly composed of three types of layers: layers that perform the **convolution**, **pooling** layers **output**. Firstly, the input tensor is entered into a convolution layer, which will capture the characteristics of it, followed by it we will find the pool layers which will highlight and compile information from the result of the previous operation. This process can be repeated multiple times within a single architecture and will progressively extract features of a higher and higher level.

In the last layers of the network we find the output layers that will generate the result of it. Next, we will describe each of these layers in greater detail, beginning with the convolution layers, which make up the essential blocks of the RNC and carry the computational cost of the network to a greater extent.

Convolution Layers

In summary, convolution layers consist of a set of filters which can be manually configured or trained. Each of them consists of a dimensional (width, height and depth) of a small size. For example, a filter in a convolution layer could have a size of 5x5x3, that is, 5 pixels wide and high and 3 channels deep. If this layer is located at the beginning of the network, we can interpret that these 3 depth channels would correspond to each other. to the 3 color channels. In the

stage equivalent to the *forward-pass* of the ANNs, the filters are shifted across the width and height of the input tensor layer and the dot product is applied between the filter values and the current portion of the input tensor traveled (we will call this portion *the receptive field*), starting from the upper left corner. As we traverse the input tensor we will produce a two-dimensional matrix or *feature map* that represents the result of the convolution operation with that filter. Generally, the NCR will train different filters that manage to capture different characteristics of the image; such as an edge, a stain or possibly part of an object (an eye or a nose) in the later layers of the network.

As we mentioned, each neuron in a layer of the NCR is connected to a unique receptive field whose dimensions must be configured. It is important to note that, in conventional convolution layers, each filter covers the entire depth of the input tensor, that is, each filter covers only a region of the width and height of the input tensor but covers the entire depth of the input tensor. From this it follows that the result of the application of convolution with a certain filter will be what is processed by a set of neurons connected to different receptive fields.

In figure 11 we can see a simple example of a neuron inside a convolutional layer. As we can see, it maintains the structure of a neuron in a conventional ANN, unlike the fact that its input is restricted to only one section of the input. In the figure $x_{0,0}$, $x_{0,m}$, $x_{n,0}$, $x_{n,m}$ represent the values of the receptive field of the neuron (with *n* and *m* being the size that it encompasses), the circle the computational unit that generates as output $h_{W,b}(x)$ (where W is the matrix of filter values), *b* a bias that may or may not be present and x the input tensor.

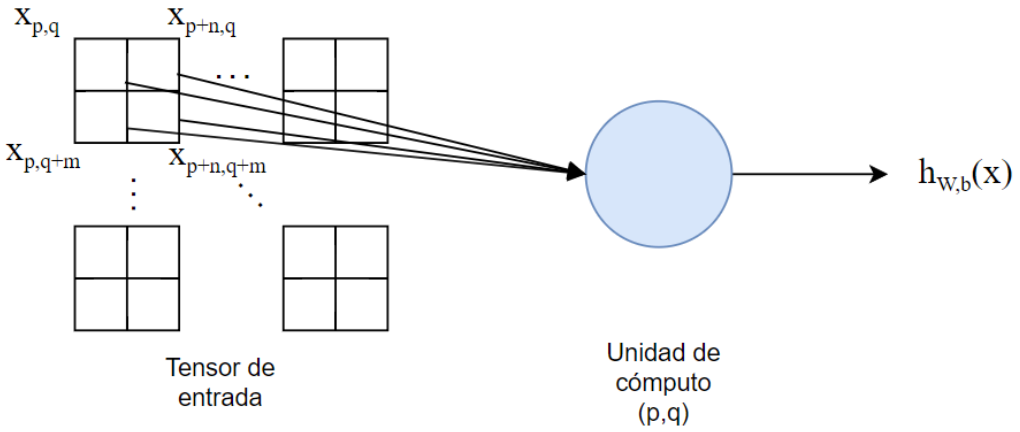


Figure 11. Example of a neuron from a convolution layer.

More formally, the computational unit generates:

$$(1) \qquad h_{W,b}(x) = f\left(\sum_{i=0}^{n} \sum_{j=0}^{m} W_{ij}\, x_{i+p,\, j+q} + b \right)$$

Where (*p,q*) is a neuron in the hidden layer generating the current window offset. As we can see, after the sum is calculated, a function *f(x)* which must be *non-linear* and can take several forms (see the available options later). In Figure 12 you can see an example of the application of the convolution operation to an image of a single color channel. In the case of an image with two or more channels, the outputs generated by $h_{W,b}(x)$ from each convolution operation in the depth dimension of the filter will be summed to produce a single feature map.

Since we have described how neurons are organized to process the input tensor, we will now go into more detail about the output produced by the convolution layers. There are three factors that influence it:

- The *depth* of the output tensor is given by the number of filters set for the layer. As we mentioned before, each convolution operation performed with a filter generates a different feature map and these will be stacked determining the depth of the output tensor of the layer.

- The displacement or *stride* of the filter by the input tensor can be modified so that it makes jumps of more than 1 place per iteration. Increasing the offset value will cause the output to decrease in width and height.
- Another influential factor is the incorporation of zeros at the edges of the image, this is convenient when you want to control the width and height of the output or more commonly to preserve the original dimensions. We will name this factor as *zero-padding*.

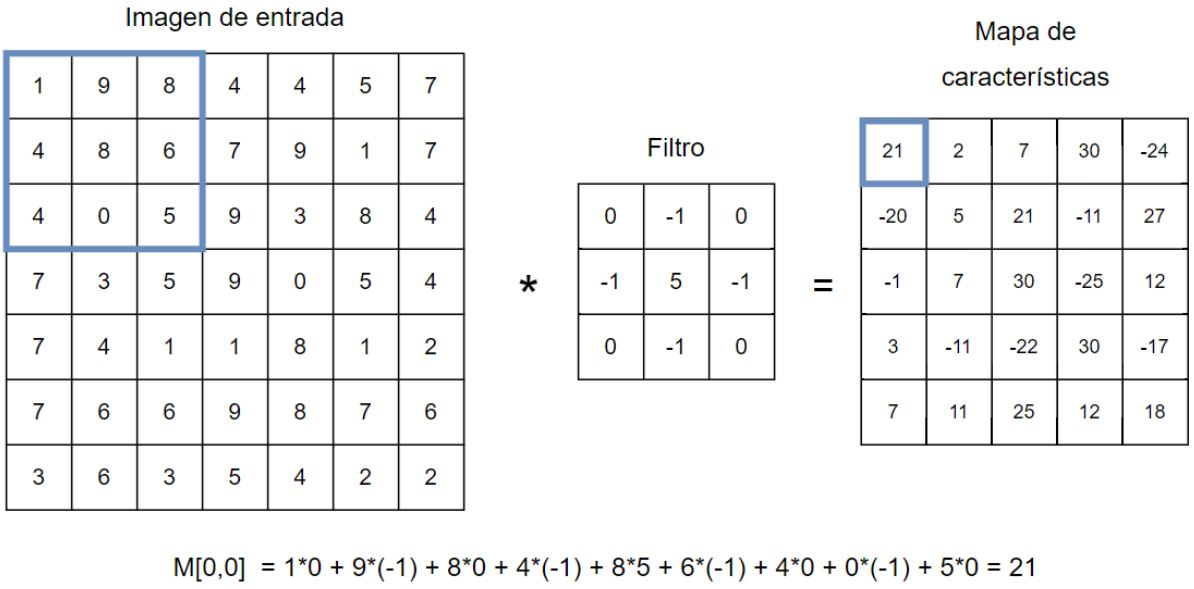


Figure 12. Example of convolution on a single channel image. It shows the current receptive field in light blue, and below the calculation to obtain M[0,0].

Finally we will be able to calculate the dimensions of the output tensor as a function of the input W, the receptive field of the neurons in the convolution layer F, the stride S, and the size of the zero-padding P. Said formula is described as

$$(W - F + 2P)/S + 1$$

and determines the size of the feature maps in the network output. For example, if W = 140 (140x140), filters of size F = 3 with offset S = 1 and zero-padding P = 0, we will have feature maps of size 138x138.

Intuitively, if in the previous example we had 12 filters, it would result in an output tensor of 138x138x12, which corresponds to 228,528 neurons with each 3x3x3 = 27 weights, resulting in 6,170,256 parameters, which would become computationally expensive to to train. However, in the NCRs the concept of *sharing parameters* in order to reduce their number in each layer and improve training speed. Este concepto surge de considerar que la detección de una característica no debería ser específicamente en una sección fija de la imagen, sino que se debería intentar detectar esa característica en toda la imagen en general. Esto lo realiza compartiendo los pesos de todas las neuronas en un mismo canal, dicho de otro modo, todas las neuronas involucradas en generar el mismo mapa de características compartirán una misma configuración de pesos. De esta manera, con el tensor de salida de 138x138x12 del ejemplo anterior, resultará en solamente 12 diferentes pesos ya que las neuronas de un filtro de 3x3 compartirán el mismo peso, resultando en 12*3*3*3 = 324 parámetros, reduciendo drásticamente su cantidad.

Capas de convolución separables por canal

A grandes términos, este tipo de capas se diferencia de las capas de convolución convencionales al realizar la operación de convolución en dos parte [3]. En primera instancia computa la convolución de manera individual en los diferentes canales (profundidad) del tensor de entrada. Esta operación se conoce como *convolución en profundidad*. Dicho de otro modo, sean W y H el alto y ancho del filtro a utilizar, luego las capas de convolución separables por canal toman un filtro de WxHx1, a diferencia de la operación de convolución habitual donde hace uso de filtros de WxHxC donde C es la cantidad de canales de la imágen.

Luego realiza una operación de *convolución píxel por píxel* en los 3 canales, es decir, toma un filtro de 1x1xC. Esta operación cumple con dos propósitos: en primer lugar logra controlar la profundidad del tensor de salida (o la cantidad de mapas de características generados) y, por otro lado, aplica una función no lineal, como veremos más adelante. Si bien ambas finalmente generan un tensor de iguales dimensiones, la gran ventaja de este tipo de capas es que reduce radicalmente la cantidad de operaciones para lograrlo. Esto se debe a que se transforma la imagen una única vez en la etapa de convolución en profundidad, y

luego se toma esta imagen para aplicar la operación de píxel por píxel separando en los resultantes canales. En la Figura 13 se puede apreciar la diferencia entre ambos tipos de capas, donde, a partir de una entrada de un tensor o imagen de tres canales de 144x144x3, se obtiene una misma salida de 140x140x5, ambas haciendo uso de 5 filtros con un campo receptivo de 5x5.

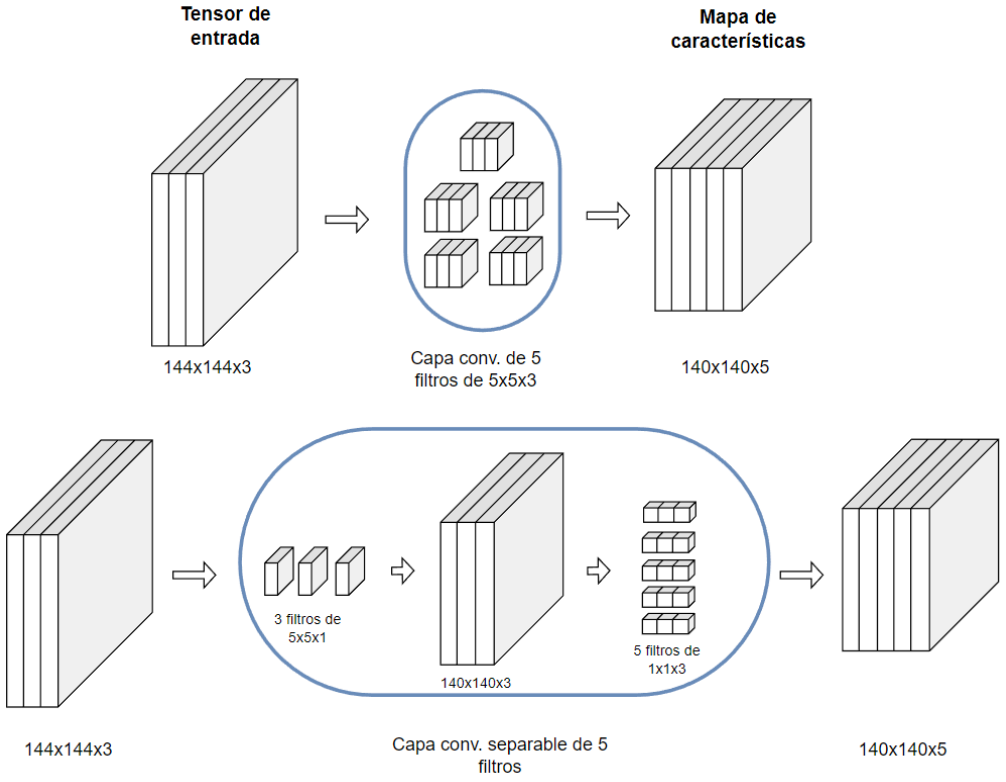


Figura 13. Ilustración representativa comparando las capas de convolución convencionales con las capas de convolución separables por canal, ambas con 5 filtros de 5x5, stride 1 y sin zero-padding.

Como mencionamos, la ventaja de las capas de convolución separables por canal es que reducen la cantidad de multiplicaciones necesarias para obtener un resultado, lo demostraremos con un ejemplo. Siguiendo con lo ilustrado en la Figura 13, supongamos un tensor de entrada de 144x144x3, filtros de 5x5, con stride 1 y sin zero-padding:

- Utilizando una capa de convolución convencional, tendríamos 5 filtros de 5x5x3 logran una salida de 144 - 5 + 1 = 140 de ancho y alto y 5 de

profundidad, lo que equivale a 140*140 iteraciones, necesitando 5*5*5*3*140*140 = 7.350.000 multiplicaciones para obtener el resultado.

- En contraste, al utilizar una capa de convolución separable, tendríamos en la operación de convolución en profundidad 3 filtros de 5x5x1 que logran un resultado intermedio de 140x140x3, lo que requiere 140*140 iteraciones resultando en 5*5*1*140*140 = 490.000 multiplicaciones. Y para la convolución píxel por píxel tendríamos 5 filtros de 1x1x3 logrando una salida de 140 de ancho y alto, lo que requiere 1*1*3*140*140 = 58.800 operaciones. Sumando ambas cantidades, obtenemos 548.800 multiplicaciones en total, lo cual evidencia la disminución de operaciones necesarias para obtener una salida de igual tamaño. Esto permite que la capa pueda procesar mayor cantidad de información en menos tiempo.

Sin embargo, este tipo de capas pueden llegar a disminuir de más la cantidad de parámetros y pueden no ser suficientes para ajustarse a los datos de entrenamiento. De todos modos, si la topología de la red utilizada es lo suficientemente extensa, la ganancia en la eficiencia compensa una pequeña posible pérdida de rendimiento.

Funciones de activación no lineales

Como mencionamos, las neuronas que componen las capas de convolución mantienen la misma estructura que las de una RNA convencional, a diferencia de que procesan solamente una porción de su entrada. Continuando con la fórmula (1), $h_{W,b}(x)$ aplica una función no lineal luego de aplicar el producto escalar entre ambas matrices, para generar en conjunto con las neuronas conectadas al mismo canal de la imagen, un mapa de características. La importancia de aplicar una función de activación no lineal reside en poder ajustar los parámetros correctamente a los datos de entrada ya que éstos generalmente siguen patrones no lineales y así incrementando la expresividad del modelo [4].

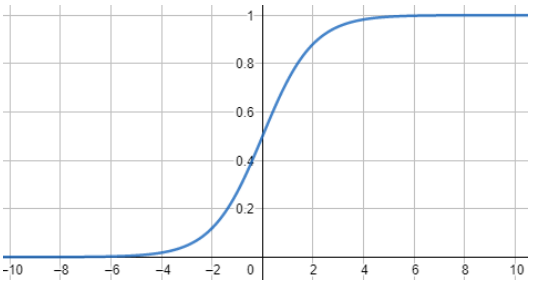
En el caso de omitir la función de activación no lineal o aplicar una función lineal y concatenar varias capas de convolución, cada capa emitirá una transformación lineal de su entrada, por lo tanto la última capa de convolución será una transformación lineal de la entrada inicial convirtiendo al modelo en un modelo de regresión lineal. Pudiendo simplemente reemplazar todas las capas de la

red por una única capa y perdiendo la capacidad de aumentar la profundidad del modelo.

A continuación describiremos algunas de las funciones de activación más comúnmente utilizadas:

- Una de las primeras funciones adoptadas en los modelos basados en RNA fue la función **Sigmoide**, con fórmula

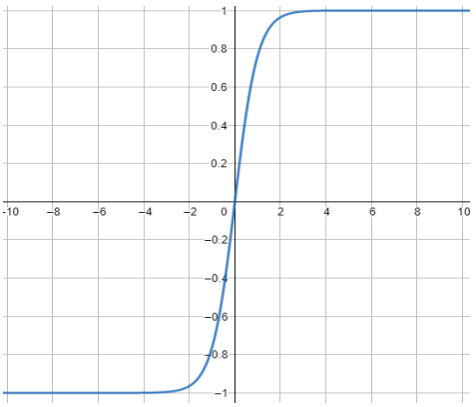
$$f(x) = \frac{1}{1 + e^{-x}}$$



  Su popularidad se vió influenciada por tener una capacidad de activación casi bipolar; desde no activarse "0" hasta activarse completamente saturando su resultado "1", siendo de utilidad en los primeros modelos donde partían de neuronas biológicas y su activación era similar a las producidas en ellas. Sin embargo, como veremos más adelante, su uso se ha vuelto escaso en las topologías más recientes y desalentado en las RNC.

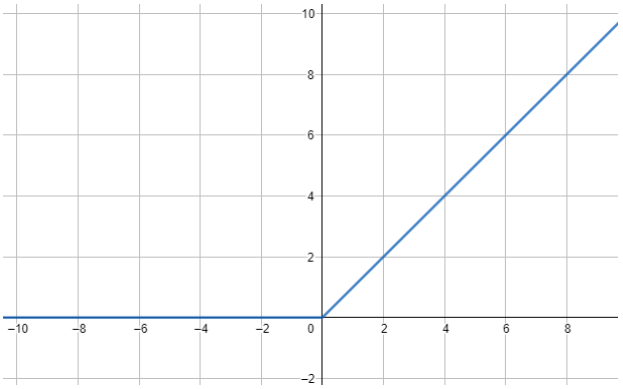
- Una de las primeras mejoras acuñadas para la función sigmoide fue **Tanh**, la cual es una función simétrica centrada en cero. Al igual que el sigmoide, su salida está acotada, pero esta vez entre -1 y 1 por lo que la hace más preferible. Nótese que Tanh es una función compuesta sobre sigmoide; $\tanh(x) = 2\sigma(2x) - 1$.

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$



- La función **rectified linear Unit (ReLU)** ha visto un incremento en su uso en los últimos años por su facilidad de cómputo. Su activación está basada en un umbral en 0 y luego la función identidad para entradas que lo superen.

$$f(x) = max(0, x)$$



- ReLU enfrenta un problema a la hora del entrenamiento del modelo, en el cual las neuronas que hacen uso de esta función pueden quedar inactivas. Para solventarlo, se ha diseñado la función **Leaky ReLU** la cual, en vez de establecer en 0 los valores negativos, añade una pequeña pendiente positiva (α) para las entradas negativas.

$$f(x) = \begin{cases} x & si \quad x > 0 \\ a \cdot x & si \quad x \leq 0 \end{cases}$$

- En las RNC clasificadoras de múltiples clases, se pretende que el modelo sea capaz de producir una probabilidad de pertenencia de la entrada a cada una de las clases, es por ello que se ha implementado la función de activación **softmax**, habitualmente encontrada en la última capa de estos modelos. Convierte valores reales en probabilidades, haciendo una generalización de la función logística y su fórmula está descrita a continuación, donde K indica la cantidad de clases a reconocer yz_x los valores del vector de entrada:

$$f(x) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Capas de pooling

Dado que la motivación de los modelos clasificadores es poder reconocer características de cada vez más alto nivel, debemos poder reducir la información para quedarnos solamente con aquellas más relevantes y así poder construir patrones de un mayor grosor. Para ello tenemos dos alternativas, por un lado podemos manipular los parámetros de la capa de convolución como el tamaño del filtro o el zero-padding para generar mapas de características de un menor tamaño, y por otro lado podemos incorporar, luego de la aplicación de la función de activación, una forma de tomar valores que representen a cada región de los mapas de características generados. Para ello se han pensado las capas de pooling, las cuales tomarán de manera eficaz las características de distintas regiones de los mapas de características. Logrando además representar la misma información de los mapas de características pero en dimensiones reducidas. Existen dos maneras mayormente utilizadas para realizar la extracción; mediante el **promedio** o el **mayor** de los valores de una región.

El modo en que se compila esta información se asemeja al recorrido de la convolución; se establece un tamaño de la región y se recorre la entrada, pero se diferencia en que las regiones capturadas no se superponen entre sí. Dicho de otro modo, podemos verlo como el desplazamiento de la operación de convolución pero con un stride del tamaño de las regiones. A modo de ejemplo, podemos ver en

la Figura 14 la aplicación de la operación de pooling con los acercamientos de **max** y **average** pooling.

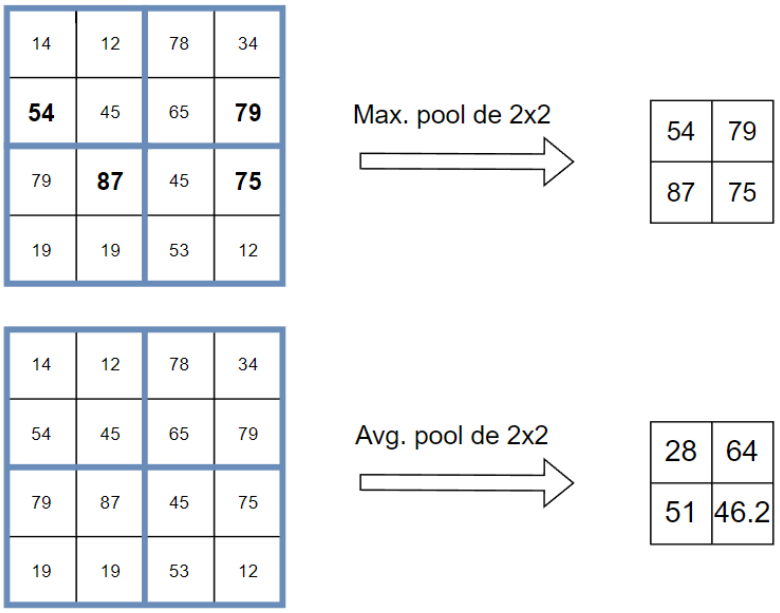


Figura 14. Ejemplo de aplicación de la operación de max y average pool de 2x2 sobre una matriz de 4x4 obteniendo una matriz de 2x2.

Cabe aclarar que la operación de pooling es aplicada individualmente por cada canal en la profundidad del tensor de entrada, o lo que es lo mismo, cada mapa de características generado por una capa de convolución y aplicado la función de activación será procesado individualmente por una capa de pooling. Por lo que la profundidad de la salida de la capa de pooling será de igual tamaño que su entrada.

Capas densas y capas de salida

Luego de procesar la información de entrada por las capas de convolución para extraer características y reducir la información mediante las capas de pooling, debemos incorporar una manera de categorizar esta información recolectada. Para ello acudiremos a las capas densas o completamente conectadas de las RNA, ya que ellas podrán aprender y determinar qué características son relevantes para qué

clase y establecer combinaciones no lineales entre ellas. Por sobre esto, el problema de clasificación múltiple requiere que el modelo produzca una distribución de probabilidad de pertenencia a las clases disponibles. Es por esto que se utiliza como función de activación la función *softmax* descrita previamente. Cabe aclarar que cada valor de la salida de la función de softmax en una neurona puede ser interpretada como la probabilidad de que la imagen pertenezca a la clase correspondiente a esa neurona, sin embargo, este valor tiende a estar descalibrado (emitiendo las clasificaciones con exceso de confianza) cuando la red se compone de múltiples capas.

Función de pérdida

Como mencionamos, las RNC deben ser entrenadas para poder ajustar sus parámetros a los datos y poder emitir clasificaciones. Al igual que las RNA, su proceso de entrenamiento puede verse dividido en dos partes: *forward-pass* y *backward-pass*. Hasta aquí hemos descrito cómo una entrada al modelo es procesada mediante diferentes operaciones dentro del mismo para producir un resultado o clasificación, es decir el forward-pass (proceso de avance). Ahora describiremos su parte complementaria de backward-pass (proceso de retroceso), donde se cuantifica el error cometido por el modelo al clasificar imágenes y se ajustan sus parámetros de manera acorde. Si bien se parte del mismo acercamiento que las RNA, enfatizaremos sobre las diferencias adoptadas para acoplar a la clasificación de imágenes.

El backward-pass consiste del proceso durante el entrenamiento del modelo en el cual se evalúa el rendimiento de la red y actualiza iterativamente los parámetros en pos de lograr un mejor rendimiento. Para cuantificar el rendimiento de la red se puede medir el error cometido en las clasificaciones emitidas. De manera intuitiva buscaremos optimizar, es decir, minimizar progresivamente el error cometido. Para ello podemos definir una *función de pérdida.* Aprovechando el uso de la función de activación softmax en la capa de salida de la red, la cual producirá un vector de probabilidades de pertenencia a las clases, se construye una función de pérdida conocida como *cross-entropy*:

$$(2) \qquad L_i = -log(\frac{e^{f_{y_i}}}{\sum_j^K e^{f_j}})$$

En la fórmula (2) denotamos con f_i como el i-ésimo valor del vector de probabilidades, y_i como el índice de la clase verdadera y K la cantidad de clases. Como podemos notar, esta función de pérdida se compone por la función softmax descrita previamente y nos podemos convencer de que logrará cuantificar el error cometido. Para tomar el error cometido de un conjunto de imágenes al entrenar el modelo (lo que suele ser más habitual), se puede tomar la media de L_i sobre su totalidad.

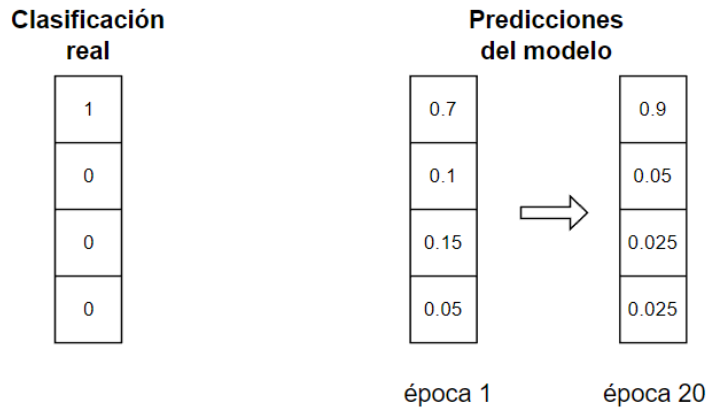


Figura 15. Ejemplo de dos salidas del mismo modelo clasificador de 4 clases en épocas distintas y comparación con la clasificación verdadera en codificación one-hot en distintas épocas de su entrenamiento.

A modo de ejemplo y tomando los valores de la Figura 15, podemos calcular la pérdida cross-entropy categórica:

$$\begin{aligned} \text{época 1} \qquad & -log_2(0.7) = 0.514 \\ \text{época 20} \qquad & -log_2(0.9) = 0.152 \end{aligned}$$

Como podemos notar, el modelo ha sido capaz de *optimizar* o reducir el error cometido al obtener una mayor probabilidad de clasificación (0.9) en la clase

verdadera en una época más tardía. El término de cross-entropy en este contexto admite que los valores de probabilidad están en codificación one-hot. También se suele utilizar la función de pérdida *sparse categorical cross-entropy* la cual admite que la representación de la clase objetivo esté codificada por un número (0 = perro, 2 = gato, etc.).

Optimizadores

De manera análoga a las RNA, luego de haber cuantificado la pérdida del modelo al realizar una calificación de una entrada (o de un conjunto) se debe proceder a modificar los parámetros de la red, de una manera tal que en cada forward-pass del proceso de entrenamiento del modelo se logre reducir el valor de la función de pérdida. Recordemos que queremos encontrar los pesos o parámetros que definen a nuestro modelo de manera que obtengan la menor media de pérdida sobre todo el conjunto del dataset, es decir la *pérdida empírica*. Más formalmente,

$$W' = argmin_W \frac{1}{n} \sum_{i=1}^{n} L(f(x^i; W), y^i)$$

El punto de esta sección es cómo poder modificar W (conjunto de todos los pesos de todas las capas de la red w_0, w_1, ...) que son inicializados de manera aleatoria, para lograr este mínimo W'. Si consideramos a la pérdida cometida por el modelo como una función de los pesos de sus capas, podemos implementar un algoritmo que busque encontrar mínimos en esta función.

Uno de los algoritmos mayormente utilizados para ello es *Stochastic Gradient Descent* (SGD). Comenzaremos con pesos aleatorios en cada capa, resultando en una pérdida determinada ya partir de allí calcularemos el gradiente de la función de pérdida:

$$\nabla L(W) = [\frac{\delta L}{\delta w_0}, \frac{\delta L}{\delta w_1}, \dots]$$

el cual, como sabemos, nos dará el vector de derivadas parciales de cada dimensión (pesos de cada capa w_0, w_1, ...) indicando el sentido en la dirección hacia la cual la pérdida crece. Por ende, tomaremos el sentido contrario (-∇L(X)) a fin de

encontrar los valores correspondientes al conjunto de pesos para los cuales cada vez se minimiza aún más su pérdida. Esto se constituye de un proceso iterativo y se realiza hasta que idealmente se alcance un mínimo global de la función de pérdida. Más detalladamente, la función de actualización de cada peso w_i se puede formalizar de la siguiente manera:

$$w_i^{n+1} = w_i^n + \Delta w_i^n$$
$$\Delta w_i = -\alpha \cdot \frac{\delta L}{\delta w_i}$$

Donde n es la iteración del proceso de convergencia y α es un factor de aprendizaje del modelo conocido como *learning rate* (α). Esta ecuación determina que el peso w_i se actualiza hacia la dirección opuesta de su derivada parcial con un factor de aprendizaje que es un número real pequeño. El learning rate tiene un rol importante en el proceso de aprendizaje, ya que si su valor es muy pequeño, el proceso terminará convergiendo en una opción óptima, pero la velocidad de convergencia será muy lenta. Por el contrario, un tamaño muy grande puede acelerar la convergencia, pero no puede garantizar que el resultado sea óptimo (o incluso no llegar a un mínimo).

Es necesario aclarar que la búsqueda del gradiente en cada iteración del algoritmo es computacionalmente costoso, sobre todo cuando se entrena en un dataset extenso. Para solventar este problema, se aproxima el gradiente de cada pérdida producida en un subconjunto o lote del entrenamiento, se calcula la media sobre ellos, y se lo utiliza para actualizar los pesos. De aquí es que obtiene un algoritmo estocástico. Podemos convencernos que esta estimación del gradiente mediante lotes es suficiente para alcanzar progresivamente el mínimo de la función de pérdida ya que no necesitamos el gradiente exacto, sino una manera estimada de progresar hacia un mínimo.

Cabe destacar que existen alternativas a este algoritmo, la búsqueda de algoritmos de optimización sigue siendo un área activa de investigación dentro de Machine Learning ya lo largo de los años se han desarrollado mejoras sobre SGD, que intentan principalmente mejorar el proceso de convergencia. Entre ellos se

encuentran RMSProp [5], ADAGRAD [6] y en refinación de estos dos, surge Adam [7], el cual está siendo comúnmente utilizado debido a su facilidad de cómputo y rápida convergencia. A grandes términos, Adam propone un método de optimización estocástica que solo requiere el cómputo del gradiente de primer orden de la función de pérdida, a diferencia de otros acercamientos actuales. Por otro lado, establece learning rates individuales para diferentes parámetros de la red o pesos de distintas capas y éstos no son estáticos, sino que varían según el valor del gradiente de la pérdida actual. En la Figura 16 se puede ver una comparación de la velocidad de convergencia hacia un coste mínimo en el entrenamiento de un modelo RNC sobre el conjunto de imágenes MNIST en 200 iteraciones, donde se destaca el rendimiento del algoritmo Adam.

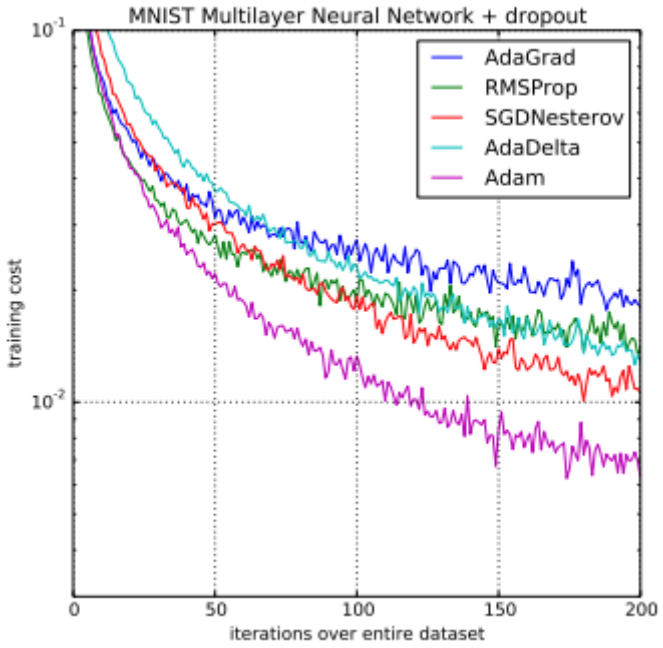


Figura 16. Comparación de diversos algoritmos de optimización. Figura tomada de [7].

Backpropagation

Una vez calculado el error cometido por el modelo al clasificar y obtenida la variación de pesos, debemos realizar la transformación de éstos para la próxima iteración del forward-pass. Esta transformación de los parámetros del modelo se denomina *backpropagation* y constituye la parte final del proceso de entrenamiento del mismo. Como podemos notar, el error cometido por las

neuronas en la capa de salida puede ser calculado directamente mediante la función de pérdida, la cual toma el valor producido y el valor real, pero esto no es el caso de las capas ocultas ya que en ningún momento conocemos el valor que producen.

Para explicar este procedimiento, tomaremos como ejemplo una neurona como en la figura 17 conectada a una región I (campo receptivo) del mapa de características de entrada y con pesos (filtro) W. De manera resumida, en el forward-pass, esta neurona computa la convolución IxW y genera como salida el valor z. Durante el backward-pass o backpropagation, una vez calculado el gradiente de la pérdida para esta configuración de pesos de la red, debemos conocer cómo actualizar los pesos W de manera acorde al algoritmo de optimización y propagar el error hacia las neuronas de la capa previa que han producido la región I. A modo de simplificar la explicación tomaremos como ejemplo la neurona relacionada a la primera iteración de la operación de convolución, es decir, el campo receptivo en la esquina superior izquierda.

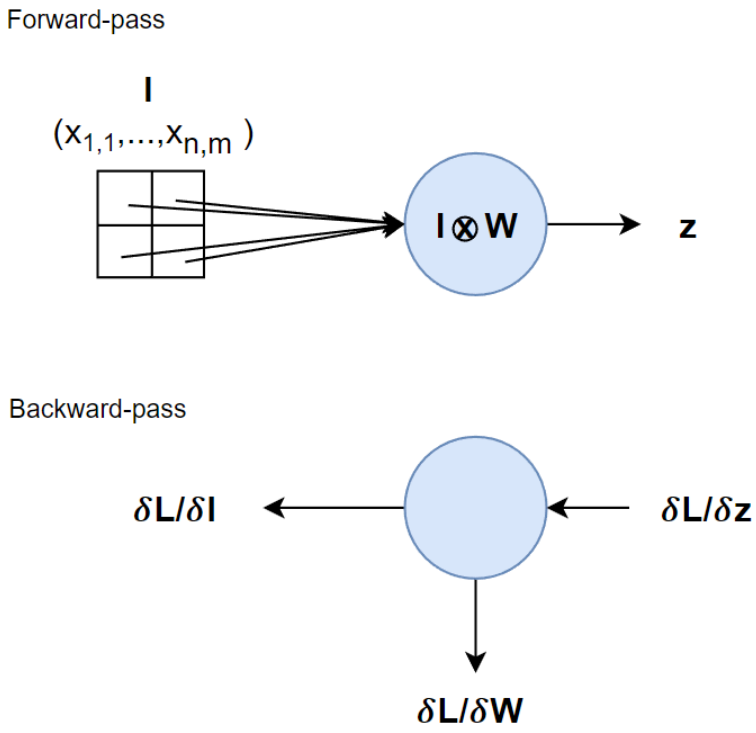


Figura 17. Ejemplo de una neurona en backpropagation.

En la figura, como I es parte de la salida de la capa previa, $\delta L/\delta I$ se convertirá en el gradiente de pérdida para su correspondiente neuronas. Por otro lado, $\delta L/\delta W$ será utilizado para calcular la actualización de los pesos del filtro usando el algoritmo de optimización elegido. Dado que se conoce solamente el valor de $\delta L/\delta z$ y los gradientes locales con respecto a la salida de la neurona sobre la entrada y el filtro; $\delta z/\delta I$ y $\delta z/\delta W$ respectivamente, podemos aplicar la regla de la cadena para conocer $\delta L/\delta W$ y $\delta L/\delta I$.

$$\frac{\delta L}{\delta W} = \frac{\delta L}{\delta z} \cdot \frac{\delta z}{\delta W}$$
$$\frac{\delta L}{\delta I} = \frac{\delta L}{\delta z} \cdot \frac{\delta z}{\delta I}$$

Con este procedimiento, el cálculo del gradiente local $\delta z/\delta W$ se puede realizar de la siguiente manera:

dado que

$$z_{1,1} = I_{1,1} * W_{1,1} + ... * I_{n,n} * W_{n,n}$$

con n siendo el tamaño del filtro, la derivada parcial de $z_{1,1}$ con respecto a $W_{1,1}$:

$$\delta z_{1,1}/\delta W_{1,1} = I_{1,1}$$

y de manera análoga para los gradientes de cada valor de la matriz de pesos. Luego podemos usar la fórmula para calcular la derivada parcial de una matriz respecto a otra matriz haciendo uso de la regla de la cadena nuevamente:

$$\frac{\delta L}{\delta W_i} = \sum_{k=1}^{n} \frac{\delta L}{\delta z_k} \cdot \frac{\delta z_k}{\delta W_i}$$

lo que resulta en

$$\delta L/\delta W_{1,1} = \delta L/\delta z_{1,1} * \delta z_{1,1}/\delta W_{1,1} * ... * \delta L/\delta z_{n,n} * \delta z_{n,n}/\delta W_{1,1}$$

Sustituyendo con los gradientes de cada valor de la matriz de peso se convierte en

$$\delta L/\delta W_{1,1} = \delta L/\delta z_{1,1} * I_{1,1} * ... * \delta L/\delta z_{n,n} * I_{n,n}$$

Y de manera análoga a los demás elementos de la matriz filtro W. Lo que en definitiva puede verse como una convolución entre el vector de entrada y el gradiente de la pérdida de salida δL/δz como en la Figura 18.

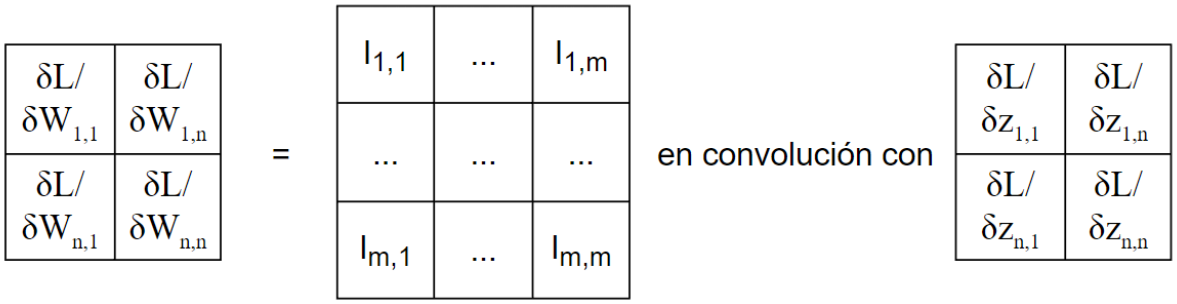


Figura 18. Vista del cálculo de gradiente local como una convolución de la entrada de la neurona con el gradiente de la pérdida de salida de la neurona. m es el tamaño de la entrada de la neurona I yn el tamaño de la matriz filtro W. Ambos restringidos a ratio 1:1 de ancho y alto para simplificar la explicación.

Por otro lado el cálculo de δL/δI:

De manera similar, como

$$z_{1,1} = I_{1,1} * W_{1,1} + ... * I_{n,n} * W_{n,n}$$

y de igual modo para los demás elementos de z, entonces

$$\delta z_{m,m}/\delta I_{n,n} = W_{n,n}$$

Tomando esta derivada parcial y usando la fórmula de cálculo de la derivada parcial de una matriz respecto a otra matriz con la regla de la cadena descrita previamente, podemos obtener la derivada con respecto a L de un elemento de I, por ejemplo, la derivada de L respecto al primer elemento de la matriz de entrada resulta:

$$\delta L/\delta I_{1,1} = \delta L/\delta z_{1,1} * W_{1,1}$$

$$\delta L/\delta I_{1,2} = \delta L/\delta z_{1,1} * W_{1,1} + \delta L/\delta z_{1,2} * W_{1,1}$$

$$\delta L/\delta I_{1,3} = \delta L/\delta z_{1,2} * W_{1,2}$$

Y de manera similar a los demás elementos hasta derivada parcial $\delta L/\delta I_{n,n}$

Esto muestra que $\delta L/\delta I$ puede verse como una convolución entre dos matrices: la matriz filtro W rotada 180 grados (primero rotada verticalmente y luego horizontalmente) y la matriz gradiente de la pérdida $\delta L/\delta z$. Pero es necesario ver a la operación de convolución comenzando de abajo hacia arriba y de derecha a izquierda y con zero-padding de n - 1.

De esta manera y con ambas derivadas parciales de la pérdida en relación a el filtro W y entrada I, podemos actualizar el filtro con el algoritmo de optimización adoptado y propagar el gradiente de la pérdida hacia las neuronas de la capa previa.

Sin entrar en detalle queremos además destacar el rol de capas de pool en el proceso de backpropagation. Por ejemplo, donde se realiza max-pool, el error propagado va a ser hacia la neurona encargada de producir el mayor valor de la región.

Las funciones de activación influyen en el error propagado hacia atrás, es por eso que mencionaremos dos funciones de activación y su influencia en

propagación. En el caso de haber utilizado la función sigmoide ($f(z) = \sigma(z)$), el error propagado va a verse influenciado por la derivada parcial de ella respecto a la salida de la neurona:

$$\delta f/\delta z = \sigma(z)(1-\sigma(z))$$

Y para la función ReLU ($f(z) = \max(0, z)$), ampliamente utilizada en las RNC:

$$\delta f/\delta z = 1(x>0)$$

Esta derivada surge de que ReLU toma la forma de la función de identidad en valores positivos con derivada 1 y 0 para valores negativos.

Para finalizar, como mencionamos, las neuronas encargadas de un canal del tensor de entrada comparten los mismos valores de la matriz filtro. Es por esto que cada neurona de una capa va a calcular el gradiente para cada uno de sus pesos pero estos gradientes van a ser utilizados para actualizar los pesos de las demás neuronas con campo receptivo en el mismo canal.

Overfitting

Cuando un modelo se entrena, puede ocurrir que las características reconocidas por el modelo no logren generalizar correctamente a una población mayor de datos. Por ejemplo, el modelo podría ser capaz de identificar características y clasificar imágenes del conjunto de entrenamiento (recordemos que entrenar estos modelos requiere separar el conjunto en secciones entrenamiento, validación y prueba) pero al medir el rendimiento del modelo con el conjunto de prueba no es capaz de extrapolar la detección de estas características en imágenes no vistas previamente. Cuando sucede este comportamiento, se puede decir que el modelo se ajustó de más a los datos de entrenamiento o hizo *overfitting*. El principal síntoma de este problema es la

diferencia de rendimiento del modelo entre los datos de entrenamiento y los datos de validación. Se puede reducir el overfitting aplicando técnicas que mejoren la generalización en datos no conocidos:

- Incorporar más datos al conjunto de entrenamiento. Esta aproximación resulta la más directa y lograríamos introducir una mayor variación de los datos para que el modelo pueda extrapolar mejor su capacidad. Sin embargo, no siempre contaremos con una mayor cantidad de datos, con lo cual deberíamos buscar una alternativa.

- Hacer uso de una técnica de *data augmentation*. Ésta consiste de crear nuevos datos a partir de existentes aplicando variaciones como rotación aleatoria de las imágenes, zoom, desplazar la imágen una cierta cantidad de píxeles, aplicar un filtro de color, entre otras. Cabe destacar que esta técnica únicamente se aplicaría al conjunto de datos de entrenamiento.

- Otro acercamiento es hacer uso de la técnica de *Dropout* de neuronas. La cual consiste en no permitir que ciertas neuronas se activen en determinadas capas. Es decir, impide que neuronas contribuyan a la información de entrada de la capa siguiente. Sin hacer uso de esta técnica, el primer conjunto de datos con el cual se entrene el modelo tendrá una mayor influencia en los parámetros del modelo y evitará aprender características que solo aparecen en datos que se ingresarán más tarde al entrenamiento. Esta técnica suele aplicarse como una capa intermedia entre otro tipo de capas.

- Aplicar normalización de los datos. Este acercamiento se ha convertido en un estándar en los últimos años y toma la idea del preprocesamiento de los datos. Dentro de las RNA y RNC es conocido como *batch normalization* y consiste en llevar los valores de la salida de las capas a una escala en común a lo largo de toda la red.

Métricas

Luego de haber entrenado un modelo, es necesario poder evaluarlo para cuantificar su rendimiento. Esta evaluación debe hacerse utilizando un conjunto

de imágenes que el modelo no haya visitado previamente con el propósito de que ésta sea objetiva.

Cuando el modelo procese los datos de prueba y asigne una clasificación a cada uno, podremos construir una *matriz de confusión*. Ésta nos ayudará a determinar si el modelo está discriminando correctamente entre los tipos de clases. Si tuviéramos K clases, consistirá de una matriz de doble entrada de KxK, cada fila indicará la clase real y cada columna la clase asignada por el modelo. En la diagonal se verá reflejado la cantidad de datos bien clasificados por el modelo y por fuera de ella los que no. Así podremos conocer de manera más detallada el rendimiento del modelo y tomar acciones sobre ello, como por ejemplo sumar mayor cantidad de ejemplos a las clases donde el modelo no logra un buen desempeño.

A partir de ella, además, podremos calcular las siguientes medidas por clase:

- *True Positives*: cantidad de casos acertados, donde el modelo clasificó a una clase correctamente.
- *True Negatives*: la cantidad de datos que han sido clasificados correctamente como no pertenecientes a la clase.
- *False Positives*: cuantas veces el modelo clasificó como de esa clase cuando en realidad no lo era.
- *False Negatives*: la cantidad de veces que el modelo consideró que un ejemplo no era de la clase pero en realidad sí.

Tomando estas medidas surgen diferentes métricas ampliamente adoptadas para medir el rendimiento de un modelo en el área de *Machine Learning* y brindan un mayor detalle sobre el comportamiento del modelo. Una de las métricas más utilizadas es *accuracy*, la cual mide cuántos ejemplos han sido correctamente clasificados del total.

$$\text{Accuracy} = \frac{T.Pos. + T.Neg}{T.Pos + T.Neg + F.Pos. + F.Neg}$$

Cabe destacar que esta medida puede resultar engañosa, sobre todo cuando los datos están desbalanceados. Por ejemplo, si consideremos un dataset que consta de un 95% de datos para la Clase A y 5% para la clase B, si el modelo predice a todos los datos como de Clase A, el modelo tendrá 95% de accuracy, pero no habrá clasificado correctamente ningún dato de la Clase B. Esto presenta un problema cuando reconocer los datos de Clase B es realmente importante, ya que la métrica fallará en indicarnos el rendimiento del modelo para esta clase y podremos pensar que el modelo está desempeñándose bien.

Por otro lado contamos con la métrica *precisión* la cual es calculada como la relación entre la cantidad de ejemplos positivos correctamente clasificados y la cantidad total de ejemplos clasificados como pertenecientes a la clase.

$$\text{Precisión} = \frac{T.Pos}{T.Pos + F.Pos}$$

Esto implica que cuando el modelo clasifique erróneamente varios ejemplos como pertenecientes a una clase, la precisión se verá reducida. Es por esto que la precisión de un modelo nos ayuda a conocer qué tan acertado es un modelo cuando determina a un ejemplo como perteneciente a una clase.

Otra métrica comúnmente empleada es el *Recall*, la cual mide la relación entre la cantidad de ejemplos clasificados como pertenecientes a una clase con el total de ejemplos de esa clase en el dataset, cuanto mayor sea el valor del recall, más ejemplos positivos habrán sido los detectados por el modelo.

$$\text{Recall} = \frac{T.Pos}{T.Pos + F.Neg}$$

Cuando el valor del recall sea alto, significa que el modelo es capaz de clasificar la mayor parte de los ejemplos de una cierta clase correctamente, entonces se podrá confiar en el modelo para clasificar los ejemplos de una clase como tal.

Por último, combinando las dos últimas métricas descritas, surge F1-score. Esta métrica permite determinar qué tan preciso es el modelo (cuantas instancias clasifica correctamente), así también qué tan robusto es (que no clasifique erróneamente una gran parte de ejemplos). Si tenemos una gran precisión del modelo pero un bajo valor de recall, significa que el modelo solo asigna una clasificación de una clase a un ejemplo que verdaderamente lo es, pero falla en reconocer a todas las instancias de tal clase. F1-score intenta establecer un balance entre precisión y recall y se define de la siguiente manera:

$$\text{f1-score} = 2 * \frac{1}{\frac{1}{\text{precisión}} + \frac{1}{\text{recall}}}$$

Acercamientos actuales

A lo largo de los años se han implementado diversos acercamientos para el análisis no solo morfológico de galaxias, sino también de detección de objetos astronómicos superpuestos, detección de características particulares de galaxias, entre otros. En esta sección haremos un breve recorrido sobre algunos de los acercamientos desarrollados en estos tópicos y veremos diferencias y oportunidades de mejora que se presentan.

En primer lugar, nos encontramos con [8], en este paper buscan clasificar morfológicamente a galaxias en imágenes, lo cual presenta un acercamiento mediante RNC similar a lo propuesto en esta tesis, pero se basan sobre una imágen estática de la región. En este proyecto no solo se podrá obtener una imágen del área que se desea consultar dinámicamente, sino que además se propone obtener una imagen de mayor resolución de cada región de interés, lo cual beneficia la capacidad de clasificación. Además mencionan el uso de de Mask RCNN (construida sobre Faster-RCNN), si bien ese acercamiento muestra resultados destacables, proponemos generar un método sencillo alternativo de extracción de regiones de interés, en donde podremos detectarlos mediante un análisis específico de la imagen e implementar nuestros propios filtros.

Como mencionamos en la sección <SECCIÓN>, se han desarrollado técnicas de clasificación no mediante modelos dentro del área de Machine Learning sino que están basados en la ciencia ciudadana, aplicando métodos de *crowdsourcing*, como lo ha hecho Galaxy Zoo[2]. Si bien ha demostrado el potencial de este acercamiento logrando que las respuestas convergen a un resultado en común gracias a la participación de miles de personas, la cantidad de datos recolectados crece rápidamente y deberíamos optar por acercamientos que aceleren el proceso de clasificación y que no se basen en aumentar el paralelismo o cantidad de participantes.

De todas maneras, en este proyecto hemos tomado aspectos de estos dos estudios, como lo son la forma de procesamiento de las Faster-RCNN basándose en la propuesta de regiones de interés y descarte a partir de la clasificación obtenida. Y por otro lado, nos hemos basado en las características detectadas por

[2] https://arxiv.org/pdf/2102.08414.pdf

los participantes del proyecto de Galaxy Zoo DECaLS para conformar los tipos de galaxias que clasificaremos, al igual que tomaremos las imágenes presentadas a cada uno de ellos en el proyecto para entrenar la RNC.

Proceso

En esta sección daremos una introducción al flujo propuesto en esta tesis a fin de reconocer y clasificar las galaxias que se encuentren en un área específica del cielo. En la Figura 19 se puede ver ilustrado el proceso propuesto, en el cual en cada etapa del mismo se procesa y analiza la información obtenida de la etapa previa para luego producir un resultado de utilidad para la subsiguiente. A modo de facilitar su comprensibilidad, describimos el proceso siguiendo el orden de su ejecución.

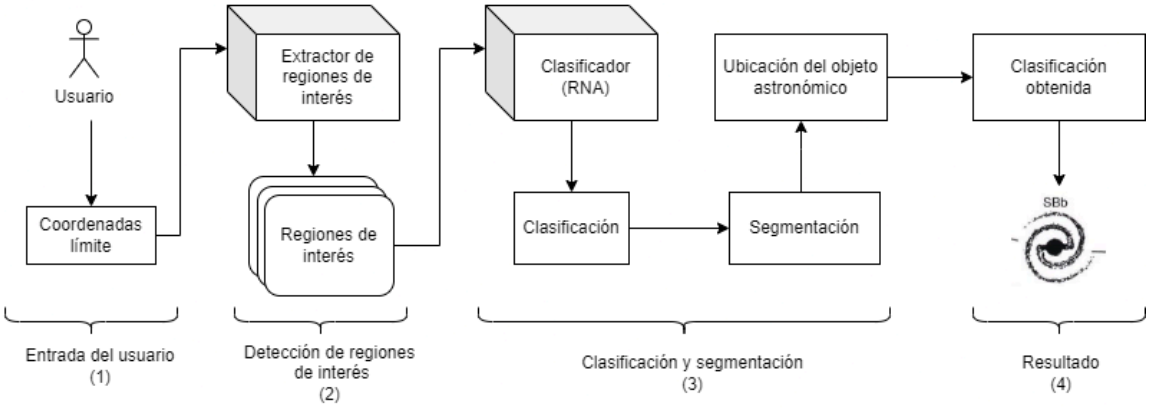


Figura 19. Proceso de obtención de objetos astronómicos dentro de un área específica del cielo.

En este documento comenzaremos explicando el proceso de selección de área del cielo en la cual el usuario tiene interés de conocer la forma de las galaxias que allí se encuentran. Para esto el usuario ingresará dos coordenadas en el sistema de coordenadas ecuatoriales que delimitan la región. Se corresponde a la sección (1) en la Figura 19.

Una vez que el usuario haya ingresado las coordenadas, haremos uso de una herramienta de acceso público provista por el observatorio encargado de llevar a cabo la Sloan Digital Sky Survey (SSDS)[3]. Con esta herramienta, podremos obtener una imagen de la zona delimitada que contendrá objetos astronómicos ya partir de allí comenzará la fase de clasificación. Dado que la imágen no solamente contendrá galaxias sino cualquier tipo de objeto astronómico, surgen las siguientes preguntas; ¿Cómo podremos diferenciar las galaxias? ¿Cómo haremos para saber

[3] https://www.sdss.org

dónde se ubican? ¿Cómo sabemos qué tamaño ocupan en la imagen? Para responder estas preguntas, debemos hallar algún método que nos indique en las partes de la imagen que puedan llegar a tener una galaxia, sección (2) de la Figura 19. El proceso de captura de estas partes es conocido en el análisis de imágenes y es denominado *extracción de regiones de interés*. Cabe destacar que estas regiones finalmente pueden o no contener una galaxia, pero debemos pensar al método que las obtiene de modo tal que nos dé una buena aproximación inicial.

Una vez obtenidas estas regiones de interés, procederemos a analizar cada una de ellas para conocer si el objeto que la compone es una galaxia, estrella, algún otro objeto celeste o ningún objeto en particular, correspondiente a la sección (3) de la Figura 19. Para ello, vamos a desarrollar y entrenar una RNC utilizando imágenes similares a las regiones de interés que pueden surgir a partir de la consulta de un usuario. El objetivo es obtener un modelo de clasificación fiable que pueda recibir imágenes de distintas zonas del cielo, y brindar una clasificación morfológica para determinar si esa zona contiene una galaxia y, en caso afirmativo, el tipo de la misma, con un grado de confianza.

Una vez realizado el proceso de clasificación para cada región de interés, sabremos cuáles efectivamente tomar y cuáles rechazar según el resultado de cada una. Con aquellas que hemos tomado, podremos entonces construir una nueva imagen de la región del cielo con cada galaxia identificada y marcada (*segmentación semántica*) mediante un rectángulo y su respectiva clasificación morfológica, sección (4) de la Figura 19. Esta imagen, que será el resultado final del flujo, será presentada al usuario quien podrá apreciar visualmente las distintas galaxias presentes en la zona del cielo consultada, cumpliendo con el objetivo propuesto. No está demás decir que el proceso contemplará aquellos casos borde, donde por ejemplo no pueda ser identificada ninguna región de interés o ningún objeto clasificado pueda ser descrito como una galaxia con certeza.

En las siguientes secciones explicaremos detalladamente cada una de las etapas descritas en los párrafos anteriores. Presentaremos las soluciones implementadas para resolver cada etapa, comentando además las investigaciones realizadas, los problemas encontrados y las decisiones adoptadas para obtener un sistema funcional.

Etapa 1 - Selección del cielo y obtención de Regiones de Interés

Como vimos de manera introductoria en la sección precedente, la primera funcionalidad provista por el flujo es permitir al usuario ingresar coordenadas delimitadoras de una región en el cielo celeste en la cual el usuario tenga curiosidad de averiguar qué tipo de galaxias allí se encuentran. En esta sección daremos una explicación detallada de la primera etapa del proceso realizado en nuestro programa para poder responder esta pregunta. Comenzando por la obtención de una imagen de la zona, siguiendo por el análisis de la misma para obtener regiones de la imagen que potencialmente contengan una galaxia y finalmente filtrando cada una de ellas según cumplan ciertas condiciones.

Ingreso de la región del cielo

Cuando el usuario ingresa al programa, la pantalla inicial le permite ingresar dos coordenadas, utilizando el sistema de coordenadas ecuatoriales, para delimitar la sección del cielo que desea explorar. Recordemos que el sistema de coordenadas ecuatoriales consiste en la ascensión recta (AR), la cual puede pensarse como el equivalente astronómico a la longitud y la declinación (DEC) equivalente astronómico de la latitud. Ambos ángulos son ingresados en grados (°) en representación punto flotante.

El primer paso realizado por el programa es la obtención de una imagen de la región, esto es facilitado por una herramienta de acceso público disponible gracias al observatorio encargado de la SDSS[4]. Hemos optado por utilizar esta herramienta debido a la gran disponibilidad de imágenes que contiene, la amplia cobertura del cielo celeste realizada (14.555 grados cuadrados; para referencia, la luna llena cubre aproximadamente 0,2 grados cuadrados), la facilidad de acceso a las imágenes y su gratuidad. Cuando el usuario ingresa la región de cielo, puede hacerlo libremente, pero resulta conveniente añadir ciertas restricciones a fin de cumplir con las limitaciones de la SDSS y mantener el uso práctico del programa. En primer lugar, las observaciones de la SDSS no han cubierto el cielo celeste en su totalidad (ver Figura 20), debiendo evitar que el área seleccionada caiga en un

[4] http://skyserver.sdss.org/dr16/en/help/docs/api.aspx

área de la cual no podremos obtener imágenes. En segundo lugar, consideramos necesario evitar que el usuario seleccione un área extensa del cielo, debido a que su análisis puede ser computacionalmente costoso. Vale aclarar que esta restricción sobre el cielo celeste no está explícitamente implementada en nuestro programa ya que el cubrimiento de la observación no está claramente delimitado, pero igualmente sabremos si la región es aceptada y reside dentro de los límites al utilizar la herramienta de la SDSS.

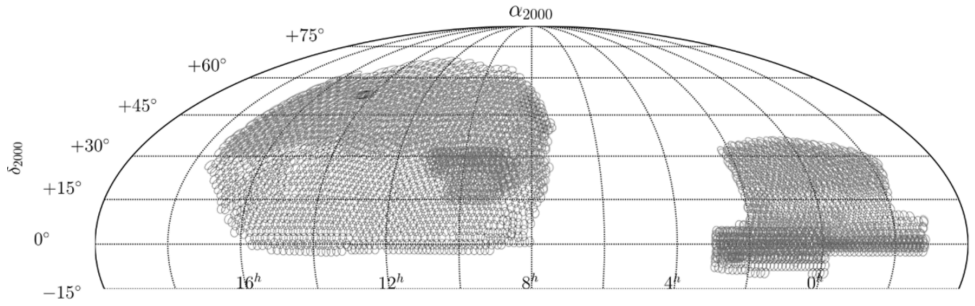


Figura 20. Área observada por la SDSS[5] hasta su lanzamiento número 17. El gráfico está en coordenadas ecuatoriales y centrado en ascensión recta de 8H.

A modo de obtener una imagen mediante esta herramienta, requerimos de tres elementos:

- Una **escala**, cuyo valor indica el acercamiento que tendrá la imagen. Específicamente, el valor determina la cantidad de segundos de arco por píxel, es decir, si tomamos una escala de 0.4, 1 píxel equivaldrá a 0.4 segundos de arco.

- El **ancho** de la imagen en píxeles, podría interpretarse como la longitud en el "eje" de coordenadas de ascensión recta en el sistema de coordenadas ecuatoriales.

- El **alto** de la imagen en píxeles, interpretado como la longitud en el "eje" de coordenadas de la Declinación en el sistema de coordenadas ecuatoriales.

Es importante notar que la escala elegida condiciona directamente al ancho y alto en píxeles que se deberá tomar. Si usamos una escala de 0.5, 50 píxeles equivaldrá a 25 segundos de arco, mientras que si la escala es de 0.1 para obtener 25 segundos de arco necesitaríamos 250 píxeles. Cabe destacar que buscaremos

[5] https://www.sdss.org/dr17/scope/

obtener una imagen con la mayor resolución posible admitida por la herramienta, la cual establece como límite superior 2048px por lado, lo cual inicialmente condiciona la escala tomada para enmarcar la imagen. A continuación daremos un ejemplo a fin de aclarar aún más el método de obtención de estos requerimientos.

Ejemplo:

El usuario desea estudiar la zona del Quinteto de Stephan (mostrado en la Figura 21) ingresando las coordenadas que lo delimitan *(α 339.06204, δ 33.99082)* y *(α 338.94632, δ 33.92960)* en grados respectivamente. Al admitir sólo áreas pequeñas, nos permitiremos despreciar el ángulo formado por la esfera entre los puntos y simplemente obtendremos el punto central como el promedio de ambas coordenadas. Quedando como punto central *(α 339.00418, δ 33.96021)*. Ahora resta obtener la escala (segundos de arco por píxel), el ancho y alto en píxeles de la imagen. Para ello mediremos la longitud de los lados del rectángulo en segundos de arco y, a fin de obtener la mayor resolución posible admitida por la herramienta de la SDSS, le asignaremos el valor máximo en píxeles (2048) al lado más largo. En este ejemplo, como los lados en AR tienen una longitud de 345" y los lados en DEC de 220", al lado extendido a lo largo de la AR le asignaremos 2048 píxeles de longitud, la escala la deducimos directamente con la longitud del lado más largo sobre 2048 píxeles, lo cual resulta en 0.168 segundos de arco por píxel y el valor en pixeles del lado más pequeño simplemente tomando su longitud sobre la escala, en este caso 1306 píxeles.

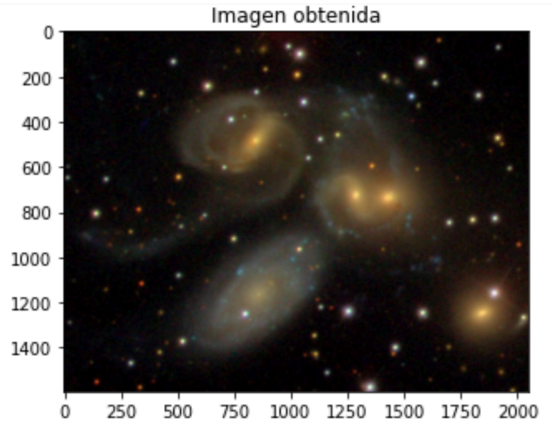


Figura 21. Imagen del Quinteto de Stephan obtenida mediante la herramienta de la SDSS a partir de la coordenada central, la escala, su ancho y alto. Ambos ejes muestran la longitud del lado en píxeles.

Extracción de Regiones de interés

El siguiente paso que haremos a fin de poder identificar las galaxias dentro de la imágen será extraer sus *regiones de interés*. A cada una de estas regiones de interés deberíamos poder ubicarlas dentro de la imagen y conocer sus dimensiones. En esta sección explicaremos cómo es nuestro acercamiento a la obtención de las mismas y, sabiendo que potencialmente no todas contienen una galaxia, cómo es el método de filtrado de las mismas para quedarnos solamente con aquellas relevantes a este estudio.

Nota: Recordemos que en el contexto de este documento, una *región de interés* refiere al área de la imagen a la cual debemos prestarle particular atención ya que podría contener una galaxia que podríamos clasificar morfológicamente.

El campo de la visión artificial ha ganado gran inercia en los últimos años debido a su utilidad en diversas prácticas de uso cotidiano. Dentro del mismo existe un área de reconocimiento de objetos, que viene a estudiar justamente cómo reconocer las partes que componen a una imágen. Diversas áreas sacan provecho de ella, como los vehículos autónomos al interpretar su entorno para desenvolverse y los sistemas de vigilancia en el reconocimiento de personas y sus posturas. Gracias a estos usos, como tanto otros, se ha profundizado su estudio a fin de optimizar su rendimiento y realizar su tarea con mayor destreza.

A raíz del estudio de reconocimiento de objetos, han sido desarrollados diversos modelos y en nuestra investigación hemos indagado sobre la manera en la cual obtienen las regiones de interés, lo cual recordemos, constituye la primera etapa para el reconocimiento de objetos en una imágen. Entre estos modelos investigados se encuentran las R-RNC [9], las cuales aplican un algoritmo de propuesta de regiones de interés basado en generar de manera arbitraria una gran cantidad de segmentaciones dentro de la imagen y luego unirlas según su similitud. Luego, cada una de ellas es clasificada por una RNC convencional que dará el veredicto sobre si contiene o no un objeto relevante en su aplicación. Por otro lado, existe un modelo ampliamente reconocido por su rendimiento, el cual es nombrado Fast R-RNC [10] y busca mejorar la velocidad con la cual se generan las propuestas de regiones de interés en las R-RNC, aprovechando el mapa de características construido por las primeras capas de convolución de la RNC.

Podemos ver el flujo de procesamiento del método Fast R-RNC de manera esquemática en la Figura 22.

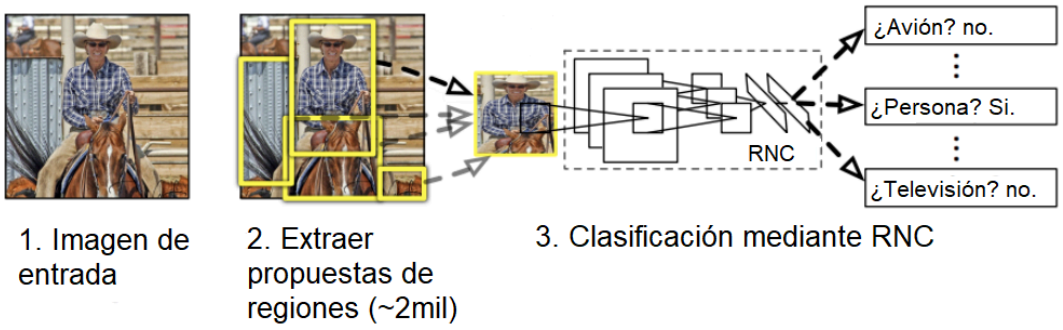


Figura 22. Flujo de procesamiento de las R-RNC.

Estos modelos tienen un gran desempeño en la extracción de las regiones de interés y su posterior clasificación, sin embargo, en este proyecto hemos optado por reemplazar el método por el cual las obtienen pero manteniendo el acercamiento de posterior clasificación de las regiones de interés. El método propuesto para obtener las regiones de interés está basado en sacar provecho de una característica intrínseca de los objetos astronómicos; *su contorno es fácilmente perceptible en contraste con el cielo oscuro.*

Obtención de contornos de regiones de interés

Comenzaremos detallando la implementación del proceso de obtención de contornos para hallar las regiones de interés. Para ello necesitaremos poder identificar aquellos píxeles contiguos que comparten una similitud en particular. En nuestro caso, para poder delimitar las galaxias, hemos tomado como factor de similitud la intensidad de los píxeles que componen la imagen. Debido a que la intensidad de un píxel no se correlaciona con el color del mismo, simplemente podremos tomar un solo canal de color de la imagen (o lo que es lo mismo convertirla a una en escala de grises tomando el valor de mayor intensidad de los 3 canales) y detectar los contornos en esta nueva escala.

A continuación a modo de hallar los contornos en una imagen en escala de grises, aplicaremos un método conocido como *Binary Thresholding*, el cual

consiste en convertir todo píxel que supere un umbral de intensidad establecido en negro y en blanco si no fuera el caso. En la Figura 23 se puede ver un ejemplo sobre la transformación descrita. Esto facilita en gran medida la labor del algoritmo [11] de detección de contornos de la librería OpenCV.

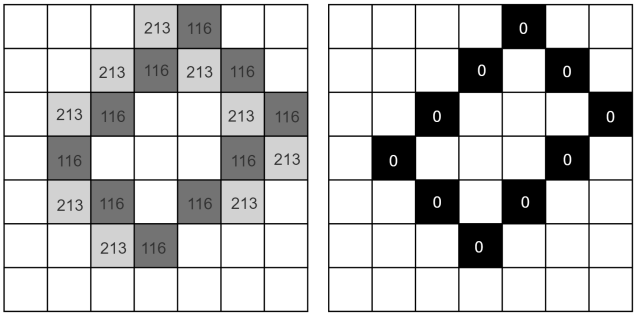


Figura 23. Ejemplo de transformación de una imagen en escala de grises (izquierda) a una imágen binaria (derecha) para la búsqueda de contornos, utilizando un umbral de 117. Los números enteros en cada pixel representan el valor en una escala de 0 a 254 (negro - blanco). Los píxeles vacíos tienen valor 254.

Observación: Cabe destacar que como el cielo de una imagen astronómica generalmente tiene una tonalidad oscura, hemos invertido el color al cual se le aplica a los píxeles que superan el umbral en Binary Thresholding, esto es, convirtiendo en blanco aquellos píxeles con alta intensidad y en negro (cielo) aquellos que no lo hagan.

En este proyecto el umbral de intensidad del píxel fue establecido en un 50, por ende todo valor del pixel en la imagen en escala de grises que supere este valor, será establecido como blanco y aquel que no lo haga como negro. Este valor se tomó a conciencia de ser estrictos a la hora de reconocer píxeles con información útil a costa de potencialmente perdernos de una parte de una galaxia, como puede ser un brazo espiral. Para minimizar este problema, a la hora de obtener un acercamiento de la región, daremos lugar a un margen amplio en cada lado.

Hasta aquí hemos descrito el algoritmo utilizado para obtener contornos, cuya implementación ha sido lograda mediante el uso de la librería openCV[6] que

[6] https://opencv.org/

provee los métodos: *opencv2.threshold()* cuyos parámetros son el umbral de intensidad y el valor máximo al cual se le aplica a los píxeles que superan este umbral, en nuestro caso 255 (blanco) y el método *opencv2.findContours()* cuyos parámetros son la imágen en escala de grises, el modo del algoritmo de búsqueda de contornos RETR_TREE[7] el cual preserva la relación de "anidamiento" entre ellos y finalmente el método de representación de los contornos CHAIN_APPROX_SIMPLE[8], que comprime las líneas verticales, horizontales y diagonales. Esta última función retorna los contornos en un arreglo de vectores conteniendo las coordenadas xey de cada píxel que forma parte de cada contorno hallado.

Finalmente obtendremos las regiones de interés hallando los rectángulos que las enmarcan. Para ello haremos uso de una función *opencv2.boundingRect()* provista por la librería OpenCV, la cual a partir de un contorno obtiene la posición del píxel al extremo superior izquierdo de un contorno, el ancho y el alto por el cual se extiende el rectángulo.

Filtrado de regiones de interés

No obstante obtener los contornos y los rectángulos de una imagen es la primera etapa para reconocer las regiones de interés, no podemos evitar notar que pueden contener información no relevante. Recordemos que la búsqueda de regiones de interés tiene como objetivo poder ingresarlas en un clasificador y poder obtener una respuesta sobre su contenido y luego filtrarlas según ésta, pero ese proceso es costoso y debemos evitar ingresar regiones que podríamos saber con anterioridad que no serán de utilidad. Una de las primeras precauciones que debemos tener en cuenta es evitar tomar un área muy chica como región de interés. Como mencionamos de manera introductoria, antes de ingresar las regiones de interés en el clasificador nosotros obtendremos una imagen de mayor resolución de las mismas aprovechando la herramienta de la SDSS. Pero un contorno reconocido puede tener un área muy pequeña y la imagen obtenida

---

7 https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#ga819779b9857cc2f8601e6526a3a5bc71

8 https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#ga4303f45752694956374734a03c54d5ff

tendrá una resolución pobre como para que el clasificador pueda procesarla correctamente. Para evitar este problema, podríamos quedarnos únicamente con aquellos que enmarquen un área lo suficientemente grande.

Para ello, el primer filtrado sobre las regiones de interés que desarrollamos consiste en seleccionar solamente aquellos rectángulos que tengan un área suficientemente grande. El acercamiento intuitivo para aplicar este filtro es tomar solamente aquellas cuya área supere un cierto valor en píxeles (área fácilmente obtenible desde los contornos), pero recordemos que estamos analizando una imágen astronómica que puede tener una escala variable y un área de un cierto valor en píxeles delimita regiones de distinto tamaño según la escala utilizada. Es por esto que hemos optado por un acercamiento que no depende de la escala sino más bien de la distancia de las diagonales del rectángulo. A fin de lograr esto debemos traducir los puntos en píxeles superior izquierdo e inferior derecho de cada región de interés a su correspondiente coordenada ecuatorial y filtrarlos según la distancia en segundos de arco que hay entre ellos. Para este proyecto hemos tomado como límite de distancia entre las coordenadas un valor de 13 segundos de arco.

Nota: Para realizar la conversión de coordenadas es necesario relativizar la posición del valor de x e y de la coordenada en píxeles a los límites inferior y superior en *AR* y *DEC* respectivamente, de la imagen de las coordenadas iniciales delimitadoras de la zona de estudio.

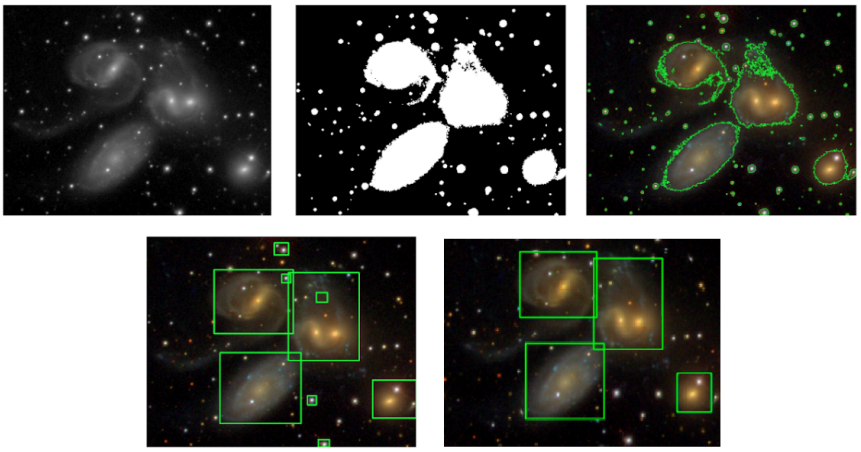


Figura 24. Ejemplo de la aplicación de los pasos para obtener las regiones de interés en una imagen del Quinteto de Stephan. De izquierda a derecha: transformación a escala de grises, aplicación del método de Binary Thresholding, obtención de contornos y rectángulos delimitadores de las regiones de interés una vez filtrados por distancia.

Otra de las precauciones que debemos tener antes de ingresar las regiones de interés al clasificador es prevenir ingresar un objeto astronómico que no sea necesariamente una galaxia, como puede ser una estrella, un defecto del telescopio, entre otros. Para solventar este problema, haremos uso de otra facilidad que provee la herramienta de la SDSS, la cual consiste en obtener una etiqueta sobre grupos de alto nivel de un objeto que se ubica en una coordenada. Pero al querer hacer uso de este acercamiento surge un problema, nosotros no conocemos dentro de la región de interés la coordenada exacta donde se ubica el objeto contenido o el centro del mismo que queremos filtrar. Lo que haremos entonces es hacer una búsqueda en la base de datos de la SDSS por objetos que se encuentren dentro de un círculo ubicado en una coordenada central del rectángulo con un determinado radio[9]. A partir de allí, si no detectamos un objeto con clasificación de galaxia, simplemente descartamos la región. Debemos aclarar que este método no resulta en un filtro completamente fiable, pero sí constituye una ayuda a la hora de descartar regiones de interés. A modo de ejemplo, de las regiones de interés que obtuvimos para el Quinteto de Stephan, este último filtro permite descartar las que no corresponden con galaxias, dejando únicamente las cuatro relevantes, como podemos notar en la Figura 24.

Obtención de regiones para clasificación

Finalmente, y para cerrar la sección de extracción de regiones de interés, explicaremos el último paso que haremos antes de ingresar las imágenes de las secciones en el clasificador RNC. Este paso consiste en obtener una imagen de mayor escala de cada una de ellas, lográndolo mediante el mismo método que hemos utilizado para obtener la imagen inicial del estudio, pero esta vez a partir de la coordenada central de la región de interés y la longitud en píxeles del lado más largo del rectángulo que la delimita.

[9] http://skyserver.sdss.org/dr17/SearchTools/sql

En la Figura 25 podemos ver el resultado de obtener las regiones de interés en una escala mayor y con la mayor resolución posible sobre el ejemplo del Quinteto de Stephan.

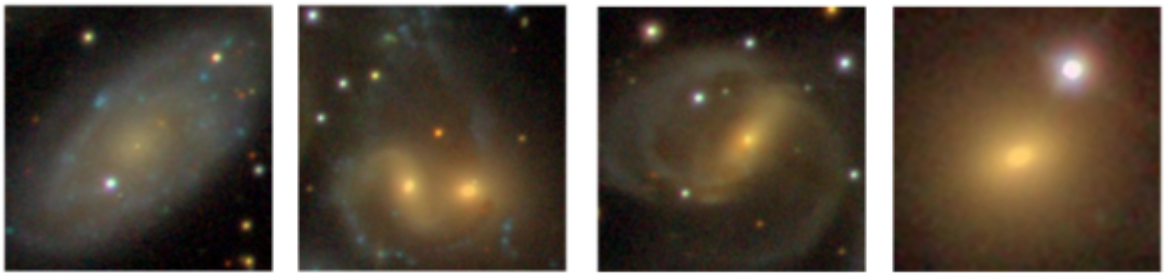


Figura 25. Imágenes de las regiones de interés detectadas en el Quinteto de Stephan.

Etapa 2 - Clasificación Morfológica

En esta sección describiremos el proceso de construcción de la RNC utilizada en la parte intermedia y de mayor importancia de nuestro flujo de clasificación. Dado que es un proceso el cual consta de varias etapas, dividiremos esta sección buscando explicar cada una de estas etapas de manera detallada y clara. Comenzaremos explicando la obtención y procesamiento de los datos necesarios para entrenar el modelo de clasificación, para luego explicar la arquitectura del modelo. Para finalizar esta sección, presentaremos el detalle del proceso de entrenamiento de la red y los resultados obtenidos.

Datos

Como mencionamos en la sección Conceptos Preliminares, el proceso de armado de una Red Neuronal Convolucional requiere de un conjunto de imágenes las cuales puedan ser usadas en las diversas etapas de su modelado. Las mismas son utilizadas del siguiente modo; en primer lugar, una sección del conjunto es tomado para realizar el entrenamiento, y al mismo tiempo otro conjunto mutuamente excluyente es tomado para validar el proceso de entrenamiento. Finalmente se elige otra sección la cual será usada para probar que la clasificación obtenida sea correcta y tenga un porcentaje de acierto suficientemente elevado como para ser fiable.

La búsqueda de un conjunto de datos adecuado no solamente toma relevancia en las RNC, sino en todos los modelos de clasificación y/o predicción basados en Machine Learning y, debido a la importancia de encontrar un conjunto de imágenes adecuado, en este capítulo nos adentraremos en el proceso de búsqueda del mismo y cómo fueron transformados para ser de utilidad en la creación de la Red Neuronal Convolucional propuesta.

Búsqueda y selección

Al buscar un conjunto de datos en nuestro caso para desarrollar una RNC, necesitamos que el mismo cumpla con las siguientes condiciones:

- Que tenga una amplia variedad de imágenes para que el modelo tenga la suficiente cantidad de información para poder detectar patrones
- Que la distribución de cantidad de imágenes por clase o tipo de galaxia sea uniforme (ahondaremos más en el tema en la sección <SECCIÓN>).
- Que esté libre de sesgos o repita patrones culturales.
- Que las anotaciones de los datos sean de confianza.

Si bien existe una gran disponibilidad de imágenes astronómicas hoy en día gracias a las observaciones realizadas a lo largo de las últimas décadas, es de vital importancia encontrar aquellas que se adecúen correctamente al caso de uso y provean información relevante a la clasificación de galaxias que queremos construir. Es por esto que inicialmente se optó por acudir a estudios de Machine Learning predecesores a éste en pos de obtener información que pueda ayudarnos a elegir un conjunto de imágenes correcto. Para ello, observamos varios aspectos de las publicaciones, a saber: qué modelo fue el utilizado, qué tipo de imágenes tomaron, de qué manera fueron modificadas para ser usadas, y finalmente si están abiertas públicamente.

Uno de los primeros conjuntos de imágenes que nos encontramos investigando las publicaciones consiste en una simulación de imágenes de galaxias conocido como LSST[10]. Una ventaja notable de este conjunto es que las clasificaciones existentes son acertadas ya que simulan las imágenes a partir de galaxias bien conocidas. Si bien inicialmente su utilización se veía factible, optamos por evitarlo ya que el objetivo de su creación era entrenar modelos de clasificación para observaciones futuras, cuyo método de captura de las galaxias no resultaba adecuado para este proyecto.

Seguidamente nos topamos con el dataset SDSS, el cual ha cubierto gran parte del cielo con sus observaciones distribuidas en distintos lanzamientos. No obstante su gran cantidad de datos, la búsqueda de galaxias en sus imágenes y obtener la clasificación morfológica de cada una se convirtió en un impedimento para este proyecto, ya que desviaba en gran medida la atención de nuestro objetivo. Sin embargo, notamos que esta recopilación de imágenes fue a su vez utilizada por

[10] https://www.kaggle.com/competitions/PLAsTiCC-2018/overview

un proyecto sin fines de lucro, el cual solventaba este último inconveniente mediante un acercamiento *crowdsourced* (colaboración abierta al público). Este método consiste en incentivar la participación de ciudadanos interesados en ayudar con la clasificación morfológica de las galaxias obtenidas en los lanzamientos del SDSS. El proyecto es nombrado **Galaxy Zoo** y han desarrollado numerosas versiones del mismo. Por fortuna, estas diversas iteraciones sobre el proyecto nos permiten hacer una fina selección de aquella que más se adecúe a entrenar nuestro modelo de Machine Learning.

Luego de evaluar las alternativas mencionadas previamente, se optó por una versión del Galaxy Zoo denominada Galaxy Zoo DECaLS. Los motivos principales de su elección fueron su amplia cantidad de imágenes, el poder identificar mediante coordenadas las galaxias en el cielo y la variedad de propiedades por las cuales se clasifican las galaxias. Si bien el método de obtención de las imágenes en el dataset escapa al alcance de esta tesis, a continuación describimos brevemente el proceso.

En primera instancia, las imágenes fueron creadas a partir de datos obtenidos por el estudio de DECaLS [12] realizado en el observatorio Cerro Tololo, Chile. Con el objetivo de filtrar y separar las galaxias de otros objetos astronómicos en las imágenes una vez creadas, se basaron en un estudio predecesor en el cual ya se habían identificado las galaxias existentes en un área similar del cielo. Dado que el rango de estudio no coincidía exactamente, solamente se tomaron las galaxias ubicadas dentro del área superpuesta [12] . Finalmente, las imágenes obtenidas de cada galaxia fueron anotadas utilizando el método de *crowdsourcing* mencionado antes.

Cabe destacar que las opciones disponibles para que los voluntarios voten no eran específicamente según la clasificación morfológica por tipos de Edwin Hubble, sino que tomaron un grado de detalle más fino.

Principalmente el proyecto de Galaxy Zoo DECaLS consistió en mostrar imágenes de galaxias a personas voluntarias, a la vez que se mostraban opciones de su aspecto morfológico (características o propiedades) para que sean seleccionadas según la que mejor se ajuste, de aquí su nombre *crowdsourced* o colaboración abierta el público. A medida que los usuarios votaban, se desplegaron distintas

opciones dependiendo de las características seleccionadas previamente. De esta manera, formando un árbol de opciones el cual se va expandiendo a medida que el voluntario indica su elección como se ilustra en la Figura 26. A su vez, notamos que la mayor parte de las propiedades presentadas para su votación, podían ser calificadas según su grado de presencia, por ejemplo, qué tan marcada es la barra central de la galaxia en el caso de poseerla.

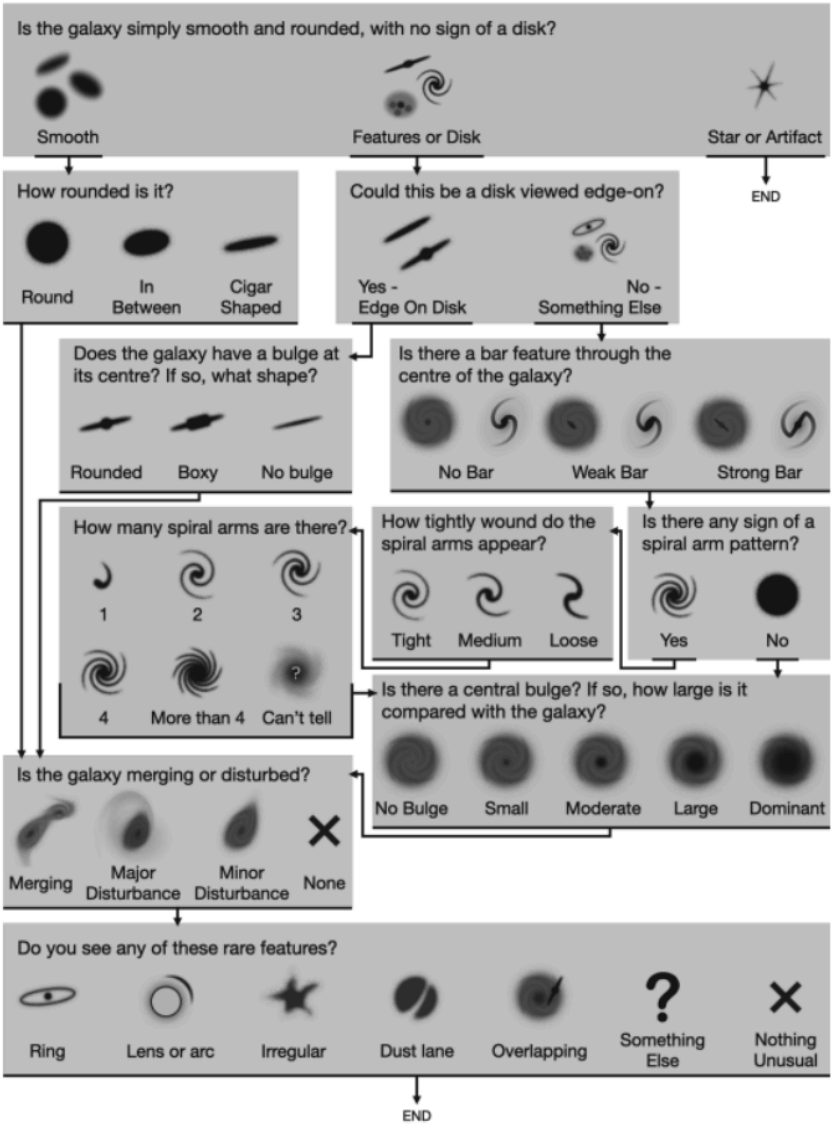


Figura 26. Árbol de opciones presentado a los usuarios. Formando caminos de la raíz hacia las hojas en donde se analiza la galaxia mostrada.

Las propiedades disponibles permiten distinguir a una galaxia según:

- Si es suave o presenta alguna particularidad.
- La presencia de un disco.
- Si está vista de canto.
- La presencia de brazos espirales y dado del caso, la cantidad de ellos.
- El aspecto del centro.
- Qué tan redonda es.
- Si aparenta estar fusionándose con otra galaxia.

Preprocesamiento - Obtención de datos

Explicaremos en detalle el curso de acciones realizado para obtener las imágenes y su manipulación para así lograr que se adecúen a nuestro caso de uso; poder entrenar y validar nuestra RNC. Para ello, narraremos cómo fue nuestro proceso de los datos del estudio de Galaxy Zoo para ser de utilidad.

El dataset obtenido consta de las imágenes de las galaxias a la par de un archivo de valores conteniendo un listado de las galaxias con los resultados del estudio en cuestión. En cada fila del mismo nos encontramos con un identificador único de la galaxia descrita, las coordenadas ecuatoriales indicando su ubicación y los valores obtenidos en cada una de las características. Podemos verlas detalladas en la Figura 27 con nombres en representación de la característica evaluada. Cada una de estas propiedades tiene asociada una columna *votos-totales* la cual indica la cantidad de votos por característica, y cada posible respuesta dentro de ella tiene asociada dos columnas; *característica_respuesta_X* indicando la cantidad de votos a la respuesta X, y *caracteristica_respuestaX_fraccion* indicando qué fracción representa la cantidad de votos de la respuesta X al total de votos en la propiedad. Por ejemplo, para la propiedad barra central, obtenemos *bar_total-votes*, junto con *bar_strong, bar_strong-fraction*, *bar_weak, bar_weak_fraction, bar_no* y *bar_no_fraction*.

smooth-or-featured_smooth	disk-edge-on_yes	bar_strong
smooth-or-featured_featured-or-disk	disk-edge-on_no	bar_weak
smooth-or-featured_artifact	has-spiral-arms_yes	bar_no
	has-spiral-arms_no	bulge-size_dominant

bulge-size_large	edge-on-bulge_none	spiral-arm-count_4
bulge-size_moderate	edge-on-bulge_rounded	spiral-arm-count_more-than-4
bulge-size_small	spiral-winding_tight	spiral-arm-count_cant-tell
bulge-size_none	spiral-winding_medium	merging_none
how-rounded_round	spiral-winding_loose	merging_minor-disturbance
how-rounded_in-between	spiral-arm-count_1	merging_major-disturbance
how-rounded_cigar-shaped	spiral-arm-count_2	
edge-on-bulge_boxy	spiral-arm-count_3	

Figura 27. Los resultados del estudio del dataset de Galaxy Zoo DECaLS están representados por estas columnas.

La cantidad de imágenes de galaxias clasificadas en Galaxy Zoo DECaLS logró alcanzar las 300.000 gracias a la participación de casi 8 millones de personas a lo largo del estudio. Debido al gran tamaño de descarga que conlleva esa magnitud de imágenes, se optó inicialmente por comenzar solamente con una primera parte, la cual consta de 24 mil imágenes (aproximadamente el 9% del total) e ir evaluando la necesidad de incorporar nuevas.

Una vez obtenidas las imágenes comenzó la etapa de preprocesamiento. Dado que el archivo de valores no solamente consta de las columnas mencionadas en la Figura 27, sino que viene en conjunto de otras que añaden información no relevante a nuestro estudio, se realizó una sección en el laboratorio encargado de eliminar las columnas dispensables del archivo de valores. Dicho laboratorio se encuentra implementado en *1. Filtrado del repositorio* anexado. A su vez, y sumado a que el archivo de valores descargado es en representación de todas las imágenes del dataset de Galaxy Zoo y no solamente de la sección descargada, elimina aquellas filas que contengan información de una imágen de una galaxia no incluida en la parte obtenida.

A continuación, como el objetivo del estudio de las imágenes en Galaxy Zoo DECaLS tuvo en cuenta aspectos más particulares de las galaxias (y se diferencia de nuestro objetivo de clasificarlas mediante su tipo de acuerdo a la clasificación de Edwin Hubble), tuvimos que obtener su tipo de manera indirecta. Afortunadamente, cada tipo de galaxia tiene características que la identifican y

algunas de las cuales sí han sido evaluadas en el estudio de Galaxy Zoo. Esto habilita poder realizar un sencillo programa el cual asigne una nueva etiqueta correspondiente a la clasificación morfológica a cada galaxia según sus propiedades. Recordamos que las clasificaciones morfológicas para las galaxias tomadas en esta tesis son *elípticas*, *espirales*, *de canto* o en *fusión*.

A su vez, debido a que el estudio realizado en Galaxy Zoo no indica que una galaxia cumpla con una propiedad de manera determinista, sino más bien la fracción de votos en relación al total de una respuesta a una propiedad, se tomaron umbrales mínimos variables de votos para tener un grado de confianza sobre la misma. Los umbrales usados pueden verse en la Figura 28. Como guía para elegir los umbrales tomamos como base la tabla 2 de Galaxy Zoo DECaLS [13] la cual indica el corte de votos mínimos para identificar una propiedad y fue construida mediante una inspección visual por los autores del mismo estudio.

El programa descrito el cual a partir de las propiedades de una galaxia asigna una nueva etiqueta consta de recorrer las filas del archivo de valores del dataset verificando la cantidad de votos en sus propiedades y asigna un tipo de galaxia si cumple con los respectivos umbrales. El algoritmo se encuentra implementado en *2. Clasificación* del laboratorio anexado. Cabe aclarar que aquellas galaxias que no cumplan con ninguno de los requisitos de los diferentes tipos de galaxias simplemente son descartadas (aunque pueden usarse para probar el funcionamiento del modelo de clasificación una vez desarrollado). Por otro lado, el algoritmo solamente clasifica teniendo en cuenta condiciones positivas, es decir, que una galaxia cumpla con una propiedad en una fracción mayor a un cierto umbral. Dicho de otro modo, no busca que no se cumpla con una propiedad.

En cuanto a la implementación del algoritmo, fue realizada usando la librería *Pandas* de Python para construir un DataFrame a partir del archivo de valores .csv provisto por el dataset. Para finalmente hacer uso de la librería *numpy* para añadir una columna dentro del dataset con la nueva clasificación correspondiente a sus propiedades.

Tipo de Galaxia	Categoría y Propiedad	Fracción de votos mínimos en relación al total
Elíptica		
	smooth-or-featured_smooth	0.7
	has-spiral-arms_no	0.7
	how-rounded_round + how-rounded_in-between	0.7
	merging_none + merging_minor-disturbance	0.9
Espiral		
	smooth-or-featured_featured-or-disk	0.3
	has-spiral-arms_yes	0.7
	spiral-arm-count_ 3 | spiral-arm-count_4 | spiral-arm-count_more-than-4	0.6
	merging_none	0.7
De canto		
	smooth-or-featured_smooth	0.6
	merging_none	0.8
	disk-edge-on_yes	0.6
Galaxias en fusión (1)		
	merging_merger	0.7
Galaxias en fusión (2)		
	merging_major-disturbance + merging_minor-disturbance	0.7
	spiral-winding_loose	0.65

Figura 28. Umbrales de votos mínimos para determinar que una galaxia es de un determinado tipo según la fracción de votos de sus propiedades. Existen sumas de fracciones de propiedades en algunos tipos de galaxias.

Como resultado de la ejecución del algoritmo, obtenemos dos archivos de valores; uno con las galaxias clasificadas y el restante con aquellas que no lograron alcanzar ninguna categoría. Para realizar una inspección visual de los resultados de la recategorización ya partir de allí realizar iteraciones sobre los umbrales para obtener aquellos que mejor se ajusten se creó un programa en Python el cual es accesible en *3. Inspección* del laboratorio anexado en este documento. Este programa brinda una herramienta que permite tomar muestras de las imágenes junto con su categoría. Puede verse un ejemplo en la Figura 29.

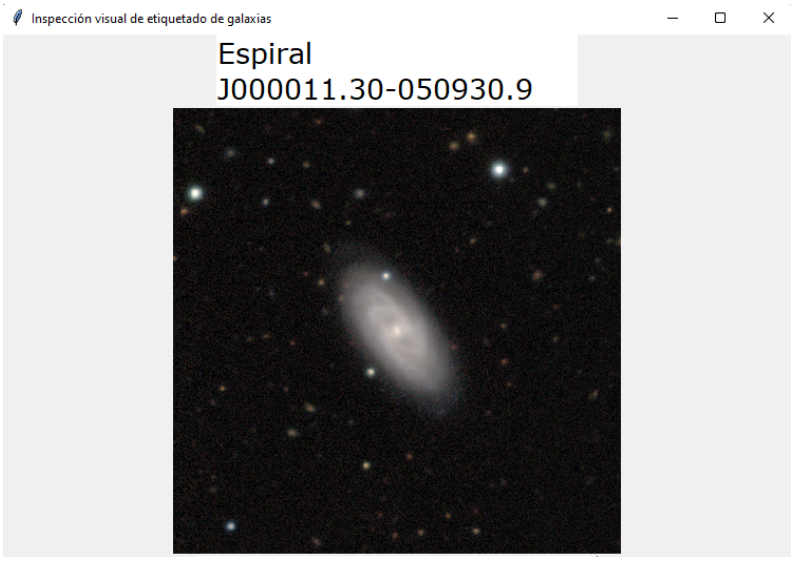


Figura 29. Ejemplo de ejecución del programa en donde podemos realizar una inspección visual de las nuevas etiquetas asignadas a las galaxias. Dentro de la ventana se encuentran el tipo asignado a la galaxia, su identificador y la imagen.

La decisión de obtener dos archivos de valores con las galaxias clasificadas y las no clasificadas, permitió al programa alternar el archivo del cual tomamos el resultado y así poder, mediante una inspección visual, ir modificando los valores de los umbrales para que aquellas que no hayan sido clasificadas y sean indudablemente de un tipo morfológico, se las clasifique de manera correspondiente. Cabe destacar que para que la inspección visual mediante el muestreo del programa sea representativa del conjunto de imágenes en total, éstas fueron seleccionadas siguiendo la técnica de muestreo aleatorio simple donde se seleccionan las imágenes al azar.

Preprocesamiento - Datos finales

Para poder entrenar, validar y probar una RNC hemos elegido la librería Keras, una de las principales alternativas para desarrollar modelos de machine learning. La librería requiere un formato particular para leer los datos que se usarán durante el entrenamiento de la red (usando la función *image_data_from_directory*). Dicho paso puede verse implementado en el laboratorio *6. Separación del repositorio* anexado.

Pese a que el conjunto inicial de imágenes es suficiente para entrenar la RNC, no quisimos dejar de lado la posibilidad de experimentar con una cantidad más elevada de imágenes para cada galaxia. Esto es debido a que puede ser uno de los factores determinantes del rendimiento de la red a la hora de evaluarla. Para ello se tuvieron en consideración dos opciones:

- Hacer uso de una técnica para incrementar la diversidad de las imágenes; la cual consta de modificar y transformar las imágenes originales y obtener otras con aspectos transformados (rotación, escala). Esta técnica es conocida como *Data Augmentation*.
- Al haber obtenido solamente una porción de las imágenes del dataset de Galaxy Zoo, hacer uso de una nueva porción.

Si bien la primera opción resulta ser de interés cuando los datos accesibles son reducidos y se busca obtener alteraciones de objetos a partir de otros ya conocidos, se optó con el segundo acercamiento por la transferencia que conlleva a los datos reales y de uso en la práctica.

De acuerdo con la decisión tomada, se procedió a obtener de la web de Galaxy Zoo DECaLS una segunda sección del dataset, y la implementación de los laboratorios fue modificada para que reflejen la disponibilidad de las nuevas imágenes. Para evitar hacer cambios destructivos, se crearon copias de los laboratorios ya existentes y hacer las modificaciones allí y, a su vez, facilitando posteriormente la comparación de los resultados obtenidos con los dos grupos de imágenes. A modo de comparación, en la Figura 30 se pueden ver la cantidad de imágenes por clase una vez etiquetadas mediante el proceso descrito previamente.

Tipo de Galaxias	Cantidad de imágenes
Elípticas	970
De canto	1268
Espirales	315
En fusión	684

Solo con una sección

Tipo de Galaxias	Cantidad de imágenes
Elípticas	1902
De canto	2489
Espirales	592
En fusión	1254

Ambas secciones

Figura 30. Cantidad de imágenes por tipos morfológicos de galaxia tanto con una primera sección del dataset de Galaxy Zoo DECaLS como con una segunda sección añadida.

Por otra parte, como podemos observar en la figura anterior, existe una desigualdad considerable en la cantidad de imágenes entre tipo de galaxias. Por ejemplo, para el tipo morfológico de galaxias en fusión contamos en el orden de 1600 imágenes mientras que en las galaxias de tipo elípticas en el orden de 14000, más de 8 veces su cantidad. Al querer entrenar un modelo de aprendizaje automático con un conjunto de clases no balanceadas, existe el problema de que el modelo no pueda ser capaz de identificar la clase que menos cantidad de datos tiene. Esto sucede al no haber visto una cantidad suficiente de datos para ella en el entrenamiento y conlleva a que el modelo se especializa mayormente en reconocer aquellas clases que más revisó.

El problema mencionado es habitual en el área de aprendizaje automático y existen distintos acercamientos para intentar abordar el problema. A continuación describimos algunos de ellos.

Un primer acercamiento podría ser utilizar tipos de métricas adecuados para un dataset no balanceado. Podemos optar por medir el rendimiento de la RNC tomando alguna de las métricas descritas en la subsección Métricas, de la sección Conceptos Preliminares.

Otro acercamiento es aplicar una técnica de Sobre Muestreo, la cual consiste en generar más datos de las clases menos frecuentes. Una ventaja de este acercamiento es que no se pierde información, pero al generar datos a partir de otros existentes, puede suceder un sobreajuste en aquella clase y no ser capaz de extrapolar a datos de la misma clase no vistos previamente.

Contrariamente existe una técnica de Sub Muestreo, la cual consiste en reducir la cantidad de datos de las clases que más contienen. Aunque su aplicación puede hacer perder información útil para el entrenamiento de las clases que han sido reducidas.

Como última opción podríamos establecer un peso de entrenamiento para cada clase. Dado que puede resultar difícil hallar más datos de las clases que menos tienen, otra técnica a utilizar consiste en asignar un peso distinto a cada clase, logrando que el modelo de aprendizaje "preste más atención" a una clase que a otra. Esto se logra haciendo que los casos más raros (como las galaxias en fusión) contribuyan más a la función pérdida que los casos más comunes.

Al no tener una preferencia concreta, hemos probado con algunos de estos acercamientos descritos para quedarnos con aquel que mejor resultados logre. Para realizar estas pruebas, generamos seis casos que combinan el uso de ambas secciones del dataset de la SDSS más las variantes mencionadas previamente.

1. Utilizar una cantidad de imágenes desbalanceadas por clase solamente con la primera sección de imágenes de la observación DECaLS.
2. Hacer uso de la técnica de Sub Muestreo, es decir, eliminar imágenes de las clases que más imágenes tienen para equipararlas con la que menos tiene.
3. Utilizar una cantidad de imágenes desbalanceadas por clase pero con ambas secciones del dataset. Como vemos en la Figura 30, las proporciones de imágenes por galaxia se mantienen respecto a utilizar una única sección de datos.
4. Hacer uso de la técnica de Sub Muestreo, pero con el límite inferior establecido en la cantidad de imágenes del tipo de galaxias que menos tiene sumando ambas secciones.
5. Generar un parámetro de peso en el entrenamiento haciendo uso del total de las imágenes descargadas.

Estas variaciones resultan en los distintos datasets que usamos para entrenar el modelo. Más adelante en este informe haremos una comparación de los resultados obtenidos con cada uno de ellos y discutiremos la razón de elección del dataset finalmente tomado. Cabe destacar que cada variación de este dataset requirió estructurar el laboratorio de tal manera de poder revertir los cambios y preservar los resultados, es decir, realizar las alteraciones del código de manera no destructiva.

Si bien ya contamos con las imágenes descargadas y con la distribución de imágenes establecida, antes de pasar a la sección de entrenamiento daremos una breve descripción del procesamiento realizado a cada imagen antes de ser ingresada al modelo de RNC. Este procesamiento consiste en el recorte y reescalado de la imagen y finalmente la normalización de los píxeles.

En primer lugar, como queremos entrenar a la red para que pueda clasificar solamente la galaxia central de la imagen y notamos mediante una inspección visual por muestreo que las imágenes descargadas contienen un gran margen desde los bordes hasta la galaxia, decidimos recortar la imagen a una región central. Podemos notar en la imagen izquierda de la Figura 31 un ejemplo de lo mencionado. Esto permitirá al clasificador ignorar los demás objetos astronómicos que pueden aparecer en la imagen y enfocarse únicamente en la región central.

En segundo lugar, en vistas de optimizar el entrenamiento de la RNC, debemos redimensionar la imagen a un tamaño lo suficientemente chico para que no sea computacionalmente costoso revisar cada imagen pero debemos evitar perder demasiada información al comprimir la imagen. Luego de varias iteraciones nos decidimos por usar imágenes de 180x180 píxeles lo cual permite un balance adecuado entre ambos. Cabe destacar que respetar el orden de estas dos etapas de procesamiento de las imágenes es importante, ya que si primero se redimensiona la imagen y luego se recorta, estaremos perdiendo información al comprimir los píxeles centrales en una escala menor. Los detalles del algoritmo usado para el reescalado de la imágen se escapan del alcance de este documento.

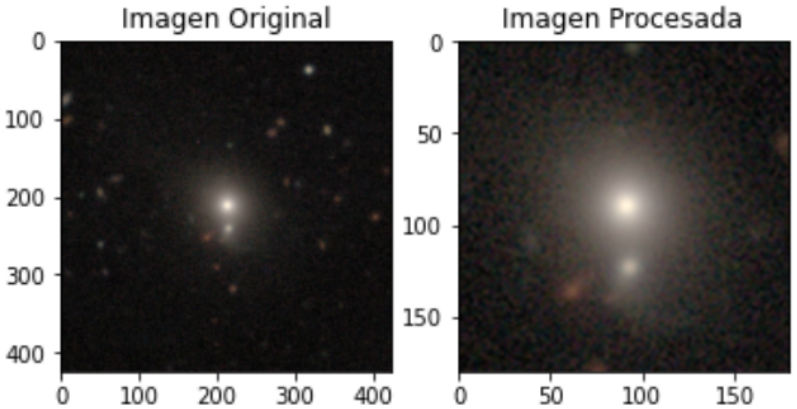
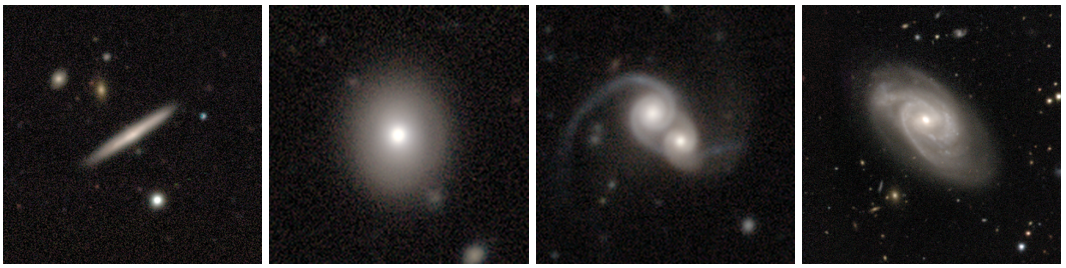


Figura 31. Imagen de una galaxia antes y después de aplicar las 2 primeras etapas de preprocesamiento.

Finalmente podemos aplicar una normalización de los valores de los píxeles para optimizar aún más el proceso de entrenamiento del modelo. Cuando usamos una imagen y la ingresamos a una RNC, emplear datos numéricos altos y muy variables suele dificultar el proceso de aprendizaje, es por eso que resulta conveniente normalizar los valores de los píxeles (0 a 255) a un rango de 0 a 1.

Como veremos en la sección de arquitectura de la RNC, estas tres etapas de preprocesamiento pueden añadirse como primeras capas de la misma gracias a la librería Keras, facilitando el proceso de entrenamiento de la RNC.



De canto Elíptica En fusión Espiral

Figura 32. Ejemplos de cada uno de los tipos morfológicos elegidos de las galaxias en el dataset de la SDSS.

Arquitectura de la Red

En esta sección daremos una explicación sobre la configuración elegida para la Red Neuronal Convolucional encargada de la etapa de la clasificación de las regiones de interés. Además incluiremos las pruebas de concepto realizadas, las decisiones de diseño tomadas y los problemas encontrados.

Antes de adentrarnos en la topología, recordemos brevemente cómo se componen las RNC. A grandes términos, este modelo de Machine Learning posee tres tipos de capas, las capas de convolución, las denominadas capas de pooling y las capas completamente conectadas.

En primer lugar tenemos las capas de convolución, las cuales llevan la mayor carga computacional dentro de la red. Estas capas utilizan una ventana deslizante más chica que la imagen, conocida como Kernel, para recorrer la imagen de izquierda a derecha y de arriba a abajo. Tanto la imagen como el Kernel son matrices de números y la idea general es que el Kernel obtenga información

relevante de cada porción de la imagen que va recorriendo durante el entrenamiento. Esto lo hace aplicando el producto escalar entre la matriz del Kernel y la porción de la imagen en cada iteración.

En segundo lugar se encuentran las capas de pooling, las cuales cumplen la función de reducir la información generada de la capa anterior preservando la relación espacial entre los elementos presentes en la imagen, aplicando un método conocido como *Submuestreo*. Son generalmente aplicadas luego de las capas de convolución y existen diversas maneras en el modo de que realizan el compilado de información.

Finalmente encontramos las capas completamente conectadas, las cuales contendrán las neuronas utilizadas habitualmente en una Red Neuronal Artificial convencional. Este tipo de red permitirá hacer un mapeo entre una entrada vectorial (en este caso la representación de una imagen generada por la parte convolucional de la red), y el resultado (clasificación) de la imagen.

Prueba de concepto

Para poder construir una red neuronal que tenga un buen rendimiento y tener una intuición de cómo utilizar de manera eficiente los datos disponibles, hemos diseñado una RNC simple a modo de poder comprender de mejor manera el dominio del problema al que nos enfrentamos. Además, recordamos que los conceptos teóricos utilizados a continuación están descritos en la sección Conceptos Preliminares.

Nota: Para el desarrollo de nuestra arquitectura hemos utilizado la librería de *TensorFlow* (tf en los recortes de código) la cual hace uso de la API de *Keras* [11].

Particularmente en nuestra arquitectura hemos añadido, previo a las capas de convolución, tres capas de entrada las cuales tienen la función de realizar las tareas de procesamiento de las imágenes como hemos mencionado en la sección anterior. En el siguiente recorte de la arquitectura vemos estas tres primeras capas, las cuales son de preprocesamiento y preceden a las capas de convolución de la Red.

[11] https://keras.io

```
tf.keras.layers.CenterCrop(center_height, center_width),
tf.keras.layers.Resizing(resize_height, resize_width),
tf.keras.layers.Rescaling(1./255)
```

La primera capa recibe una imagen de 424 x 424 píxeles y aplica un recorte al centro de la misma quedando solamente la región central de 160 x 160 píxeles. La segunda capa escala la imagen de 160 x 160 píxeles a su tamaño original de 424 x 424 píxeles para finalmente pasar esta imagen a la última capa de preprocesamiento encargada de normalizar los valores de la imagen a un rango de 0 a 1.

Luego de estas capas de procesamiento, la RNC incluye una serie de capas de convolución y pooling. Cada capa recibirá una entrada de la capa previa y emitirá un resultado que será utilizado por la siguiente capa. A continuación vemos un recorte ilustrando un bloque de convolución + pooling de la red.

```
tf.keras.layers.Conv2D(filters=16, kernel_size=7, activation='relu'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
tf.keras.layers.Dropout(0.1),
```

Partiendo desde la primer línea de código de este recorte, vemos una capa de Convolución, la cual está configurada para que:

- Use un Kernel con un tamaño de 7x7 píxeles. En nuestro caso, la idea general adoptada para seleccionar el tamaño del Kernel fue tomar inicialmente un tamaño grande, para poder obtener las características más dominantes de la imagen (como líneas, el fondo o cielo y contrastes), y luego ir reduciendo su tamaño para capturar las características de mayor detalle.

- Genere 16 Kernels. Es decir, que se entrenen 16 Kernels de 7x7 píxeles para poder obtener 16 mapas de características de 418x418 píxeles.

- Y para aplicar la no linealidad antes de hacer la siguiente operación de convolución, indicamos que haga uso de la función de ReLU. La adopción de esta función de activación se debe principalmente a dos factores. En primer lugar su facilidad de cómputo al ser simplemente una comparación

entre el valor de su entrada y 0. Además, su derivada es 1 o 0, dependiendo si la entrada es positiva o negativa, esto ayuda particularmente a la hora de realizar el backpropagation en el entrenamiento ya que el gradiente se obtiene sencillamente. Y por otro lado, si tenemos en cuenta que los filtros de las capas ocultas de la red entienden solamente una determinada característica, para cada imagen analizada será relevante un diferente conjunto de filtros, pero para aquellos que no sean relevantes no queremos que influyan negativamente en el resultado de la red, sino que dejen a los demás filtros determinar el resultado. Esto se cumple en la función ReLU ya que establecen en 0 los valores negativos.

Seguido de esto, notamos que luego de la capa de convolución nos encontramos con una capa de normalización por lote, lo cual es habitual que las acompañen en las RNC. El motivo de su uso reside en evitar el sobreajuste en el entrenamiento, evitando que se propague un valor alto producido por una capa temprana de la red a las subsiguientes capas, llevando los valores producidos luego de la operación de convolución a una escala en común o normalizada.

A continuación procedemos a aplicar la operación de max-pooling. Como explicamos en la sección Conceptos Preliminares, esta operación sirve para reducir la cantidad de información generada por la capa de convolución (los mapas de características) solamente tomando el valor más alto de una ventana establecida de 2x2 píxeles, de esta manera resaltando además la característica más fuerte dentro de la misma.

Finalmente aplicamos una capa de cancelación de neuronas o *Dropout*, la cual ayuda a la red a evitar el sobreajuste. Esto lo logra impidiendo que las primeras muestras del conjunto de entrenamiento influyan sobre algunas neuronas seleccionadas arbitrariamente. Ya que si dejáramos a la red ajustar los pesos de todas las neuronas, al iniciar el entrenamiento se verían ajustadas de manera desproporcionada en relación al ajuste que recibirán al finalizar el entrenamiento. Específicamente en nuestra arquitectura hemos establecido descartar un 10% de las neuronas de entrada de la capa.

La descripción hecha hasta aquí corresponde a la primera "sección" (la capa de convolución junto con sus operaciones siguientes) de la parte oculta de la red.

En nuestra arquitectura hemos replicado esta sección cuatro veces y en cada una de ellas hemos alterado algunos de sus parámetros. Principalmente incrementamos la cantidad de filtros creados en las capas subsiguientes en potencias de dos, pero a su vez disminuyendo su tamaño de a un píxel en ambas dimensiones. El motivo de este acercamiento es poder ir obteniendo en las primeras "secciones", donde los filtros son de tamaño mayor, filtros que sean capaces de capturar características más generales de la imagen, a medida que se achican, debemos requerir mayor cantidad de filtros, pero a medida que se avanza en la red incorporar más filtros que ayuden a identificar características de más detalle en la imagen, pero para ello necesitaremos más filtros dado que el nivel de detalle en el análisis de la imágen se incrementa.

Para cerrar nuestra arquitectura debemos realizar la tarea de clasificación a partir de los mapas de características obtenidos. Podemos lograr esta tarea mediante capas densas o redes completamente conectadas en la salida de la parte convolucional de la red. Es necesario aclarar que una capa densa toma como entrada un tensor (o vector) unidimensional, mientras que la salida de la parte convolucional de la red será un tensor tridimensional. Para lograr conectar ambas subredes se requiere “aplanar” la salida de la parte convolucional utilizando una capa *Flatten*.

```
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(512, activation='relu'),
tf.keras.layers.Dropout(0.2),

tf.keras.layers.Dense(256,  activation='relu'),
```

La red incorpora una primera capa densa con 512 neuronas con una función de activación ReLU y una capa subsiguiente de cancelación de neuronas pero está vez con un 20% de las neuronas descartadas. La adopción de esta configuración fue resultado de una base empírica ya que en general no existe una norma para la selección del tamaño de una capa densa de neuronas. Acompañamos esta capa con otra completamente conectada con 256 neuronas, lo que incentiva a la capa a generalizar mejor la información recibida. Es importante destacar que el objetivo de estas capas completamente conectadas es poder clasificar las características

detectadas por las capas de convolución y determinar cuáles características son relevantes o no para una clase y establecer combinaciones no lineales entre ellas.

```
tf.keras.layers.Dense(num_classes,  activation='softmax')
```

Para finalizar la arquitectura, hemos incorporado una capa completamente conectada a la anterior, pero compuesta por 4 neuronas, esto es, una neurona por cada tipo de galaxia posible (*elíptica*, *espiral*, *de canto* o en *fusión)*. Además, debido a que queremos resolver un problema de clasificación múltiple donde existen 4 tipos de galaxia y son mutuamente excluyentes entre sí, debemos usar una función que obligue a realizar una distribución de probabilidad, esto es, que la suma de probabilidades por clases para una imagen en la salida de la red sea igual a 1. Para ello hemos indicado a la capa que utilice como función de activación la función *softmax*. Cabe aclarar que cada valor de la salida de la función de softmax en una neurona es interpretada como la probabilidad de que la imagen pertenezca a la clase correspondiente a esa neurona, sin embargo este valor está descalibrado, ahondaremos sobre ello en la sección <SECCIÓN>. En la Figura 33 podemos ver ilustrada la arquitectura del modelo utilizado como prueba de concepto.

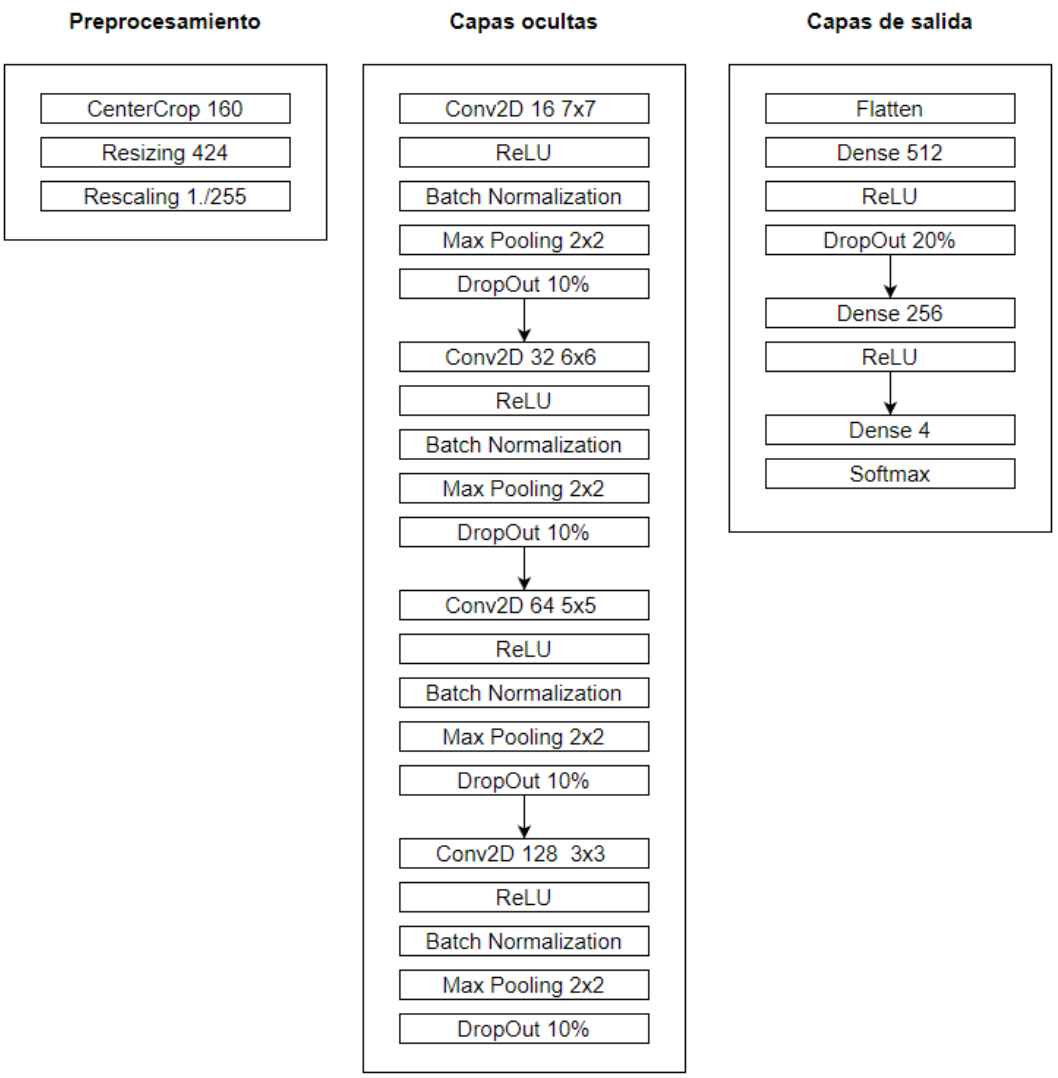


Figura 33. Arquitectura del modelo de RNC utilizado inicialmente.

Arquitectura adoptada

A modo de obtener el mejor rendimiento en la etapa de clasificación de nuestro flujo de identificación de galaxias en una región, en vez de idear una arquitectura desde cero, en efecto, decidimos partir de topologías de red que estén siendo ampliamente utilizadas por la comunidad y sean conocidas por su alto rendimiento. Es por ello que hemos realizado un proceso de investigación sobre modelos clasificadores que derivó en aquel propuesto por Francis Chollet; la RNC Xception [3]. Este clasificador ha logrado obtener un 94% de *accuracy* en la clasificación del proyecto ImageNet, el cual consta de 1000 clases, lo que es una muestra clara de su capacidad de clasificación.

Sin entrar en detalle, este modelo está basado íntegramente en *capas de* convolución *separables por canal* como está descrito en Conceptos Preliminares. Por sobre esto, la arquitectura toma lo que se conocen como *skip connectors* los cuales brindan principalmente la ventaja de permitirle a las capas convolucionales de poder tomar características observadas por otras capas más allá de la capa inmediatamente anterior a ella, entre otras ventajas. A grandes rasgos la arquitectura está seccionada por una parte de entrada que consta de 2 capas convolucionales seguido de 6 capas de convolución separables por canal, cada una de ellas acompañada por capas de normalización de lote, la operación no lineal ReLU y la operación de submuestreo max pooling y skip connectors entre algunas de ellas. Esta parte está seguida por una etapa intermedia formada por 3 capas de convolución separables con un *skip connector* de principio a fin y se repite 8 veces. Finalmente la arquitectura en cierra con un *Global Average Pooling* que es considerado como un reemplazo a las capas completamente conectadas al final de las RNC, que permiten generar una correspondencia directa entre los mapas de características y las clases a detectar. La Figura 34 se muestra un esquema de la arquitectura.

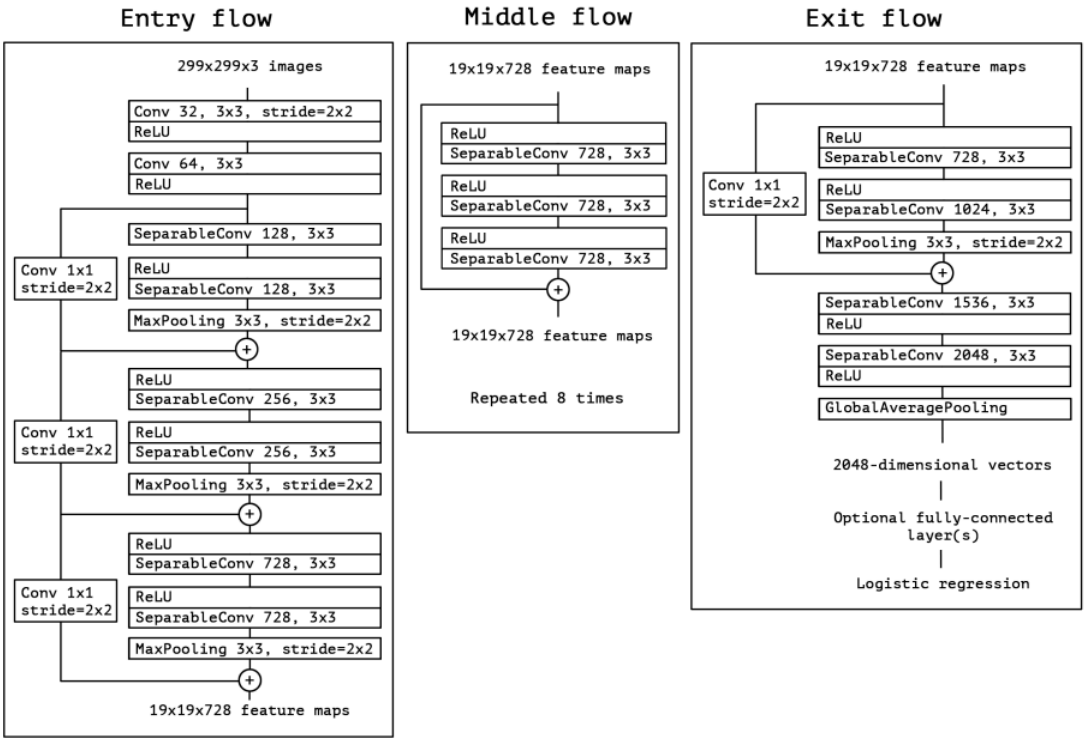


Figura 34. La arquitectura de la RNC Xception. Figura tomada de la publicación original.

En esta tesis hemos implementado la arquitectura de Xception y puede verse en <CÓDIGO>, en la siguiente subsección debatiremos sobre los resultados obtenidos con está arquitectura y sobre su adopción el el flujo propuesto.

Entrenamiento y Evaluación

En esta sección nos adentraremos en el proceso de entrenamiento realizado para la topología de la RNC descrita en la sección anterior. Se describirán las pruebas realizadas, los resultados obtenidos así como su interpretación.

Parámetros de la red

A continuación detallaremos los parámetros configurables del proceso de entrenamiento de la red:

- **Función de Pérdida:** Debido a que estamos trabajando en un problema de clasificación con múltiples clases, debemos poder cuantificar el error cometido tomando la diferencia entre dos distribuciones de probabilidades (la clasificación correcta frente a la esperada). Para ello hemos hecho uso de la función *Sparse Categorical Cross Entropy.* Si bien comparte la misma función de pérdida que *Categorical Cross Entropy*, la opción elegida admite que la distribución correcta no esté en codificación one-hot sino más bien a cada clase le corresponde un número.

- **Optimizador:** Para el algoritmo encargado de modificar los pesos de la red hemos optado por el optimizador Adam por su popular adopción en las RNC.

- **Métrica:** Como método de evaluación del rendimiento de la RNC hemos tomado las métricas *accuracy* y *F1*, la cual combina las métricas de precisión y el recall. Usamos esta métrica ya que los datos pueden no estar balanceados y una métrica como accuracy podría no estar representando verdaderamente el rendimiento de clasificación de las clases con menor cantidad de datos.

- **Tamaño del batch**. Debemos considerar que si el valor elegido es grande la red actualizará menos veces sus parámetros y ayuda a que el tiempo de entrenamiento disminuya y que se pueda reducir la posibilidad de overfitting pero a su vez puede suceder que la red no logre actualizar suficientemente sus pesos para lograr una correcta representación. Es por esto que hay que encontrar un balance con este parámetro. Además se tomó en cuenta la restricción de recursos para realizar el entrenamiento para tomar un valor de bath de 16 imágenes.

El proceso de entrenamiento y evolución requiere de un dataset de imágenes que consista de tres partes; una sección establecida para el entrenamiento en sí (el modelo aprenderá sobre esta parte), otra sección para la validación del modelo (el modelo usará ocasionalmente esta parte para evaluar su rendimiento de manera no sesgada pero nunca aprenderá de ella) y finalmente una reserva de imágenes la cual será usada para evaluar el modelo (solamente usada al finalizar el entrenamiento para evaluar el aprendizaje realizado).

Si bien no hay una regla estricta en Machine Learning sobre cómo dividir un dataset en secciones, una buena heurística resulta guiarse por la cantidad de imágenes dispuestas. Como en nuestro caso contamos con 6237 imágenes de galaxias, optamos por utilizar el 10% tanto para validación como para evaluación, y el 80% restante para el entrenamiento.

Como explicamos anteriormente, el dataset (tanto con la primera o ambas secciones) obtenido luego de ser reetiquetado resultó en una cantidad de imágenes por clase desbalanceada. Ante esta situación es sugerido balancear los datos, pero optamos por tomar las opciones a nuestro alcance (limitadas por los recursos disponibles) y probamos con 5 variaciones para quedarnos con aquella que mejor resultados logre.

A continuación mostraremos los resultados obtenidos con cada una de ellas utilizando la red descrita como prueba de concepto durante 12 épocas y luego haremos un análisis sobre ello. Acompañando los resultados se encuentra la cantidad de datos final utilizada para cada variación, recordamos que esta es la suma total y sobre ella se realizó la división hecha para entrenamiento, validación y evaluación.

1. Utilizar una cantidad de imágenes desbalanceadas por clase solamente con la primera sección de imágenes de la observación DECaLS.

Cantidad de imágenes por clase:

1268 De canto, 970 Elípticas, 684 En fusión, 315 Espirales

	precision	recall	f1-score	support
de_canto	0.78	0.77	0.77	126
eliptica	0.65	0.76	0.70	97
en_fusion	0.43	0.37	0.40	67
espiral	0.34	0.29	0.31	31
accuracy			0.63	321
macro avg	0.55	0.55	0.55	321
weighted avg	0.62	0.64	0.63	321

2. Hacer uso de la técnica de Sub Muestreo en la primera sección, es decir, eliminando imágenes de las clases que más imágenes tienen para equiparar con la que menos tiene.

Cantidad de imágenes por clase:

315 *De canto,* 315 *Elípticas,* 315 *En fusión,* 315 *Espirales*

	precision	recall	f1-score	support
de_canto	0.61	0.65	0.63	31
eliptica	0.58	0.62	0.60	31
en_fusion	0.63	0.65	0.63	31
espiral	0.62	0.52	0.56	31
accuracy			0.61	124
macro avg	0.61	0.61	0.61	124
weighted avg	0.61	0.61	0.61	124

3. Utilizar una cantidad de imágenes desbalanceadas por clase pero con ambas partes del dataset descargadas.

Cantidad de imágenes por clase:

2489 *De canto,* 1902 *Elípticas,* 1254 *En fusión,* 592 *Espirales*

	precision	recall	f1-score	support
de_canto	0.83	0.60	0.70	248
eliptica	0.74	0.76	0.75	190
en_fusion	0.42	0.80	0.56	125
espiral	0.32	0.39	0.35	59
accuracy			0.66	622
macro avg	0.58	0.64	0.59	622
weighted avg	0.68	0.67	0.65	622

4. Hacer uso de la técnica de Sub Muestreo, pero con el límite inferior establecido en la cantidad de imágenes del tipo de galaxias que menos tiene usando ambas secciones.

Cantidad de imágenes por clase:

592 *De canto,* 592 *Elípticas,* 592 *En fusión,* 592 *Espirales*

	precision	recall	f1-score	support
de_canto	0.72	0.69	0.71	59
eliptica	0.67	0.68	0.67	59
en_fusion	0.67	0.69	0.68	59
espiral	0.69	0.68	0.68	59
accuracy			0.69	236
macro avg	0.69	0.69	0.69	236
weighted avg	0.69	0.69	0.69	236

5. Generar un parámetro de peso por clase para el entrenamiento haciendo uso del total de las imágenes descargadas.

Cantidad de imágenes por clase:

2489 *De canto,* 1902 *Elípticas,* 1254 *En fusión,* 592 *Espirales*

Pesos por clase para el entrenamiento

De Canto	0.627

Elípticas	1.024
En Fusión	1.554
Espirales	3.292

	precision	recall	f1-score	support
de_canto	0.83	0.56	0.67	248
eliptica	0.71	0.76	0.74	190
en_fusion	0.45	0.80	0.57	125
espiral	0.38	0.51	0.43	59
accuracy			0.65	622
macro avg	0.59	0.66	0.60	622
weighted avg	0.67	0.67	0.65	622

En las tablas de rendimiento hallamos los valores de las métricas de evaluación *precisión*, *recall* y *f1* para cada clase y como última columna bajo el título de *support* la cantidad de imágenes por clase con la cual se ha realizado la evaluación del modelo entrenado bajo esa variación de datos. En las últimas filas encontramos las mediciones de accuracy, el promedio y el promedio ponderado de cada métrica.

Con el valor de estas métricas tenemos suficiente información para conocer qué acercamiento tomar para entrenar la red. Por sobre esto, cuando tenemos un dataset balanceado podemos usar distintas métricas casi indiscriminadamente, a diferencia de cuando no está balanceado dado que tenemos que usar una métrica que trate a todas las clases por igual, como lo es macro o micro *f1*. Es por esto que finalmente evaluaremos los diferentes entrenamientos con esta última métrica.

Comenzando con la variante (1), notamos que los valores de *f1* por clase tienen una desigualdad considerable, esto puede correlacionarse directamente con el desbalance de datos disponibles por clase. El promedio macro de la métrica de *f1* no pudo lograr un valor bueno debido a la baja capacidad de clasificación para la clase espiral y galaxias en fusión al haber aprendido más características de los tipos de galaxias que más imágenes contenían. En la variante (2), la cantidad de imágenes balanceada logra que el modelo pueda elevar e igualar los valores de *f1*

por clase, a costa de bajar la capacidad de reconocimiento de las galaxias que más datos tenía en la variación previa. En la variante (3) hemos hecho uso del total del dataset disponible, pero advertimos que el modelo replica la capacidad de clasificación de la variante (1), ya que aproximadamente la proporción de datos por clase del total se mantuvo y los valores de *f1* mantienen la misma desigualdad. En la variante (4), si bien utilizamos el mismo acercamiento que la variante (2), podemos señalar que se obtuvieron mejores valores de *f1* en todas las clases y se puede atribuir a que el modelo ha sido capaz de captar mayores características de cada galaxia al poseer más datos pero manteniendo una igualdad en el valor de *f1* por clase. Finalmente, en la última variación (5) se han aplicado pesos para poder determinar la influencia de una mala clasificación de una clase en la optimización de la pérdida durante el entrenamiento, pero notamos que no ha sido capaz de solventar el rendimiento de la clasificación por clase.

En conclusión, podemos decir que el mejor rendimiento del modelo se obtuvo al entrenarlo con la mayor cantidad de datos balanceados, esto es, usando la técnica de sub-muestreo con ambas secciones del dataset correspondiente a la variante (4). Esto además se puede confirmar mirando el valor del promedio macro de la métrica *f1* cuyo valor es *0.69* y es el más elevado de todas las demás variaciones.

Si bien dentro de las variaciones de las accesibles hemos obtenido la que mejor desempeño logró en la arquitectura de prueba de concepto, no quiere decir que el rendimiento del modelo sea bueno, pero nos encamina a saber cuál variación de datos tomar para entrenar el modelo a base de la arquitectura de Xception mencionado en la sección anterior, el cual requiere mayor carga para entrenar y será el que finalmente formará parte en el flujo de clasificación de las regiones de interés.

	precision	recall	f1-score	support
edge_on	0.83	0.84	0.83	59
elliptic	0.83	0.81	0.82	59
merger	0.82	0.80	0.81	59
spiral	0.77	0.80	0.78	59
accuracy			0.81	236
macro avg	0.81	0.81	0.81	236
weighted avg	0.81	0.81	0.81	236

Figura 34. Reporte de clasificación del modelo desarrollado a partir de la arquitectura Xception entrenado bajo la configuración de datos (4).

Los resultados obtenidos de distintos entrenamientos utilizando 24 épocas de entrenamiento, muestran un valor de promedio accuracy de 97% y 87% de accuracy en el proceso de validación del modelo como se puede apreciar en la Figura 35. Por otro lado, si miramos la capacidad de extrapolar la clasificación del modelo a imágenes nunca vistas fuera de la validación, vemos en la Figura 34 que el modelo reduce su accuracy a un valor del 81% lo que puede indicar que el modelo haya hecho un sobreajuste, de todas maneras y en comparación con el resultado obtenido de la prueba de concepto, podemos concluir que ha habido una notable mejora.

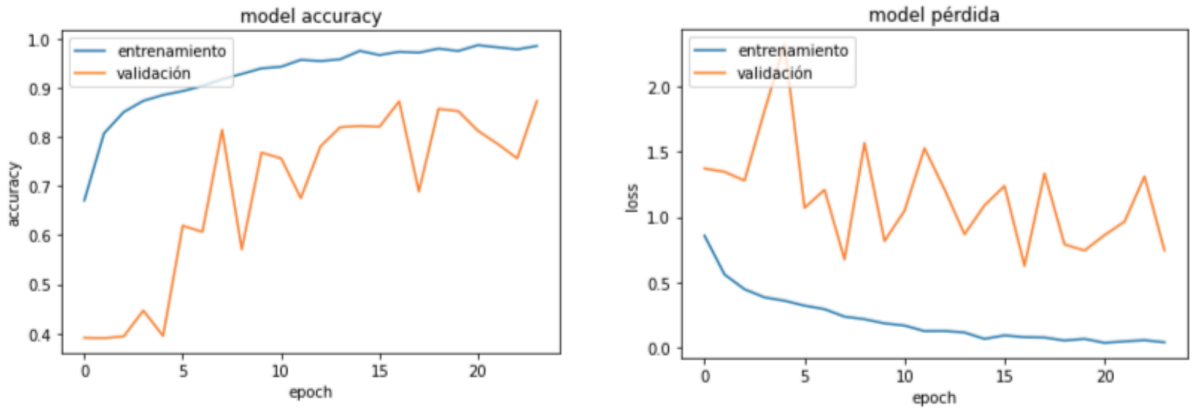


Figura 35. Progreso del entrenamiento de la RNC Xception sobre el dataset de la SDSS con variación de datos número 4.

Para finalizar la sección daremos un resumen del recorrido que realiza la RNC entrenada en el proceso de clasificación de galaxias. Comenzando por el preprocesamiento de la imagen pensándola como una matriz (tensor de profundidad 1), se recorta una sección central, se escala para encontrar una dimensión adecuada para las siguientes capas y se realiza la normalización de los valores de los píxeles. A continuación el tensor de entrada es procesado por las capas de convolución convencionales y separables a las cuales las acompaña la función de activación y la operación de max pooling. Cada operación de convolución genera un mapa de características de la imagen por kernel aplicado el cual resalta un determinado aspecto de la galaxia, como podemos ver en la Figura 36, donde se presentan diversos mapas de características generados por las capas de convolución. En ellos se puede apreciar la diferencia entre las características obtenidas de la galaxia, como sus brazos, el centro o el cúmulo globular acompañándolo.

Sobre esto, por cada tipo de morfológico, en la Figura 36 cada columna representa una capa convolucional y contiene dos mapas de características generados por ella. Las columnas se muestran según el orden de aparición de la capa que representa y, si tomamos únicamente las capas de convolución de la red, se corresponden con la capa 4, 11, 21, 80 y 121. Como última columna se tomó una muestra de mapas de características de una capa de convolución en una etapa más tardía de la red para poder ilustrar la capacidad de reducción de información de las capas de pooling. Por último cabe destacar que las capas mencionadas generan específicamente tensores (mapa de característica de ancho, alto y profundidad) de 62x62x64, 62x62x128, 31x31x256, 8x8x728 y 4x4x1024 respectivamente. Por ejemplo, la última columna muestra dos mapas de características ambos siendo una matriz de 4x4 píxeles de ancho y alto y se han generado 1024 de ellos.

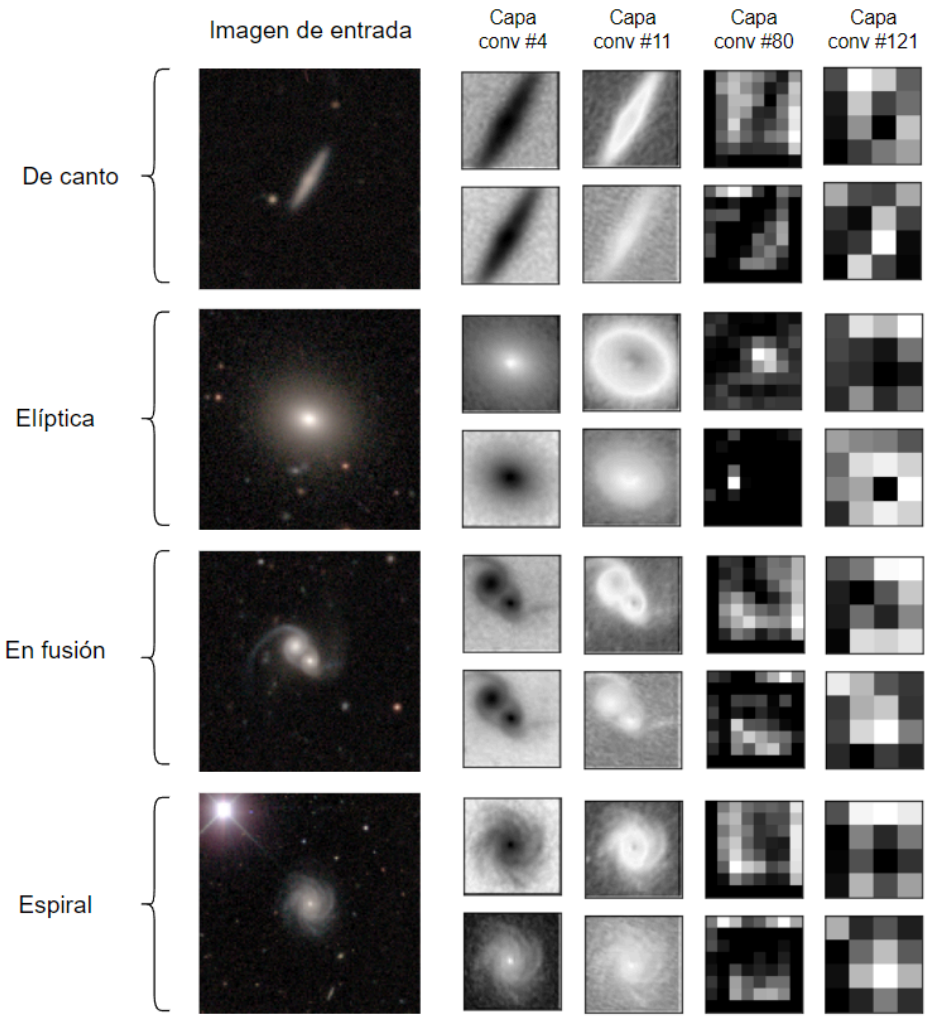


Figura 36. Muestras de mapas de características generados por la RNC a partir de una imágen de cada tipo de clase morfológico seleccionado.

Etapa 3 - Generación del resultado final

Esta etapa compone la parte final del flujo propuesto para detectar y clasificar las galaxias dentro de una región del cielo. Para llegar a este punto del proceso, el usuario inicialmente consultó por una zona del cielo, sobre la cual hemos obtenido y procesado imágenes para obtener las regiones de interés en la zona consultada. Una vez computadas las regiones de interés, realizamos una clasificación de cada una utilizando una Red Neuronal Convolucional, con el objetivo de identificar las galaxias presentes junto con el grupo morfológico de cada una.

En esta sección explicaremos la manera en la cual se presenta el resultado al usuario, como respuesta a su consulta. La devolución al usuario se hará mediante una representación visual que consiste en una imagen de la región del cielo astronómico, junto con un marcado de las galaxias identificadas y sus respectivas clasificaciones.

Interpretación de las clasificaciones

Para lograr esto, debemos recopilar la información generada por las etapas previas, esto es, las regiones de interés y la clasificación obtenida para cada una de ellas a fin de poder reconstruir la imagen. Cabe destacar que la clasificación obtenida de las galaxias está dada con un nivel de certeza, es decir, hay una probabilidad de que el contenido de una región de interés sea de una determinada galaxia. Un acercamiento inicial es tomar un umbral de confianza y solamente tomar las galaxias que lo superen, pero las predicciones generadas por la función softmax en la última capa de la red no representan la probabilidad mencionada y no deberíamos basarnos en ella para descartar. Este es un problema conocido en los últimos acercamientos de las RNC [13] y generan una confianza más elevada de lo que realmente es. En la Figura 37 podemos ver dos predicciones con una confianza mayor al 99% que ilustran el problema. Lo esperable es que el resultado de la RNC esté calibrado, esto es, si tenemos 100 predicciones y cada una con una confianza del 80% esperaríamos que 80 regiones de interés estén correctamente clasificadas. En esta tesis simplemente retornaremos el *argmax* del tensor resultante de la RNC luego de la función de activación *softmax* y notaremos como

oportunidad de mejora la calibración de la red, en donde Chuan Guo et. al. [16] han realizado investigación sobre el caso.

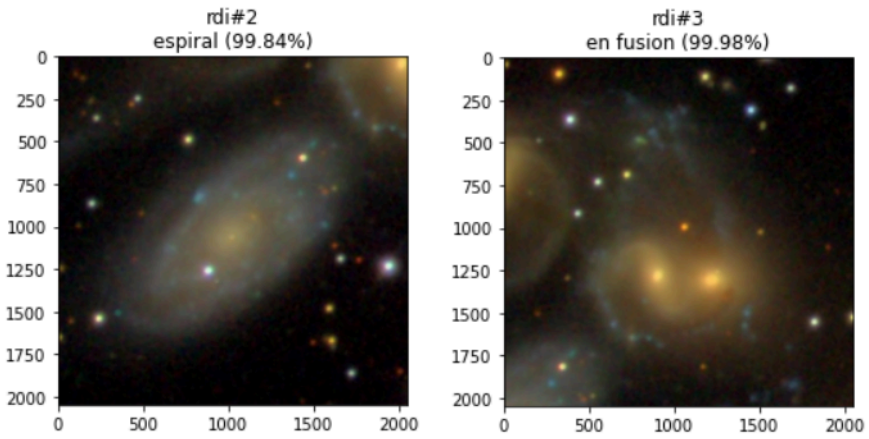


Figura 37. Ejemplo de clasificación de algunas regiones de interés del Quinteto de Stephan junto con el valor generado por la función de activación *softmax.*

Momentos

Por otro lado, a modo de poder descartar las regiones de interés no relevantes, es decir, que contenga un objeto astronómico distinto a una galaxia o algún defecto del telescopio y continuando con el filtro mencionado en la sección <SECCIÓN> a fin de poder descartar las regiones de interés no relevantes, hemos refinado el filtro mediante la herramienta de la SDSS. Este paso se debe a que el mismo estaba siendo flexible y, sumado a no poder establecer un umbral de confianza en las clasificaciones, se notó que el flujo estaba generando clasificaciones a regiones de interés no deseadas. Durante el análisis del problema se encontraron dos situaciones:

1. El radio de búsqueda de objetos cercanos era fijo y no variaba con el tamaño de la región de interés.
2. No siempre se encontraba un objeto donde en verdad había uno.

De (1) se genera el problema de que ante una región de interés chica se encuentren objetos no deseados por fuera de la misma y si alguno de ellos era una galaxia la región de interés era entendida como galaxia. De (2) las regiones de interés grandes podrían contener una galaxia extensa cuyo centro no se encontraba dentro del círculo ubicado precisamente en el centro de la región de interés con el radio establecido, por ende, la coordenada necesaria para ubicar la galaxia deseada quedaba fuera de la zona consultada. Para solventar ambos problemas, en primer lugar se estableció arbitrariamente una relación de tamaño de región de interés y el radio de búsqueda, esto logra que ante regiones de interés extensas se agrande la zona de consulta y viceversa. En segundo lugar, para aproximar de una mejor manera la ubicación del centro del objeto contenido y poder ubicar el círculo de consulta de una manera más estratégica, se tomó el concepto fundamental del procesamiento de imágenes conocido como *momentos de imágenes*. De manera resumida, los momentos son un conjunto de valores que permiten estimar la distribución de la ubicación de los píxeles tomando la intensidad que aportan a la imagen, o dicho de otro modo, un promedio ponderado de la intensidad de los píxeles. Más formalmente, podemos definir los momentos (i,j) de una imagen como

$$M_{ij} = \sum_{x} \sum_{y} x^i y^j I(x, y)$$

Donde (x,y) denotan la fila y columna de la posición del píxel e I(x,y) su intensidad. A partir de esta fórmula podemos obtener el centroide ponderado el cual nos dará la coordenada en píxeles del núcleo aproximado del objeto buscado. Esta propiedad se deriva de los momentos como describe la siguiente fórmula

$$\{\bar{x},\ \bar{y}\} = \left\{ \frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right\}$$

Con estos cambios propuestos se ha logrado mejorar la capacidad de búsqueda y filtrado por la herramienta de la SDSS. Como se puede ver en la Figura 38, el centroide ponderado estima de mejor manera la ubicación del punto donde debe centrarse la búsqueda de objetos astronómicos para retornar el o los objetos deseados.

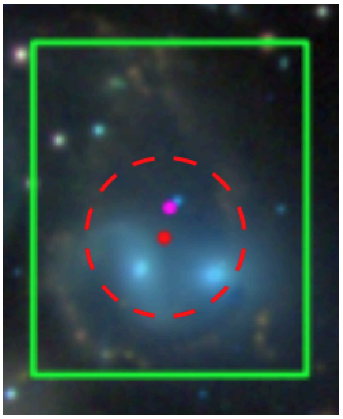


Figura 38. El rectángulo verde presenta una región de interés, el punto violeta su centro geométrico, el punto rojo el centroide ponderado y el círculo rojo a rayas el área de búsqueda mediante API, en este caso con radio 0.21 segundos de arco.

Resultado final

Finalmente, a fin de concluir el proceso, generamos la devolución al usuario para que pueda apreciar las galaxias identificadas. Para ello tomamos las regiones de interés que han pasado los filtros mencionados, junto con la clasificación obtenida por la RNC y construiremos la imagen de la zona del cielo celeste. El modo de visualización de las regiones identificadas será mediante una delimitación con un rectángulo enmarcando la zona y una etiqueta identificando la clasificación morfológica obtenida. En la Figura 39 se puede apreciar un ejemplo de la devolución realizada al usuario de la región del cielo celeste de interés del Quinteto de Stephan junto con las galaxias identificadas y su clasificación morfológica. Como se puede apreciar, cada clasificación se corresponde a su verdadero grupo morfológico, las galaxias *NGC 7318B* junto con *NGC 7318A* etiquetadas como galaxias en fusión al igual que la galaxia NGC 7319, la galaxia espiral *NGC 7320* clasificada como tal, y finalmente la galaxia *NGC 7317* perteneciente al grupo morfológico de elípticas.

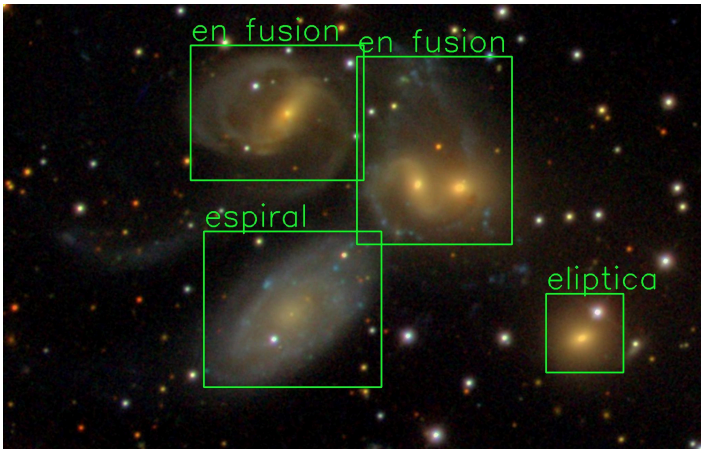


Figura 39. Resultado del uso de Redes Neuronales Artificiales para la detección y clasificación de galaxias.

Aplicación

En esta sección daremos un recorrido sobre la implementación de la herramienta para la inspección visual de una región del cielo astronómico. Recordamos que incluye el ingreso de coordenadas ecuatoriales delimitando la región consultada, la detección y posterior clasificación de las galaxias que allí se encuentren y finalmente la presentación del resultado final con las galaxias demarcadas y la clasificación obtenida para cada una de ellas.

Comenzaremos explicando el stack de tecnologías utilizado para implementar la herramienta, para ello hemos organizado la herramienta utilizando una arquitectura simple pero a la vez comúnmente utilizada; una interfaz y un servidor proveyendo una API. Para la implementación del primero se optó por NextJS, el cual es un framework creado en base a la librería de Javascript ReactJS. Su adopción se basó en la facilidad de implementación de la parte gráfica, el Server Side Rendering y la experiencia del autor en ella. Para el desarrollo del servidor, dado que se utilizó como lenguaje de programación Python para implementar el modelo clasificador debido a la disponibilidad de librerías de Machine Learning y procesamiento de imágenes para este lenguaje, se tomó como framework Flask utilizando solamente el sistema de rutas e implementación de APIs.

De parte de la interfaz presentada al usuario, inicialmente se presenta el nombre de la aplicación, y campos de texto para que el usuario ingrese ambas coordenadas delimitadoras en el sistema de coordenadas ecuatoriales (α, δ), luego un botón de confirmación el cual iniciará el flujo de búsqueda y clasificación. Hasta aquí corresponde la Etapa 1 del flujo propuesto y se puede observar un ejemplo en la Figura 40.

Figura 40. Ingreso de coordenadas ecuatoriales delimitadoras de la región a consultar.

Una vez ingresadas las coordenadas, se harán dos consultas HTTP GET, la primera corresponde a la obtención de una imagen de la región sin analizar y la segunda a realizar el procesamiento que retornará finalmente el resultado. Este último corresponde a la Etapa 2 del flujo propuesto. Mientras se realiza el análisis de la región, se presenta una barra de proceso indicando que está en curso el análisis. En la Figura 41 se puede apreciar una ejemplo de uso de la aplicación durante el análisis de la región.

Figura 41. Interfaz de la aplicación durante el análisis de la región.

Finalmente para la presentación del resultado final, se muestra lo ilustrado en la Figura 42 y corresponde a lo obtenido por la Etapa 3 del flujo. En este momento el usuario podrá ver la imagen de la región con las galaxias encontradas y clasificadas demarcadas por un rectángulo estarán acompañados por una etiqueta con la clasificación obtenida.

De su parte, el servidor provee dos rutas que hacen de interfaz al flujo clasificación y de esta manera independiente del cliente que lo solicite. La primera ruta corresponde a la obtención de una imagen de la región consultada delimitada por dos coordenadas ecuatoriales y la segunda ruta corresponde al procesamiento de una región delimitada con coordenadas ecuatoriales nuevamente. Cabe destacar que el procesamiento del flujo no es inmediato y por ende es conveniente mostrar un estado de carga al usuario.

Figura 42. Resultado de la herramienta al analizar el Quinteto de Stephan.

Discusión y Conclusiones

En este proyecto hemos desarrollado una herramienta capaz de obtener una imágen de una región delimitada por el usuario del cielo astronómico haciendo uso de herramientas disponibles públicamente, reconocer regiones de interés de la imagen que potencialmente contienen galaxias, filtrarlas según su tamaño y contenido, clasificar aquellas que contengan galaxias según su morfología para finalmente realizar una segmentación de la imagen y presentársela al usuario.

La extracción y propuesta de las regiones de interés es realizada mediante el procesamiento de la imagen, analizando los contornos formados por las partes más brillantes de la imagen. Se obtiene una imagen de las regiones filtradas mediante la misma herramienta que se utilizó para obtener la imagen inicial, a modo de poder obtener mayor resolución, y por lo tanto más información que el clasificador puede tomar. Se realiza un filtro de las regiones de interés mediante la API de SDSS donde se consulta por objetos cercanos a un punto de mayor intensidad de la región, donde se busca distinguir galaxias de otro tipo de objeto astronómico.

El modelo clasificador de las regiones propuestas está basado en una arquitectura de uno de los avances más recientes en clasificación de imágenes, conformada por capas de convolución separables, y es capaz de clasificar de manera eficiente con un f1-score promedio del 87%. Este modelo fue entrenado con imágenes clasificadas utilizando un método abierto al público (crowdsourcing), donde se obtuvieron características de las galaxias y en este proyecto hemos reetiquetado las galaxias tomando las características que identifican a los tipos de galaxias seleccionados. Hemos creado previamente una arquitectura de prueba de concepto la cual no solamente nos ayudó a diseñar el modelo final, sino que también permitió obtener la mejor variación de datos para entrenarlo.

Este proyecto comenzó con la exploración de datos astronómicos disponibles en diversos formatos (archivos FITS, espectros de diferentes bandas, entre otros) para los cuales se consideraron diversos modelos clasificadores, optando finalmente por utilizar imágenes y consecuentemente las RNC. Las cuales el autor estaba interesado en estudiar gracias a herramientas y conocimientos brindados por los estudios realizados para la Licenciatura en Ciencias de la

Computación; conocimiento de Inteligencia Artificial, abarcando desde la comprensión del propósito del área hasta el estudio de diferentes modelos diseñados dentro de ella (incluidos las RNA y RNC), junto con los detalles de los algoritmos en su implementación y el procesamiento de datos. A este conocimiento se le agregan patrones y estándares comprendidos en materias de Diseño y Desarrollo de Software, Ingeniería de Aplicaciones Web, Algoritmos y Complejidad, sin dejar de mencionar a las demás materias que comprenden la carrera que han aportado a la construcción de conocimiento fundamental para desarrollar este proyecto.

Este estudio ha intentado diseñar una plataforma a partir de fuentes gratuitas y disponibles en la web para facilitar la comprensión del Universo y no solo eso, sino que también se construyó el modelo clasificador a partir de información recolectada en solo un subconjunto de éste, permitiendo extrapolar el poder de clasificación a demás áreas. A su vez ha sido un camino tanto práctico como teórico que ha asistido a comprender el diseño y funcionamiento de las RNC logrando hacer una implementación concreta de una de ellas, junto con el estudio de procesamiento de datos, conceptos de astronomía, análisis de imágenes y diseño de aplicaciones web.

Creemos que la herramienta ha logrado cumplir con el objetivo principal del proyecto, poder brindar una forma de análisis e inspección de las galaxias de una región particular del cielo astronómico, junto con un estudio de su morfología dentro de una clasificación de 4 tipos. Es capaz de obtener imágenes de las regiones de interés de mayor calidad para utilizarlas en el modelo clasificador y ésta presenta una mejora respecto a otros acercamientos de segmentación que no pueden sacar provecho de ello. Cabe destacar que en el proceso de desarrollo de la misma nos hemos topado con diversos problemas y puntos de mejora. Por ejemplo, al momento de realizar el filtrado de las regiones de interés mediante la herramienta de la SDSS, nos encontramos que se obtenían objetos astronómicos más allá del solicitado, ahora el flujo propuesto es capaz de establecer un tamaño de búsqueda dinámico y realizar corrimiento del punto de búsqueda a un lugar más acertado. Por otro lado, de la prueba de concepto de la arquitectura del modelo RNC surgió el interés de conocer y partir desde los acercamientos actuales en clasificación de imágenes.

Si bien la aplicación provee una interfaz gráfica para interactuar y visualizar el resultado, la carga del cómputo del flujo se encuentra en un servidor que es independiente del cliente que lo acceda. Esto facilitará crear otras interfaces gráficas o incluso herramientas que automaticen la búsqueda de galaxias de tal forma en regiones del cielo.

Antes de finalizar nos parece necesario mencionar los aspectos del flujo que presentan una oportunidad de mejora en futuros estudios. En primer lugar, nos encontramos con que los modelos basados en RNC que poseen diversas capas ocultas presentan el problema de no estar calibrados, este es el caso de nuestro modelo empleado, idóneamente se buscarían formas de calibrarlo y presentar la confianza de las calificaciones al usuario. Por otro lado, muchas veces se presenta que una región de interés conteniendo un objeto distinto a una galaxia no es filtrado correctamente y se le asigna una clasificación morfológica correspondiente a una galaxia, futuros acercamientos podrían utilizar los diferentes canales de color para detectar solamente galaxias o incluso monitorear la distribución de los píxeles de la región a fin de identificarlas. Por sobre esto, si bien la intención del resultado es que el usuario pueda hacer una inspección visual mediante una imagen de la región de estudio, podría incluirse la opción de obtener las coordenadas de las galaxias halladas junto con la clasificación de cada una en un formato diferente (json, csv) y así admitir que otros programas puedan tomar estos valores y construir sobre ello. Otro punto de mejora reside en la separación de galaxias superpuestas, por ejemplo, la herramienta en el proceso de extracción de regiones de interés puede contemplar tanto a dos galaxias en conjunto fusionando como *en fusión* como a las dos o más partícipes de la fusión como galaxias *en fusión* separadas.

Por último, la herramienta presenta limitaciones, en primer lugar, se encuentra restringida al alcance de las observaciones de la SDSS de donde se toman la imágenes para el análisis, y segundo, si bien no es posible establecer una relación directa entre el tamaño de la región a estudiar y el tiempo requerido ya que depende de factores como la velocidad de descarga red, una región muy grande puede demorar tiempos de espera prolongados lo que puede presentar un impedimento para su uso de manera extensa.

Anexo

- Repositorio de los laboratorios referenciados en el documento https://github.com/EzeGmnz/Tesis

Bibliografía

[1] Sellwood JA, 2014, Reviews of Modern Physics.
[2] John, D. (2006). Astronomy: The definitive guide to the universe. Bath, UK: Parragon Publishing.
[3] F. Chollet, 2017, Xception: Deep Learning with Depth Wise Separable Convolutions.
[4] Yingying Wang, 2020, The Influence of the Activation Function in a Convolution Neural Network Model of Facial Expression Recognition.
[5] https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf
[6] John Duchi et. al. 2011, Adaptive Subgradient Methods for Online Learning and Stochastic Optimization.
[7] Diederik P. Kingma and Jimmy Lei Ba, 2015, Adam: a Method for Stochastic Optimization.
[8] Nour Eldeen M. Khalifa et. al. 2017, Deep Galaxy: Classification of Galaxies based on Deep Convolutional Neural Networks.
[9] Ross Girshick et. al, 2014, Ross Girshick, Rich feature hierarchies for accurate object detection and semantic segmentation.
[10] Ross Girshick, 2015, Fast R-CNN.
[11] Satoshi Suzuki et. al., 1985, Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing.
[12] Mike Walmsley et. al., 2019, Galaxy Zoo DECaLS: Detailed Visual Morphology Measurements from Volunteers and Deep Learning for 314,000 Galaxies.
[13] Chuan Guo et. al., 2017, On Calibration of Modern Neural Networks.