UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
Departamento de Ingeniería en Informática

Clasificación de objetos astronómicos mediante sus primeras alertas utilizando aprendizaje profundo

Javier Andrés Arredondo Contreras

Profesor guía: Pablo Román Asenjo
Profesor co-guía: Francisco Förster Burón

Tesis para optar al grado académico de Magíster de Ingeniería en Informática y al título de Ingeniero Civil en Informática

Santiago – Chile
2021

RESUMEN

Las redes neuronales convolucionales han sido una de las técnicas más utilizadas para resolver el problema de clasificación de datos espacialmente correlacionados, como lo son las imágenes. En el área de la astronomía, se han utilizado para clasificar pequeñas imágenes de objetos astronómicos con un buen desempeño. Una red convolucional exitosa, corresponde al clasificador *stamp classifier* de ALeRCE, el cual utiliza una tripleta de imágenes de la primera observación para predecir, sin embargo no utiliza observaciones posteriores que podrían mejorar su rendimiento. En esta tesis, se propone un nuevo modelo de clasificación de pares de alertas del telescopio Zwicky Transient Facility (ZTF), es decir, utilizar las dos primeras observaciones. De esta forma se utiliza el contexto de las imágenes y además se agrega un componente temporal al modelo. Este enfoque permite funcionar como un clasificador temprano de objetos astronómicos, por lo que se evita realizar cálculos de curvas de luz u otro tipo de cálculo costoso. Los resultados muestran que al utilizar las dos primeras alertas y la diferencia de días en que se han obtenido cada una, sugiere una mejora respecto al modelo *stamp classifier* de ALeRCE. Concretamente se proponen dos arquitecturas basadas en el modelo *stamp classifier*, denominadas *stacked two stamp classifier* y *separated two stamp classifier*, que tienen un *accuracy* de 0.937 ± 0.003 y 0.932 ± 0.006 respectivamente, frente al 0.923 ± 0.005 de *accuracy* del modelo *stamp classifier* en un conjunto balanceado de prueba. Al realizar una prueba estadística de Welch entre las arquitecturas propuestas en esta tesis y la red *stamp classifier*, la evidencia indica diferencias significativas, siendo los modelos que utilizan las primeras dos alertas mejores que el modelo *stamp classifier*.

Otra contribución de este trabajo, ha sido la conformación de un conjunto de datos de las primeras dos alertas con etiquetas de ALeRCE, donde se propone una estratificación a partir de las etiquetas, además se conforma otro conjunto de datos no etiquetado para evaluar el comportamiento de los modelos propuestos.

Palabras Claves: astroinformática; clasificación; aprendizaje profundo; redes convolucionales; grandes volúmenes de datos

ABSTRACT

Convolutional neural networks have been one of the most widely used techniques to solve the problem of spatially correlated data classification, such as images. In the area of astronomy, they have been used to classify small images of astronomical objects with good performance. A successful convolutional network corresponds to ALeRCE's stamp classifier, which uses a triplet of images from the first observation to predict, however it does not use subsequent observations that could improve its performance. In this thesis, a new model for the classification of alert pairs of the Zwicky Transient Facility (ZTF) telescope is proposed, that is, to use the first two observations. In this way, the context of the images is used and also a temporary component is added to the model. This approach allows it to function as an early classifier for astronomical objects, thus avoiding light curve calculations or other costly calculations. The results show that when using the first two alerts and the difference in days in which each one was obtained, it suggests an improvement with respect to the ALeRCE stamp classifier model. Specifically, two architectures are proposed based on the stamp classifier model, called stacked two stamp classifier and separated two stamp classifier, which have a accuracy of 0.937 ± 0.003 and 0.932 ± 0.006 respectively, compared to the accuracy 0.923 ± 0.005 of the stamp classifier model in a balanced test set. When performing a statistical Welch test between the architectures proposed in this thesis and the stamp classifier network, the evidence indicates significant differences, with the models using the first two alerts being better than the stamp classifier model.

Another contribution of this work has been the creation of a data set of the first two alerts with ALeRCE labels, where a stratification is proposed based on the labels, in addition, another unlabeled data set is formed to evaluate the behavior of the proposed models.

Keywords: astroinformatics; classification; deep learning; convolutional neural network; big data

Dedicado especialmente a mi madre.

AGRADECIMIENTOS

Agradezco en primer lugar a mi madre Cristina Contreras por enseñarme a ser una buena persona en la vida y por dar todo y más por mí, gracias por darme la oportunidad de estudiar. Sin tu apoyo no creo que haya podido ser lo que soy hoy. También agradecer a cada uno de mis familiares, que más de una vez me dieron una palabra de aliento para cerrar con éxito este período universitario. De verdad muchas gracias.

También quiero agradecer a Paula Verdugo, por soportarme todo este tiempo de estrés y acompañarme en este último trayecto universitario. Gracias por preocuparte por mí día a día. Te amo.

Gracias a los profesores y profesoras de la universidad, más de alguna vez tuve charlas con ellos en los pasillos del DIINF y es impactante como una simple conversación te puede motivar y enseñar más que una cátedra. Quiero agradecer especialmente a Max Chacón y Felipe Bello por creer en mí, darme oportunidades que posiblemente eran un riesgo, gracias por ser mis senseis.

Al equipo de ALeRCE y al Instituto Milenio de Astrofísica, que han sido mis compañeros y compañeras por más de dos años, de verdad que me han enseñado y capacitado para ser el profesional que soy. Aún no asimilo ser parte de un proyecto de categoría mundial. Gracias Francisco Förster y Guillermo Cabrera por aportar en mi formación profesional y por la oportunidad. A Camilo Valenzuela, Diego Rodríguez, Esteban Reyes, Ignacio Reyes y Ernesto Castillo por la paciencia, chistes y buena onda. Al equipo de astronomía por compartir su conocimiento conmigo y enseñarme cosas desconocidas para mí sobre el universo, en especial a Paula Sánchez, por ser una profesional admirable y por las buena disposición a ayudarme.

A los amigos y amigas que hice en la universidad: Miguel Cárcamo, Natalia Guzmán, Juan Pablo Martínez, Cristian Espinoza, Shalini Ramchandani, Héctor Pérez, Diego Mellis, Andrés Muñoz, Carlos Pérez, Ignacio Ibáñez, Sandra Hernández, Nicolás Alarcón y Luis Migryk. En verdad a cada persona que me acompaño en la vida universitaria, gracias por las risas, estudios, juegos, foros y buena onda.

Agradezco además a la amiga de la vida, Sofía Díaz. Gracias por los consejos, ánimos, conversaciones y amistad incondicional.

También a mi guía en esta tesis, Pablo Román. Gracias por jugársela conmigo, confiar en mí, la buena disposición, darme ánimos y por gestionar máquinas para desarrollar este proyecto con éxito.

Y no puedo dejar atrás a mis compañeros perrunos, a la Chipi que me alcanzó a acompañar en mis primeros años de universidad, ahora estás en el cielo de perritos, gracias por marcar presencia cuando me quedaba estudiando hasta tarde. Y el Baily, que me ha acompañado estas últimas noches de trasnoche sin falta y que ha servido de modelo para algunas figuras de la tesis.

TABLA DE CONTENIDO

1 Introducción **1**
- 1.1 Antecedentes y motivación . . . 1
- 1.2 Descripción del problema . . . 3
- 1.3 Solución propuesta . . . 5
- 1.4 Objetivos y alcance del proyecto . . . 6
  - 1.4.1 Objetivo general . . . 6
  - 1.4.2 Objetivos específicos . . . 6
  - 1.4.3 Alcances . . . 6
- 1.5 Metodología y herramientas utilizadas . . . 7
  - 1.5.1 Metodología . . . 7
  - 1.5.2 Software . . . 8
  - 1.5.3 Hardware . . . 10
- 1.6 Organización del documento . . . 11

2 Marco teórico y revisión de la literatura **12**
- 2.1 Astronomía . . . 12
  - 2.1.1 Conceptos básicos . . . 12
  - 2.1.2 Objetos astronómicos de interés . . . 14
  - 2.1.3 *Astronomical surveys* . . . 15
- 2.2 *Zwicky Transient Facility* . . . 17
  - 2.2.1 Funcionamiento de la cámara . . . 18
  - 2.2.2 Procesamiento interno de ZTF . . . 21
  - 2.2.3 Extracción de eventos y generación de alertas . . . 22
  - 2.2.4 Distribución de datos públicos . . . 23
- 2.3 Redes neuronales artificiales . . . 24
  - 2.3.1 Funciones de activación . . . 26
  - 2.3.2 Ajuste de una red neuronal . . . 27
  - 2.3.3 Redes neuronales profundas . . . 28
  - 2.3.4 Redes neuronales convolucionales . . . 29
  - 2.3.5 Capas de regularización . . . 34
  - 2.3.6 Métricas en el problema de clasificación . . . 34
- 2.4 Estado del arte: Clasificación utilizando imágenes de *astronomical surveys* . . . 35

3 Conjunto de datos **40**
- 3.1 Descripción y características de los datos . . . 40
  - 3.1.1 Contenido de una alerta . . . 40
  - 3.1.2 Metadatos . . . 41
  - 3.1.3 Estampillas . . . 43
  - 3.1.4 Etiquetas . . . 43
- 3.2 Obtención de los datos . . . 45
- 3.3 Análisis de los datos . . . 46
- 3.4 Conjunto de entrenamiento, validación y prueba . . . 51
- 3.5 Preprocesamiento . . . 53
  - 3.5.1 Estampillas . . . 53
  - 3.5.2 Metadatos . . . 54
- 3.6 Serialización en TFRecords . . . 54

4 *Two Stamp Classifier* **56**
- 4.1 Arquitecturas . . . 56
  - 4.1.1 *Stacked two stamp classifier* . . . 58

4.1.2 *Separated two stamp classifier* . . . 59

5 Metodología y experimentos **61**

6 Resultados **63**
6.1 *Stamp classifier* . . . 63
6.2 *Stacked two tamp classifier* . . . 64
6.3 *Separated two stamp classifier* . . . 65
6.4 Comparación entre modelos . . . 66
6.5 Tiempos de inferencia del modelo . . . 67
6.6 Desempeño en conjunto no etiquetado . . . 69
6.7 Importancia de metadatos . . . 71
6.8 *Two stamp classifier* sin metadatos . . . 73
6.9 *Two stamp classifier* con estampillas y *deltajd* . . . 73
6.10 Posibilidad de agregar más clases . . . 74
6.11 Discusión . . . 75

7 Conclusiones **78**
7.1 Solución al problema planteado . . . 78
7.2 Análisis de los objetivos . . . 79
7.3 Aciertos y desaciertos . . . 80
7.4 Trabajo futuro . . . 80

Glosario **82**

Referencias bibliográficas **83**

Anexos **88**

A Taxonomía de ALeRCE **88**

B Algoritmo de corte y balanceo de sub-etiquetas **89**

C Análisis del resto de los metadatos **90**

ÍNDICE DE TABLAS

Tabla 2.1 Cuadro comparativo de artículos revisados 39

Tabla 3.1 Contenidos generales de las alertas de ZTF 40
Tabla 3.2 Lista de metadatos utilizados para el desarrollo de *stamp classifier* de ALeRCE 42
Tabla 3.3 Etiquetas de ALeRCE 44
Tabla 3.4 Muestra de archivo CSV de etiquetas 45
Tabla 3.5 Conjunto de entrenamiento, validación y prueba 53

Tabla 4.1 Bloque de convoluciones que realiza DeepHiTS 57
Tabla 4.2 Arquitectura de red neuronal de *stamp classifier* 57
Tabla 4.3 Arquitectura de red neuronal *stacked two stamp classifier* 59
Tabla 4.4 Arquitectura de red neuronal *separated two stamp classifier* 60

Tabla 6.1 Mejores 10 *stamp classifier* en *accuracy* de validación 63
Tabla 6.2 Mejores 10 *stacked two stamp classifier* en *accuracy* de validación 64
Tabla 6.3 Mejores 10 *separated two stamp classifier* en *accuracy* de validación 65
Tabla 6.4 Resumen mejores modelos 67

ÍNDICE DE ILUSTRACIONES

Figura 1.1 Cantidad de supernovas públicas y confirmadas en TNS por año 3
Figura 1.2 Cantidad de supernovas clasificadas y reportadas por ALeRCE que han sido confirmadas espectroscópicamente 4

Figura 2.1 Magnitud aparente y magnitud absoluta. 13
Figura 2.2 Curva de luz de objeto ZTF17aaaecvk. 13
Figura 2.3 Diagrama simplificado del funcionamiento de los *astronomical surveys* . . . 17
Figura 2.4 Longitud de onda de los filtros de ZTF 18
Figura 2.5 Campo de visión de la cámara ZTF comparado con el de otras cámaras de gran tamaño. 18
Figura 2.6 Campo de visión versus área de recolección de luz para una selección de telescopios ópticos terrestres y espaciales actualmente operativos o planificados . 19
Figura 2.7 Imagen de un CCD de 2/3 pulgadas 19
Figura 2.8 Cámara y filtro de ZTF. 20
Figura 2.9 Galaxia Espiral M81 capturada por ZTF. 21
Figura 2.10 Imágenes de *science*, *reference* y *difference* capturadas por ZTF. 23
Figura 2.11 Recorte de ejemplo de objeto ZTF20aaelulu captada por ZTF 24
Figura 2.12 Representación gráfica de un perceptrón 25
Figura 2.13 Representación gráfica de una MLP. 25
Figura 2.14 Ejemplo de arquitectura de una red convolucional. 30
Figura 2.15 Operación de convolución en una matriz. 31
Figura 2.16 Ejemplo de imagen digital. 32
Figura 2.17 Ejemplo de convolución con *kernel* de 10×10 diagonal. 32
Figura 2.18 Ejemplo de convolución con *kernel* de 10×10 vertical. 33
Figura 2.19 Ejemplo de *max pooling* de 2×2. 33
Figura 2.20 Arquitectura de Deep-HiTS 37
Figura 2.21 Arquitectura de *stamp classifier* 37

Figura 3.1 Distribución de clases en el conjunto de datos 47
Figura 3.2 Distribución de sub-clases en el conjunto de datos 48
Figura 3.3 Histograma de *deltajd* por clase 49
Figura 3.4 Histograma de *elong* por clase 49
Figura 3.5 Puntuación de calidad de *real bogus* entregado por ZTF para cada alerta . . 50
Figura 3.6 Cambio de filtros de las dos primeras alertas 50
Figura 3.7 Muestra de estampillas por clase 51
Figura 3.8 Sub-clases del conjunto de datos 52
Figura 3.9 Sub-clases limitadas en 30.000 52

Figura 4.1 Capas convolucionales de *stamp classifier* 56
Figura 4.2 *Stacked two stamp classiffier* 58
Figura 4.3 *Separated two stamp classiffier* 60

Figura 6.1 Matriz de confusión para mejor *stamp classifier* 64
Figura 6.2 Matriz de confusión para mejor *stacked two stamp classifier* 65
Figura 6.3 Matriz de confusión para mejor *separated two stamp classifier* 66
Figura 6.4 Tiempo de inferencia y *accuracy* según arquitectura 68
Figura 6.5 Tiempo de inferencia según tamaño de *batch* 69
Figura 6.6 Distribución espacial para objetos en el conjunto no etiquetado por clase predicha 70

Figura 6.7 Matriz de confusión de *random forest* que solo usa metadatos en conjunto de prueba 71
Figura 6.8 Importancia de metadatos según *random forest* 72
Figura 6.9 *Stacked two stamp classifier* sin metadatos 73
Figura 6.10 *Stacked two stamp classifier* con estampillas y *deltajd* 74
Figura 6.11 *Stacked two stamp classifier* prediciendo 6 clases 75
Figura 6.12 Tamaño de conjunto de alertas diarias de ZTF 76

Figura A.1 Taxonomía de ALeRCE 88

Figura C.1 Primera porción de metadatos utilizados 90
Figura C.2 Segunda porción de metadatos utilizados 91
Figura C.3 Tercera porción de metadatos utilizados 92
Figura C.4 Cuarta porción de metadatos utilizados 93

ÍNDICE DE ALGORITMOS

Algoritmo B.1 Algoritmo de corte de etiquetas. 89

CAPÍTULO 1. INTRODUCCIÓN

1.1 ANTECEDENTES Y MOTIVACIÓN

En los últimos años el tratamiento de los datos en astronomía ha cambiado de paradigma. Un ejemplo de ello son los *surveys*[1], herramientas de carácter informático para capturar, procesar y distribuir grandes muestras de objetos astrofísicos observados en el universo, en otras palabras, se ha digitalizado el proceso de observación en el campo de la astronomía óptica.

La operación de recolección de datos es de carácter repetitivo, es decir, se observa constantemente las mismas regiones del cielo, con el objetivo de detectar y alertar movimientos y/o variaciones de luminosidad, produciendo una gran cantidad de datos en función del tiempo. La recolección de estos datos facilita el estudio de varios tipos de objetos en grandes zonas del universo, permitiendo investigar distintas propiedades de los astros en distintas longitudes de onda (Mickaelian, 2012).

Según Tyson (2019) se extraerán cientos de petabytes de datos complejos con alta dimensionalidad que permitirán estudiar y explicar diversos fenómenos que varían en el tiempo.

Algunos *surveys* son capaces de transmitir en tiempo real los datos recolectados en paquetes llamados alertas astronómicas. Por lo tanto, una rápida inspección y análisis de las alertas sería un aporte a la comunidad astronómica, ya que se podrían descubrir objetos interesantes o con comportamientos llamativos que fueron avistados recientemente para diversos casos científicos (Graham et al., 2019).

Los *astronomical surveys* se encuentran en diferentes partes del mundo, como por ejemplo Asteroid Terrestrial-Impact Last Alert (ATLAS) en Hawaii (Tonry et al., 2018), Hyper Suprime-Cam Subaru Strategic Program (HSC-SSP) en Japón (Aihara et al., 2018), Zwicky Transient Facility (ZTF) en Estados Unidos (Bellm et al., 2018), entre otros.

En los próximos años, entrará en funcionamiento el Vera C. Rubin Observatory y su telescopio Legacy Survey of Space and Time (LSST) en Chile, el telescopio óptico más grande jamás construido, con una cámara de 3200 mega píxeles y un espejo de 8 metros de diámetro, el cual será capaz de captar 20 terabytes de datos por noche (Brough et al., 2020).

Un *broker* astronómico es una organización que es capaz de procesar en tiempo real alertas de los *astronomical surveys*. Los *brokers* enriquecen las alertas a partir de procesamientos internos. El objetivo principal es proveer una clasificación rápida de los datos que son transmitidos en tiempo real.

[1] También conocidos como *astronomical surveys*. Corresponden a instrumentos que son capaces de monitorear objetos astronómicos, generalmente son telescopios terrestres o satélites.

En esta tesis se colabora con Automatic Learning for the Rapid Classification of Events (ALeRCE), que es una colaboración inter-institucional liderada en Chile y cuenta con apoyo de instituciones del extranjero. ALeRCE ha estado trabajando con los datos en tiempo real de ZTF y ha sido uno de los primeros *brokers* en aplicar métodos de aprendizaje automático para la clasificación de objetos astronómicos y detección de *outliers* (Förster et al., 2020).

Existen otros *brokers* que han destacado en el área y cubren diferentes casos de uso y objetivos científicos. Los más conocidos son Arizona-NOAO Temporal Analysis and Response to Events System (ANTARES) en Estados Unidos (Narayan et al., 2018), Fink en Francia (Möller et al., 2020) y Lasair en Reino Unido (Smith, 2019).

Un *pipeline* en el desarrollo de *software* es una técnica que permite procesar en simultáneo un flujo de datos secuencial. Actualmente ALeRCE utiliza éste método para la ingesta datos en tiempo real de ZTF (Förster et al., 2020) .

El *pipeline* de ALeRCE abarca la ingesta, almacenamiento, cómputos que enriquecen las alertas astronómicas y clasificación mediante aprendizaje automático (Förster et al., 2020).

Existen dos fases de clasificación en el *pipeline* de ALeRCE: (a) clasificación temprana mediante la primera alerta a través de una red convolucional. Para esto se utilizan recortes de imágenes astronómicas contenidas en una alerta (Carrasco-Davis et al., 2020) y (b) un clasificador tardío que utiliza características de la serie de tiempo del brillo del objeto, lo cual utiliza un *hierarchical random forest* (Sánchez-Sáez et al., 2021).

Un tipo de objeto astronómico interesante son los transientes, específicamente las supernovas; aquellos objetos que realizan una reacción volátil en el tiempo o mejor dicho una violenta explosión en el universo. Raramente se puede observar y estudiar una supernova en el momento de su explosión, pero con la llegada de los *astronomical surveys* y modelos de detección temprana, es posible caracterizarlas e investigarlas en etapas tempranas de su evolución.

Transient Name Server (TNS) es el mecanismo oficial de la International Astronomical Union (IAU) para reportar nuevos objetos transientes (e.g supernovas). Los *brokers* utilizan este mecanismo diariamente para reportar posibles supernovas. Un reporte está constituido por la fecha de descubrimiento, coordenadas ecuatoriales, grupo que genera el reporte, instrumento con el cual se descubrió, entre otros campos que permiten identificar el descubrimiento y el grupo que lo ha reportado. Posteriormente, otros grupos de investigación a través de instrumentos de rayos x pueden confirmar espectroscópicamente[2] la existencia de la supernova y queda registrado en TNS.

En la Figura 1.1 se puede ver la cantidad de supernovas confirmadas espectroscópicamente en TNS. Como se logra apreciar desde el año 2011 a la actualidad sólo se han confirmado 8675 supernovas.

[2]Proceso por el cual se confirma y estudia la explosión de una supernova.

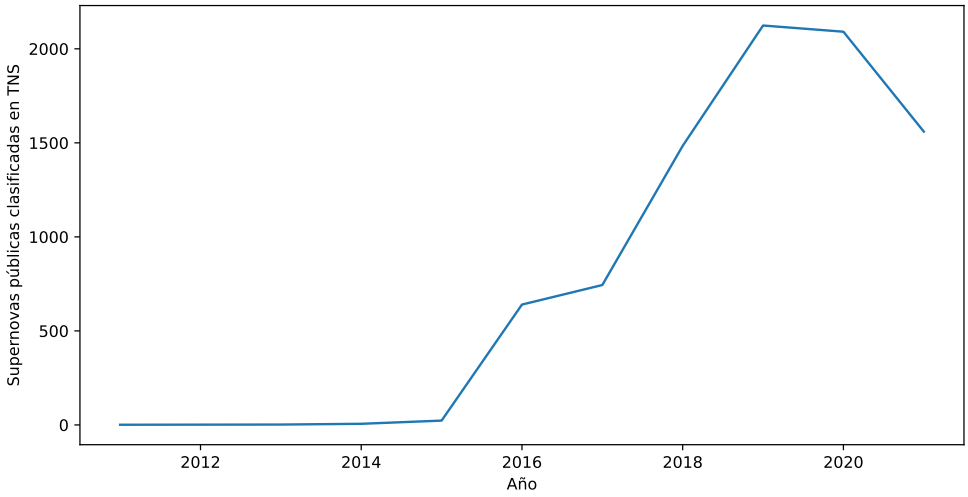


Figura 1.1: Cantidad de supernovas públicas y confirmadas en TNS por año (hasta 01/09/2021). Fuente: Elaboración propia con datos de TNS (`https://www.wis-tns.org/stats-maps`)

1.2 DESCRIPCIÓN DEL PROBLEMA

Para Bloom et al. (2012) el problema de clasificación en un *stream* de datos tiene la particularidad de desafiar la forma de hacer ciencia en astronomía, dado que corresponde a un medio para poder maximizar el rendimiento científico en un entorno con recursos limitados. La clasificación permitiría el descubrimiento de objetos nuevos, posibilitando la realización de *follow up*[3] inteligente y estudiar objetos que aún no han sido estudiados, lo que permitiría caracterizar rápidamente nuevos hallazgos astronómicos.

En el contexto del *broker* ALeRCE, se utiliza el modelo de clasificación temprana, llamado *stamp classifier* para clasificar **datos de la primera alerta de un objeto astronómico** y tiene como objetivo principal descubrir supernovas jóvenes. Además ha resultado útil para clasificar otro tipos de objetos, como asteroides, estrellas variables, entre otros (Carrasco-Davis et al., 2020).

El modelo recibe el nombre de *stamp classifier*, dado que utiliza pequeños recortes de imágenes astronómicas contenidas en las alertas, denominadas estampillas (*stamps* en inglés). Cada alerta contiene una tripleta de estampillas que dan contexto a la observación.

Diariamente el modelo clasifica los datos de ZTF en tiempo real, luego los expertos de ALeRCE inspeccionan manualmente los resultados predichos para su posterior reporte a TNS. El *broker* ALeRCE ha logrado reportar 1431 supernovas que han sido confirmadas

[3]Corresponde a la realización de seguimiento con instrumentos particulares a un determinado objeto astronómico, es decir, observar en detención un fenómeno que ocurre en el universo y supervisar eventos futuros en esa zona.

espectroscópicamente. Esto ha permitido posicionar a ALeRCE como uno de los *brokers* más prometedores en clasificar este tipo de objetos, véase Figura 1.2.

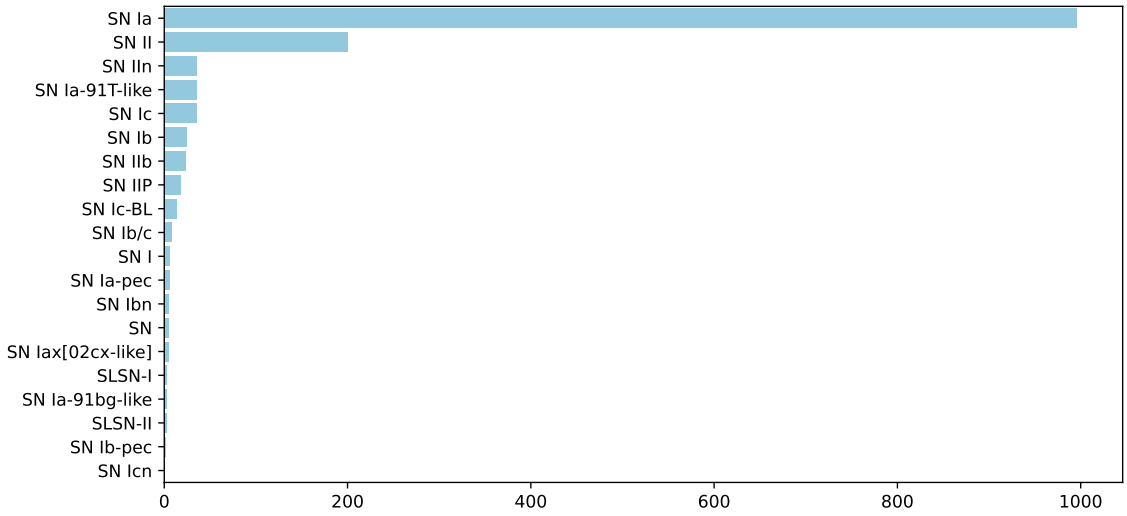


Figura 1.2: Cantidad de supernovas clasificadas y reportadas por ALeRCE que han sido confirmadas espectroscópicamente (hasta 01/09/2021). En el eje y indica el tipo de supernova y el eje x la cantidad de veces que ese tipo de objeto ha sido reportado. Las supernovas se agrupan según las líneas de absorción de los elementos químicos que aparecen en sus espectros. El tipo de supernova que se ha detectado en mayor medida en ALeRCE es la tipo Ia (carecen de helio y presentan una línea de silicio en el espectro). Fuente: Elaboración propia con datos de TNS (`https://www.wis-tns.org/stats-maps`)

El modelo *stamp classifier* sólo utiliza la primera alerta de cada objeto para clasificar, es decir, no utiliza las alertar posteriores para afinar la clasificación. Es posible que en objetos como las supernovas, un conjunto de las primeras imágenes del objeto provean información acerca de la evolución que se está teniendo, esto podría mejorar el rendimiento del modelo y/o la incorporación de inferencia a otras clases.

Por otro lado, el clasificador tardío que utiliza series de tiempo, debe esperar una cantidad determinada de alertas de un objeto para poder operar. Esta condición puede cumplirse en un lapso de días a meses. Por lo que no sirve para detectar supernovas jóvenes.

El proceso de clasificación automática está sujeto a error, por esta razón el equipo de astrónomos de ALeRCE que investigan transientes deben estar día a día revisando los resultados del modelo en una aplicación web llamada *SN Hunter*[4], la que permite a los expertos explorar, filtrar y observar las supernovas clasificadas por el modelo *stamp classifier* (Carrasco-Davis et al., 2020).

[4]https://snhunter.alerce.online/

1.3 SOLUCIÓN PROPUESTA

La clasificación de datos astronómico en tiempo real es un área que se ha iniciado durante este siglo. Según Bloom et al. (2012) el aprendizaje de máquina permite resolver este tipo de problemas, debido a que cumple con:

- Las máquinas que han sido entrenadas pueden actuar más rápido que un humano en cuanto a la tarea de descubrir y clasificar candidatos astronómicos, además de solo estar limitadas por recursos computacionales, mientras que un humano puede estar limitado por condiciones físicas, psicológicas, experiencia en uso instrumentos, etc.
- Se podría realizar un *follow up* totalmente automático, lo que permitiría utilizar este tipo de herramientas de una manera más eficiente en términos de tiempo, dado que al clasificar un objeto de alguna clase de interés, el seguimiento se puede gatillar de manera autónoma y observar el objeto avistado lo más pronto posible.
- Los resultados de aprendizaje de máquina son reproducibles, en cambio la clasificación humana no.
- La experimentación para un nuevo esquema de clasificación requiere poco más que volver a ejecutar nuevos códigos en datos existentes, mientras que un cambio a enfoques basados en humanos requiere trabajo adicional intensivo en mano de obra.

La solución propuesta corresponde a un modelo basado en *stamp classifier*, el cual pueda recibir datos de las primeras dos alertas de un objeto observado por ZTF. El modelo debe ser capaz de clasificar al menos la misma cantidad de clases que el modelo propuesto por Carrasco-Davis et al. (2020) y obtener una métrica de clasificación comparable con dicho modelo.

La solución se basa en este modelo, puesto que ha sido uno de los modelos más exitosos en cuanto a la clasificación en tiempo real de estampillas de objetos, además de ser uno de los primeros modelos en distinguir entre 5 clases sólo utilizando las estampillas de la primera alerta. Por lo tanto, el rendimiento de la solución propuesta debería ser al menos igual a la de *stamp classifier*.

Un aspecto importante a destacar de la solución propuesta, es que ha diferencia del modelo *stamp classifier* se considera un componente temporal, que puede aportar información de la evolución que ha tenido el objeto en el brillo y forma en función del tiempo. Por esta razón se considera el uso de las dos primeras alertas para resolver el problema de clasificación.

1.4 OBJETIVOS Y ALCANCE DEL PROYECTO

1.4.1 Objetivo general

Lograr establecer cuál es el impacto de un componente temporal en el modelo de clasificación temprana de ALeRCE, comparando el desempeño de diferentes modelos de tal manera evaluar si existen mejoras significativas al respecto.

1.4.2 Objetivos específicos

- Disponer de un conjunto de datos de las primeras dos alertas a partir de las etiquetas de ALeRCE.
- Disponer un conjunto de datos de las primeras dos alertas que no tengan etiquetas asociadas (para verificación del modelo).
- Lograr un diseño de arquitectura de red neuronal basada en *stamp classifier* que utilice dos estampillas.
- Entrenar el modelo de *stamp classifier* con la primera alerta del conjunto etiquetado.
- Entrenar arquitectura diseñada con las dos primeras alertas del conjunto etiquetado.
- Evaluar los resultados obtenidos mediante métricas de clasificación.
- Evaluar los resultados obtenidos en el conjunto no etiquetado.
- Reportar comparación de los resultados con la literatura.
- Evaluar el impacto de la componente temporal en el modelo propuesto.

1.4.3 Alcances

Los modelos propuestos son similares al presentado por Carrasco-Davis et al. (2020). Las variaciones posibles a investigar corresponden a modificaciones en dicha red, con la finalidad de utilizar las primeras dos alertas como entrada.

La combinatoria de hiper-parámetros de los modelos propuesto tiene que ser suficiente pero acotada. Cada combinación se repetirá 5 veces para disponer margen de error.

Se utilizan datos que provienen del *survey* ZTF, en donde los datos tienen las siguientes características:

- Obtención de datos es de manera irregular, es decir, las observaciones a los objetos astronómicos no ocurren con una diferencia temporal equidistante.
- Las imágenes son de 63×63 píxeles, por lo cual la solución debe tolerar entradas de este tamaño o cortes de menor dimensión de estas mismas.
- Se consideran características provenientes de las alertas de ZTF como por ejemplo las coordenadas ecuatoriales y la magnitud de la detección, además de otras características propuestas por Carrasco-Davis et al. (2020). Por lo tanto no se considera enriquecimiento externo como lo puede ser un *cross match* externo u otro proceso.

El conjunto de datos etiquetado para el desarrollo del modelo considera las clases que contempla la taxonomía de ALeRCE, dicha taxonomía se especifica en detalle en Anexo A.

Las comparaciones con otros modelos se hará con los mismos conjuntos de entrenamiento, validación y prueba. Para la comparación con el modelo de Carrasco-Davis et al. (2020) solo se utiliza la primera alerta de los conjuntos descritos anteriormente.

El desempeño del modelo va a estar centrado principalmente en la clase supernova, dado que el objetivo del clasificador es actuar de manera temprana para el descubrimiento de nuevos objetos de este tipo, esto porque existen objetos astronómicos que varían poco en comparación a la clase supernova con las dos primeras alertas.

1.5 METODOLOGÍA Y HERRAMIENTAS UTILIZADAS

1.5.1 Metodología

El problema de clasificación de objetos astronómicos mediante el uso de *machine learning* ha sido estudiado por parte de la comunidad, es por esto que se debe abarcar etapas de investigación y experimentación. Se debe estudiar algunas arquitecturas de redes neuronales artificiales que resuelvan el problema propuesto, por lo cual resulta bastante útil basarse en el método científico para dirigir este trabajo.

Según Carey (2011), el método científico corresponde a una metodología que permite obtener conocimiento a través de un hecho, hipótesis, experimentos y resultados, asegurando al reproducibilidad de los resultados expuestos por el investigador.

La metodología usada corresponde a lo propuesto por Hernández et al. (2010):

- Planteamiento del problema: Véase Sección 1.2.
- Formulación de la hipótesis: Utilizando datos de objetos detectados por ZTF que tengan más de una detección, es posible entrenar un modelo basado en *stamp classifier* que incorpore componentes temporales y que tenga mejor rendimiento en métricas de clasificación en comparación a un modelo que utiliza una sola detección (métricas como *accuracy*, *f1-score*, etc).
- Revisión de la literatura: Véase Capítulo 2.
- Desarrollo del diseño de investigación: Véase Capítulo 4.
- Análisis de resultados y conclusiones: Véase Capítulo 6.

En el ámbito de la investigación científica, es necesario utilizar una metodología de desarrollo acorde a la solución, dado que la solución propuesta está compuesta por un *sofware* se determina el uso de *eXtreme Programming* (XP).

Beck (2000) plantea XP como una metodología de desarrollo de ingeniería de *software*, una de las características esenciales de esta metodología corresponde a la adaptabilidad en términos de cambios sobre la marcha del proyecto.

Para esta investigación se utiliza XP para diseñar, programar las arquitecturas, entrenar y evaluar los modelos realizados. Por otro lado, en la fase de validación de los resultados se interactúa de manera periódica con astrónomos de ALeRCE (A. Múñoz, F. Förster, F. Bauer, P. Sánchez, L. Hernández).

1.5.2 Software

El *stack* tecnológico a utilizar está asociado a herramientas para la programación científica. Es por esto que se considera como eje principal el lenguaje de programación Python para el desarrollo de este trabajo. El *software* que se ha utilizado tiene el objetivo de potenciar características de Python y la elaboración del informe final. A continuación se presentan que herramientas se utilizaron y por qué motivo:

- Anaconda: Es una distribución libre de ambientes para lenguajes de programación relacionados con ciencia de datos (Anaconda Inc., 2020). Anaconda permite el acceso a diversos paquetes a utilizar en el proyecto y la creación de entornos virtuales que permiten la reproductibilidad de los experimentos.
- Jupyter: Conjunto de herramientas compatible con Python (Kluyver et al., 2016). Una de las herramientas utilizadas en esta tesis han sido los *notebook* de Jupyter para el análisis exploratorio de los datos, creación de visualizaciones y experimentación.
- Python: Lenguaje de programación de alto nivel con el que se desarrollará el proyecto, se utilizará la versión 3.8.5.
- TikZ: Herramienta para la producción de gráficos vectoriales a partir de sintaxis de LaTeX (Mertz & Slough, 2007). Se ha utilizado para la elaboración propia de diagramas y figuras.
- Overleaf: Para la escritura y documentación del trabajo realizado. Se utiliza el compilador de XeLaTex y formato[5] LaTeX de acuerdo a las normas expresadas en el manual de tesis del año 2014 de la Universidad de Santiago de Chile.
- Además se utilizan los siguientes paquetes disponibles para Python:
  - Matplotlib: Librería para crear gráfico y grillas (Hunter, 2007).
  - NumPy: Dentro del campo de la ciencia de datos, NumPy ha sido una de las librerías más populares de la última década, principalmente por la eficiencia de los cálculos vectoriales (debido a que gran parte de su código fuente está optimizado y escrito en C) y por la simpleza de su uso (Harris et al., 2020). Esta herramienta es utilizada en este proyecto para cálculos y como herramienta auxiliar de Pandas. Se utiliza en su versión 1.21.0.
  - Pandas: Es una librería de alto nivel para la computación científica, fue propuesta por Wes McKinney (2010). Ésta ha permitido trabajar con estructuras de datos complejas como arreglos, *data frames*, series, etc. Además permite la lectura y escritura de conjuntos de datos en distintos formatos que ya son un estándar en la ciencia de datos, como por ejemplo CSV, parquet, pickle, entre otros. Pandas ha sido utilizado para el preprocesamiento de datos y otros cálculos de interés que han permitido obtener los resultados de este trabajo. La versión utilizada corresponde a la 1.2.4.
  - PySpark: Corresponde a una interfaz para poder utilizar Apache Spark desde Python. Esta herramienta es utilizada para manejar una gran cantidad de datos en recursos distribuidos, por lo que permite operar y analizar los datos a trabajar (Zaharia et al.,

[5]Formato desarrollado por estudiantes del Departamento de Ingeniería Informática. Más información aquí.

2016). El uso de Apache Spark en este proyecto tiene la finalidad de recuperar un conjunto de datos del total de datos, por lo cual su uso se hace imprescindible, ya que permite navegar de manera fluida en los datos hasta la fecha. En esta ocasión se trabaja con PySpark 3.1.2.

- Seaborn: Librería complementaria a matplotlib que permite crear gráficos con *dataframes* de pandas (Waskom, 2021).
- TensorFlow: Es una librería desarrollada por parte del equipo de Google, actualmente es utilizada para el desarrollo de modelos de aprendizaje automático y para otros cálculos matemáticos de manera eficiente. La elección de esta librería recae principalmente por su basta documentación, su comunidad activa y porque permite hacer cálculos en GPU (Abadi et al., 2015). Para este trabajo se utiliza la versión 2.4.0.

1.5.3 Hardware

Computador de escritorio para el análisis de datos y resultados, que tiene las siguientes especificaciones:

- RAM: 32 GB
- CPU: Intel Core i7-10700 CPU @ 2.90GHz × 16
- GPU: Nvidia Geforce RTX 2060 de 6 GB
- SSD: 500 GB
- HDD: 1 TB
- Sistema operativo Ubuntu 20.04

Para la conformación del conjunto de datos se utiliza Amazon EMR, el cual es un servicio que provee la creación de clústers a medida, además permite comunicación directa con el almacenamiento de Amazon S3, las especificaciones del clúster a ocupar son:

- 1 nodo principal y 30 nodos secundarios
- Nodo *master* en una instancia m5.xlarge (4 vCPU y 16 GB de RAM)
- Nodos *cores* en instancias m5.2xlarge (8 vCPU y 32 GB de RAM)

Para el preprocesamiento, serialización en TFRecords y entrenamiento de los modelos, se utiliza un servidor alojado en el Departamento de Ingeniería en Ingeniería Informática (DIINF) con las siguientes características:

- RAM: 252 GB
- CPU: Intel(R) Xeon(R) CPU v4 @ 2.40GHz × 40
- GPU: Nvidia Tesla P100 de 16 GB × 4
- SSD: 1.7 TB
- Sistema operativo Ubuntu 18.04.5

1.6 ORGANIZACIÓN DEL DOCUMENTO

El documento se organiza de la siguiente manera. En el Capítulo 1 se introduce el problema y se expone la solución propuesta. El Capítulo 2 realiza un barrido de los conceptos involucrados en la investigación, además de exponer el estado del arte de la clasificación de objetos astronómicos mediante estampillas. En el Capítulo 3 se expone el conjunto de datos utilizados en este trabajo, un análisis de éstos y cómo se conforma el conjunto de entrenamiento, validación y prueba. Luego en el Capítulo 4 se presentan las arquitecturas propuestas y en el Capítulo 5 el diseño experimental. En el Capítulo 6 se presentan los resultados y un análisis de éstos. Finalmente en el Capítulo 7 se destaca los principales hallazgos y se realiza una conclusión al respecto.

CAPÍTULO 2. MARCO TEÓRICO Y REVISIÓN DE LA LITERATURA

En este capítulo se presenta y explica los conceptos importantes para el desarrollo de este trabajo de título. Se explica en detalle los conceptos básicos relacionados con astronomía, como es el proceso de obtención de imágenes astronómicas y algunas propiedades de objetos de interés de estudio. Por otro lado se explican conceptos relacionados con aprendizaje profundo y como éstos se relacionan con la tarea de clasificación.

2.1 ASTRONOMÍA

2.1.1 Conceptos básicos

Las **estrellas** se conforman por la condensación de acumulación de gas y de polvo; las estrellas visibles desde la tierra se encuentran en la galaxia local, la Vía Láctea. Una estrella se podría definir como una esfera extremadamente caliente de plasma que genera energía a partir de la fusión nuclear en su núcleo. La masa de las estrellas define el tamaño, temperatura y longevidad; siendo las estrellas más grandes más calientes y con corta vida, mientras que las estrellas más pequeñas son más longevas y frías (Kippenhahn et al., 1990).

Para medir el brillo de una estrella, los astrónomos utilizan dos escalas separadas: por un lado se utiliza la **magnitud aparente**, que expresa que tan brillante es el objeto desde la perspectiva de la Tierra, mientras que la **magnitud absoluta** corresponde al brillo real del objeto, es decir el brillo intrínseco. En la Figura 2.1 se puede apreciar dos estrellas A y B que tienen la misma magnitud aparente (se ven igual de brillantes desde la Tierra), pero la magnitud absoluta de A es mayor que B. B se ve tan brillante como A dado que es más cercana a la Tierra. Mientras menor sea el valor de la magnitud, significa que la estrella es más brillante.

Generalmente para observar el universo se utilizan ciertos **filtros de banda ancha**, los cuales se encargan de eliminar ciertas líneas espectrales[1], la idea es restringir el rango del espectro que se puede ver. Además suelen usarse para obtener más información, por ejemplo eliminando parte de la contaminación lumínica o permitir la observación objetos del cielo profundo.

Una forma de ver la evolución de la magnitud a través del tiempo, ya sea de alguna región del cielo o de algún astro en particular, es mediante la **curva de luz** (Milone, 1993). Ésta

[1] En la absorción de la radiación frente a la longitud de onda o frecuencia aparecen una serie de líneas o bandas denominadas líneas espectrales.

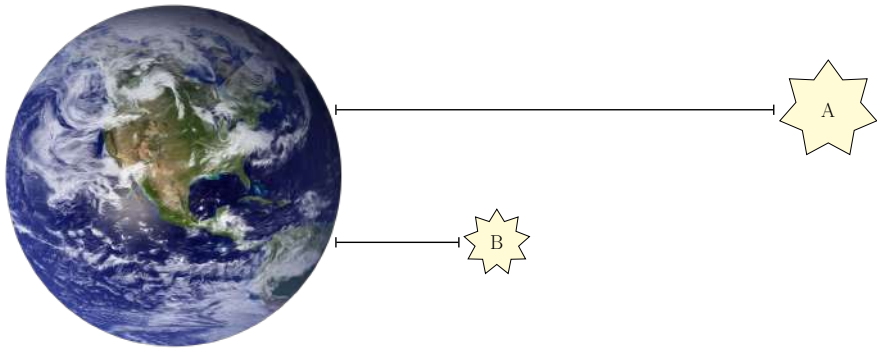


Figura 2.1: Magnitud aparente y magnitud absoluta. La magnitud es una medida del brillo de un astro, donde la magnitud aparente corresponde al brillo que se observa desde la tierra y la magnitud absoluta es el brillo empírico del astro. Fuente: Elaboración propia, 2021

corresponde a una serie de tiempo. En la Figura 2.2 se muestra la evolución de la magnitud del objeto ZTF17aaaecvk[2]. La curva de luz permite discriminar que tipo de objeto astronómico puede ser, a través de características de las series de tiempo como por ejemplo el periodo, amplitud, etc. Es por esta razón que la curva de luz a través de un análisis determinado permite resolver el problema de clasificación. La unidad de tiempo utilizada en las curvas de luz es el *Modified Julian Date*[3] (MJD).

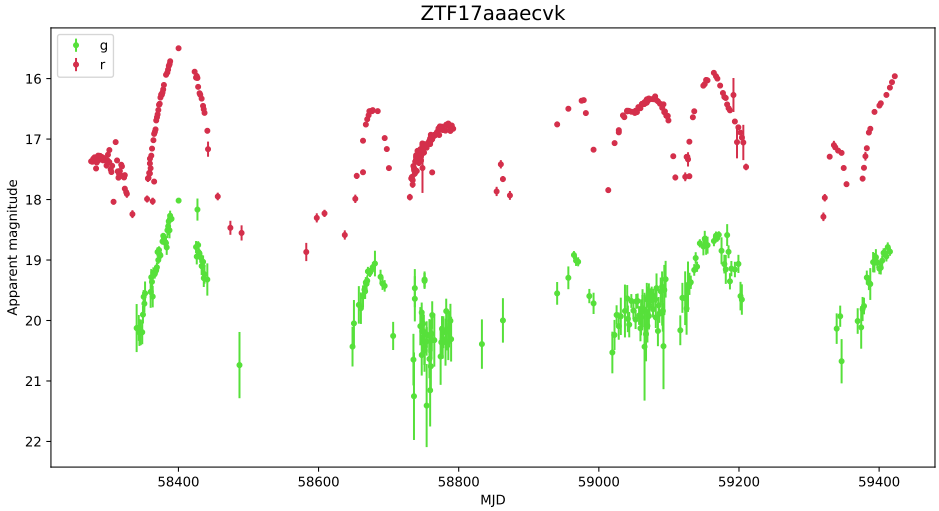


Figura 2.2: Curva de luz de objeto ZTF17aaaecvk. En verde se muestra la banda *g* y en rojo la banda *r*, éstas bandas diferencian los filtros de banda ancha utilizados en la observación. Fuente: Elaboración propia, 2021

Para ubicar un objeto en la esfera celeste se utiliza el sistema de coordenadas

[2] `https://alerce.online/object/ZTF17aaaecvk`
[3] Número de días solares desde la medianoche del 17 de noviembre de 1858 (UTC).

ecuatoriales, el cual permite localizar objetos astronómicos respecto a coordenadas equivalentes a la latitud y longitud. Estas coordenadas reciben el nombre de **ascensión recta** (*right ascension*) y **declinación** (*declination*) y se les llama por su acrónimo en inglés RA y Dec respectivamente. La unidad suele ser en grados; el rango de RA es [0, 360] grados y en Dec es de [-90, 90] grados.

2.1.2 Objetos astronómicos de interés

Los objetos astronómicos son entidades físicas que se encuentran en el universo observable. Los astrónomos se dedican a estudiar e investigar estos objetos para conocer su composición, medir distancias, etc. Algunos ejemplos de objetos astronómicos pueden ser un sistema solar, estrellas, galaxias, entre otros. Los objetos astronómicos involucrados en este trabajo de tesis son los siguientes:

- Asteroides: Son rocas gigantes que orbitan por el Sistema Solar. Generalmente se encuentran en el ”cinturón de asteroides” que está entre Marte y Júpiter, aunque en ocasiones algunos de ellos se acercan de manera peligrosa a la Tierra.
- Núcleo activo de galaxia: También conocidas como AGN por su sigla en inglés (Active Galactic Nucleus). Generalmente las galaxias giran en torno a un agujero negro central supermasivo, pero en ciertas galaxias (AGN) la materia está cayendo en el agujero negro, esto provoca que la materia se consuma y desgarre por la gravedad. Luego la materia gira alrededor del centro galáctico formando un anillo de polvo y gas sobrecalentado. Es posible ver dos chorros de partículas (ambos se disparan para afuera del anillo) que viajan a la velocidad de la luz y estirándose por miles de años luz[4] en el espacio.
- Estrellas variables: Una estrella puede variar su brillo por distintas razones, por ejemplo una estrella en un sistema binario es una estrella que se mueve delante de otra, haciendo que el sistema varíe su brillo en el tiempo, son conocidas como binarias eclipsantes *(eclipsing binary stars*).
- Supernovas: Cuando una estrella sufre una explosión extremadamente violenta al final de su vida, recibe el nombre de supernova. La explosión emite una gran cantidad de radiación y una nueva luz brillante permanece en el cielo desde días a meses. El avistamiento de este tipo de objeto ocurren raramente debido a su volatilidad.

Además de los objetos empíricos descritos anteriormente, los *astronomical surveys* suelen producir *bogus* (alertas falsas), es decir indicar que existe algo interesante en una

[4]Distancia que recorre la luz en un año.

región que no tiene una fuente de interés científico. Esto ocurre por efectos de la cámara del telescopio: píxeles saturados en centros de regiones muy brillantes, errores astrométricos, formación defectuosa de la imagen, entre otros.

Cuando se descubre algún objeto de interés, se registra en un **catálogo astronómico**. Este último corresponde a un listado de objetos interesantes que se agrupan según una característica en común (tipo de objeto, instrumento utilizado, origen, etc). Los catálogos astronómicos sirven para informar a la comunidad astronómica sobre donde encontrar ciertos tipo de objetos.

Los catálogos astronómicos se pueden operar entre sí mediante el ***crossmatch***, esto significa realizar una búsqueda espacial de objetos entre dos o más catálogos, a través de sus coordenadas ecuatoriales y un radio definido (esta búsqueda también es conocida como *conesearch*). Algunos catálogos astronómicos conocidos son Simbad (Wenger et al., 2000), AllWISE (Cutri et al., 2014), Sloan Digital Sky Survey (SDSS) (York et al., 2000), entre otros.

2.1.3 *Astronomical surveys*

Los ***astronomical surveys*** corresponden a telescopios de última generación, ya sean terrestres o satelitales, que mapean de manera sistemática el universo y sus componentes, con la finalidad de descubrir nuevos objetos o fenómenos de interés. El mapeo consiste en la captura de imágenes, posiciones, distancias, etc. Actualmente los *astronomical surveys* son los mayores generadores de datos en astronomía, debido a la rápida evolución tecnológica en el área de la computación. Esto ha transformado la manera de hacer astronomía (Djorgovski et al., 2013).

Matson (2012) cuenta que los astrónomos durante mucho tiempo han sido capaces de mapear manualmente los objetos astronómicos en el cielo. Este sondeo era bastante limitado y su objetivo principal consistía en la producción de catálogos astronómicos para ciertos tipos de objetos. Gracias a los *astronomical surveys* los mapas celestiales van a sufrir cambios importantes, puesto que se pueden afinar con más detalles dichos mapas con el transcurso del tiempo. Se pueden complementar con nuevas posiciones y distancias de miles de millones de estrellas y galaxias cercanas y lejanas.

Otra particularidad de los *astronomical surveys* es que pueden limitar a una banda del espectro electromagnético (por limitación instrumental), aún así, es posible utilizar mecanismos robóticos que permiten realizar sondeos en múltiples longitudes de onda (Mickaelian, 2016).

Los *astronomical surveys* tienen un gran valor científico, ya que muchas veces son el paso inicial para explorar algún objeto en particular. Es posible hacer ciencia solo con los datos de un *astronomical survey* en particular, aunque la combinación de varias fuentes permitirían

recolectar más antecedentes de ciertas regiones del cielo. Por otro lado, estos instrumentos permitirían notificar objetos potencialmente interesantes o desconocidos, gatillando un ***follow up*** prometedor y con un alto interés científico (Djorgovski et al., 2013).

Otro aspecto importante de los *astronomical surveys*, es la ***cadencia*** con la que observan. La cadencia corresponde al ritmo con que se observan los objetos. En particular los *astronomical surveys* tienen una alta cadencia, por lo que dispone la variable tiempo de forma más detallada. Esto permite encontrar objetos que se caracterizan por variar en escalas de tiempo cortas.

Generalmente los *surveys* están situados en lugares con cielos limpios y condiciones climáticas que permitan un funcionamiento periódico. Muchos de estos telescopios están optando por la transmisión en tiempo real de los datos, dando paso a la recolección de grandes volúmenes de datos. Algunos *astronomical surveys* a destacar son: Asteroid Terrestrial-Impact Last Alert (ATLAS) en Hawaii (Tonry et al., 2018); Zwicky Transient Facility (ZTF) en Estados Unidos (Bellm et al., 2018); entre otros. Para los próximos años, entrará en funcionamiento el Vera C. Rubin Observatory y su telescopio Legacy Survey of Space and Time (LSST) en Chile, el telescopio óptico más grande jamás construido, con una cámara de 3200 mega píxeles y un espejo de 8 metros de diámetro, el cual será capaz de captar 20 terabytes de datos por noche (Brough et al., 2020).

El funcionamiento de un *astronomical survey* está dado por la recolección de luz repetitiva en distintas regiones del cielo noche a noche. En la Figura 2.3 se muestra el flujo general de procesamiento en un *astronomical survey*: (1) el telescopio apunta a una región con un filtro de banda y recolecta grandes imágenes donde hay varios objetos astronómicos. (2) se consulta por las imágenes de referencia[5] de la región apuntada. (3) se comparan las imágenes de *science* con *reference*, calculando si existen diferencias significativas entre las imágenes (variación en la posición o brillo). (4) en caso de haber diferencias significativas, se identifica la zona que ha variado y se recorta en cada imagen (*science*, *reference* y *difference*). Los recortes se empaquetan en una alerta y además se adjuntan metadatos de la observación. Por ejemplo la posición en el cielo, magnitud del objeto, etc. En la Sub-sección 2.2.3 se detalla acerca de las imágenes de *science*, *reference* y *difference*.

En el desarrollo de esta tesis se trabaja con datos provenientes del *survey* ZTF, en la Sección 2.2 se detalla el funcionamiento particular de este telescopio y sus características, aunque se debe tener en cuenta que el funcionamiento de los distintos *astronomical surveys* es similar.

[5]Corresponde a imágenes que se han tomado desde el inicio del funcionamiento de ZTF. Las imágenes de referencia se van actualizando con el tiempo.

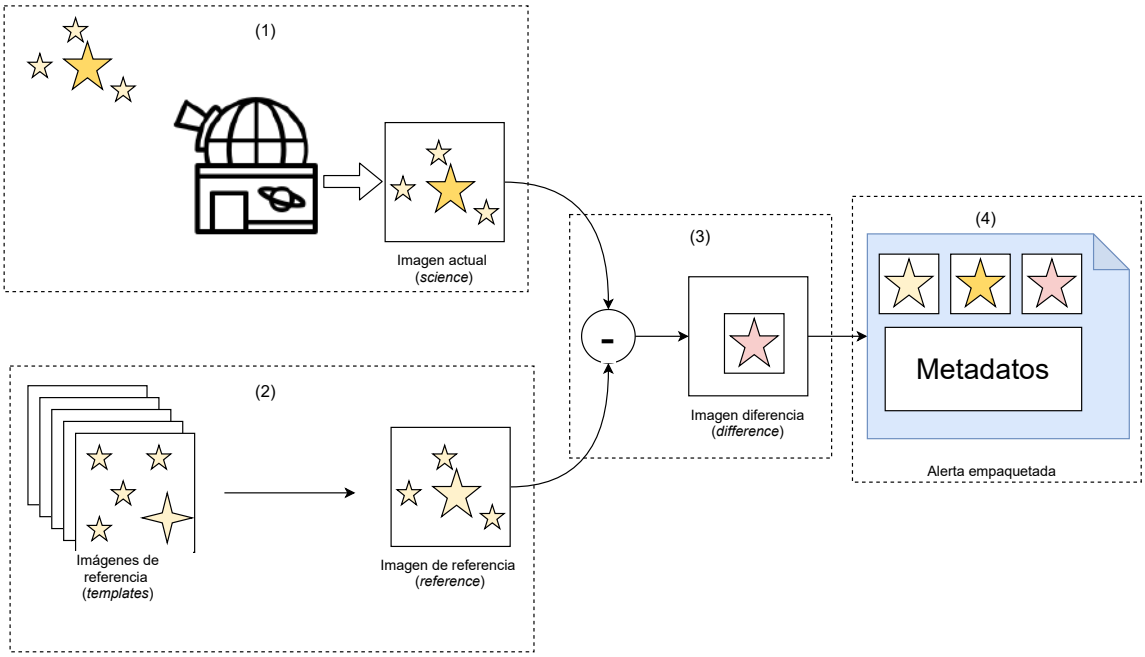


Figura 2.3: Diagrama simplificado del funcionamiento de los *astronomical surveys*. El funcionamiento comienza en (1) cuando se obtiene una imagen nueva de un objeto, en (2) se busca una imagen de referencia de la zona observada, luego en (3) se compara la imagen nueva con la imagen de referencia, para esto se hace un recorte de la imagen centrada en el objeto variante. Si la diferencia de las imágenes es significativa en (4) se gatilla una alerta astronómica con la información de la observación. Fuente: Elaboración propia, 2021

2.2 *ZWICKY TRANSIENT FACILITY*

ZTF es un telescopio óptico que está situado en el hemisferio norte, específicamente en el Observatorio Palomar en San Diego, California, Estados Unidos. Es un *survey* que tiene la capacidad de monitorear el cielo cientos de veces en tres bandas, en la Figura 2.4 se aprecia los tres filtros utilizados por ZTF y su longitud de onda[6]. Las observaciones son tomadas con una cadencia que va desde los minutos a meses (Bellm et al., 2018).

Uno de los aspectos que más llaman la atención de los telescopios ópticos suele ser el campo de visión (FOV por su sigla en inglés), esto corresponde a la porción del cielo observable que cubre el instrumento. En el caso de ZTF el FOV es uno de los más amplios actualmente. En la Figura 2.5 se muestra una comparación del FOV de telescopios operativos y en desarrollo.

Sin embargo, puede existir telescopios con un amplio FOV, pero con una baja recolección de luz. Una medida justa para comparar este tipo de instrumento es el *etendue*, que corresponde al producto del FOV y la recolección de luz, en otras palabras es una medida del volumen del universo que un telescopio puede observar en una sola exposición. En la Figura 2.6

[6]Esta investigación ha hecho uso de *SVO Filter Profile Service* (`http://svo2.cab.inta-csic.es/theory/fps/`) apoyado por el MINECO español a través de la subvención AYA2017-84089

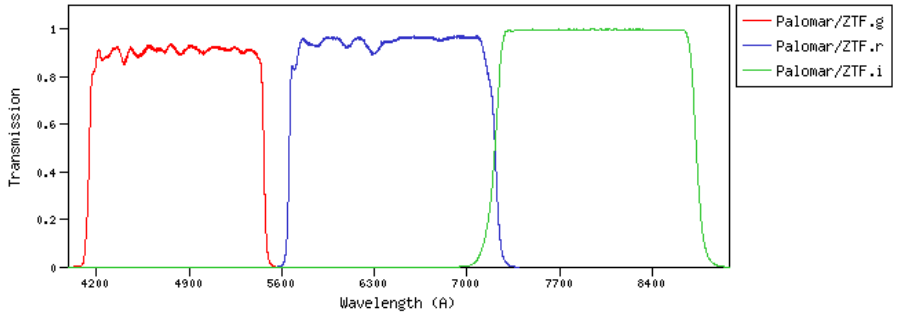


Figura 2.4: Longitud de onda de los filtros de ZTF. Los filtros de ZTF son *g*, *r* e *i*, en rojo, azul y verde respectivamente. Fuente: Rodrigo & Solano (2020)

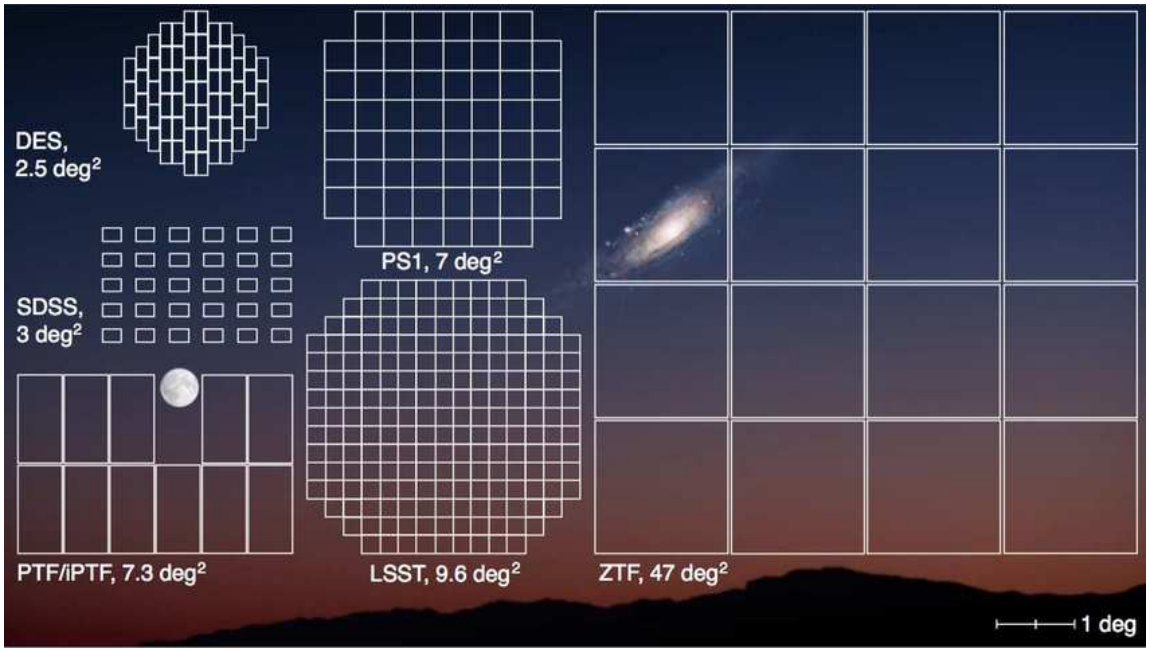


Figura 2.5: Campo de visión de la cámara ZTF comparado con el de otras cámaras de gran tamaño. La Luna y la Galaxia de Andrómeda se muestran a escala (se cuenta con el permiso de Joel Johansson). Fuente: Laher et al. (2017)

se aprecia el *etendue* de diversos telescopios, el color de cada círculo indica la cantidad de píxeles de la cámara del instrumento y al lado derecho se presenta la escala de colores correspondiente (Förster et al., 2020).

2.2.1 Funcionamiento de la cámara

Actualmente las cámaras de los *surveys* astronómicos están construidos en base a *charge coupled devices* (CCD). Estos dispositivos son circuitos integrados que se encargan

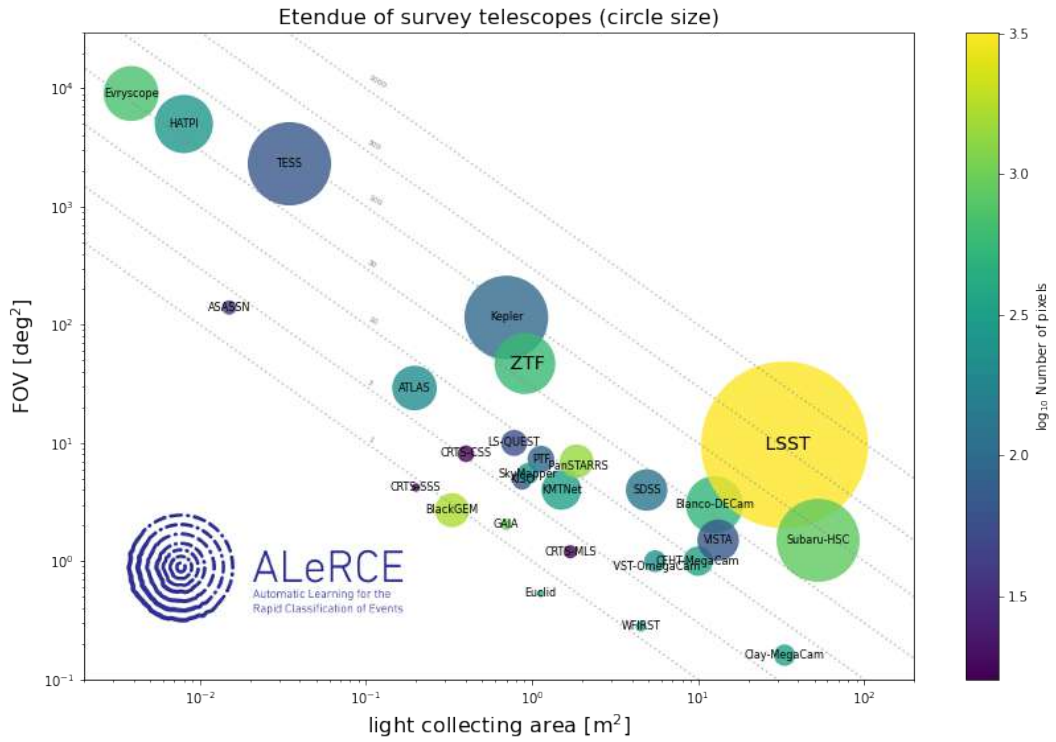


Figura 2.6: Campo de visión (FOV por su sigla en inglés) versus el área de recolección de luz para una selección de telescopios ópticos terrestres y espaciales actualmente operativos o planificados. Fuente: Förster et al. (2020)

de transformar una señal luminosa en una señal eléctrica; son capaces de obtener señales del espectro visible, infrarrojo y ultravioleta (Janesick, 2001). Un CCD está compuesto por una cuadrícula de sensores que reaccionan a la luz. En la Figura 2.7 se muestra un CCD de 2/3 pulgadas.

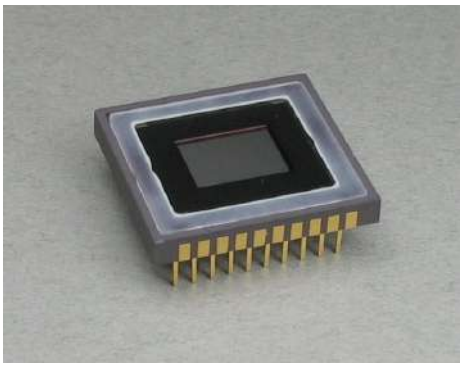
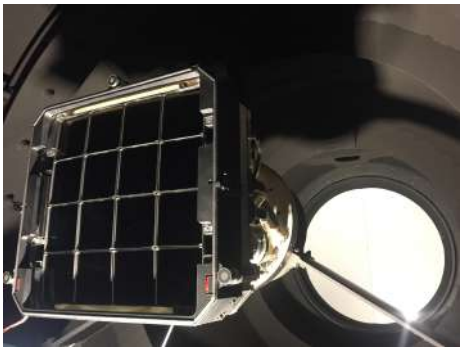


Figura 2.7: Imagen de un sensor CCD de 2/3 pulgadas. Fuente: `https://commons.wikimedia.org/wiki/File:CCD_Image_sensor.jpg` bajo la licencia CC BY-SA 3.0

En el caso de ZTF se tiene un mosaico de 4x4 CCDs (en total se tienen 16 CCDs), en la Figura 2.8a se muestran los 16 CCDs de ZTF. Cada CCD de ZTF se divide en cuatro cuadrantes

de 3072x3080 píxeles, por lo tanto un CCD es de 6144x6160 píxeles (Laher et al., 2017). Por otro lado, en la Figura 2.8b se puede dimensionar el tamaño de los filtros ocupados para tapar todos los CCDs de la cámara.



(a) *Cámara de ZTF*

(b) *Filtro de ZTF*

Figura 2.8: Cámara y filtro de ZTF. En 2.8a se aprecian los 16 CCDs de ZTF operando y en 2.8b se observa uno de los filtros utilizados en la cámara. ZTF tiene una mano robótica que intercambia los filtros después de cada exposición. Fuente: Palomar Observatory/California Institute of Technology

Cuando la cámara de ZTF entra en funcionamiento, se abre el obturador en un tiempo de exposición de 30 minutos y se captura la luz proveniente de múltiples fuentes que se transmiten por el espacio. La luz pasa por la atmósfera de la Tierra y finalmente llega a la cámara de ZTF. Durante el trayecto de la luz recibida es posible que ocurran acciones que la distorsionen. La luz llena con carga los sensores de cada píxel de los CCDs, de manera proporcional a su intensidad. Una vez que se cierra el obturador, las cargas de los CCDs se traducen a una matriz de números enteros. Luego el telescopio apunta a otra región y vuelve a realizar el procedimiento. Un ejemplo de imagen astronómica tomada por un cuadrante de un CCD de ZTF se puede ver en la Figura 2.9.

Como se ha mencionado anteriormente, los señales lumínicas llegan con distorsión a los CCDs. La **Point Spread Function (PSF)** es una función que describe como se distribuyen los fotones de una fuente puntual de luz en los CCDs debido a las distorsiones. En el área de procesamiento de señales, la PSF corresponde a la respuesta del impulso entre la fuente y los datos capturados del sensor. Reyes Jainaga (2019) ejemplifica en términos prácticos que una atmósfera turbulenta provoca que desde la Tierra se vea que las estrellas titilen mucho, lo cual se traduce en una una PSF más extendida. En cambio en un cielo más limpio con buenas condiciones atmosféricas la PSF es más concentrada. ZTF por su parte se encarga de limpiar el efecto de la PSF para cada CCD, para posteriormente entregar estos datos en tiempo real.

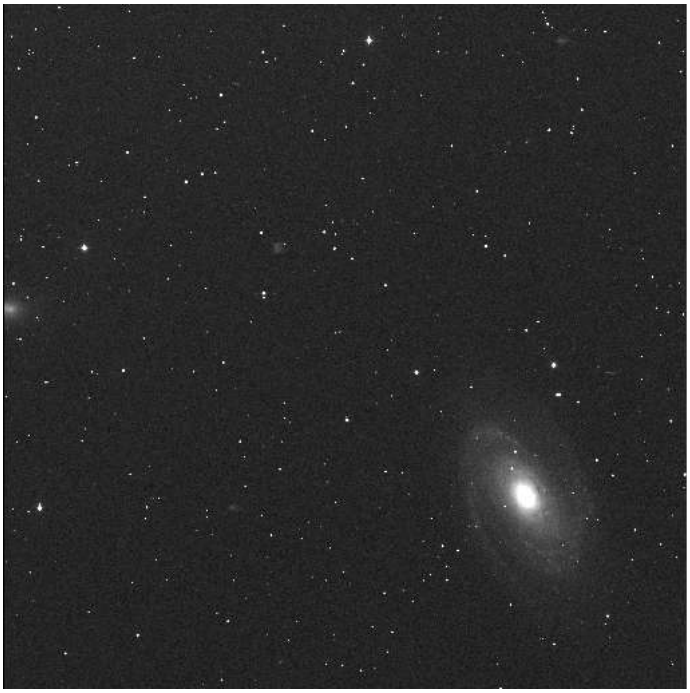


Figura 2.9: Galaxia Espiral M81 capturada por ZTF. Esta imagen de 3072x3080 píxeles, corresponde al primer cuadrante del sexto CCD en la banda *g*. Imagen capturada el día 27 de marzo del 2018. Imagen preprocesada con fines de visualización. Fuente: Zwicky Transient Facility

2.2.2 Procesamiento interno de ZTF

Para el procesamiento interno existe un sistema denominado ZTF Science Data System (ZSDS), cuyo funcionamiento cubre: la gestión de transferencia desde el telescopio; ingestión de datos en bruto; mantenimiento de *software*, *hardware* y bases de datos; *pipeline* de procesamiento; distribución en tiempo real de alertas, entre otras funciones que permiten la operación eficiente por parte de ZTF (Masci et al., 2018).

Según Masci et al. (2018) el sistema de ZSDS está diseñado para cumplir con un sistema de archivo y procesamiento que ofrezca productos de calidad científica lo más cercano al tiempo real. En 20 minutos el sistema debe ser capaz de recuperar las imágenes desde la cámara (10 minutos) y procesarlos (10 minutos) por observación. La idea está impulsada para descubrir objetos transientes de manera rápida y así desencadenar observaciones de seguimiento relevantes.

Técnicamente ZSDS procesa de manera concurrente las observaciones de la cámara. Para esto se utiliza un clúster de 66 nodos de cómputo. Cada nodo trabaja con las observaciones en un archivo de formato FITS[7] y se encargan de ejecutar la *pipeline* de

[7]Flexible Image Transport System (FITS) es un formato de facto en astronomía el cual puede contener bastante

procesamiento. La *pipeline* de procesamiento de ZSDS tiene el deber de generar distintos productos de datos en base al procesamiento, calibración y cómputo, como por ejemplo la obtención de imagen de ciencia, referencia y diferencia, ajustes de PSF, estimación de calidad de imagen, etc (Masci et al., 2018).

2.2.3 Extracción de eventos y generación de alertas

La extracción de eventos es una de las etapas más importantes para la generación de alertas en tiempo real, para esto se necesita buscar diferencias significativas en las imágenes recientemente capturadas con imágenes de referencia del universo. Las imágenes existentes en este proceso son:

- *science*: Corresponde a una exposición tomada en un instante t determinado.
- *reference*: Suele ser un *stack* de exposiciones representativas tomadas cada cierto tiempo. Esta permite conocer cómo es la zona que se está observando.
- *difference*: Es la diferencia entre una exposición de *science* y el *stack* de *reference*.

En la Figura 2.10 se muestran las tres imágenes para el tercer cuadrante del tercer CCD, como se logra apreciar hay una galaxia en la parte inferior derecha (imagen de *science*). En la imagen de *difference* no se puede apreciar información a simple vista, aún así existe una gran cantidad de información que sirve para gatillar una alerta.

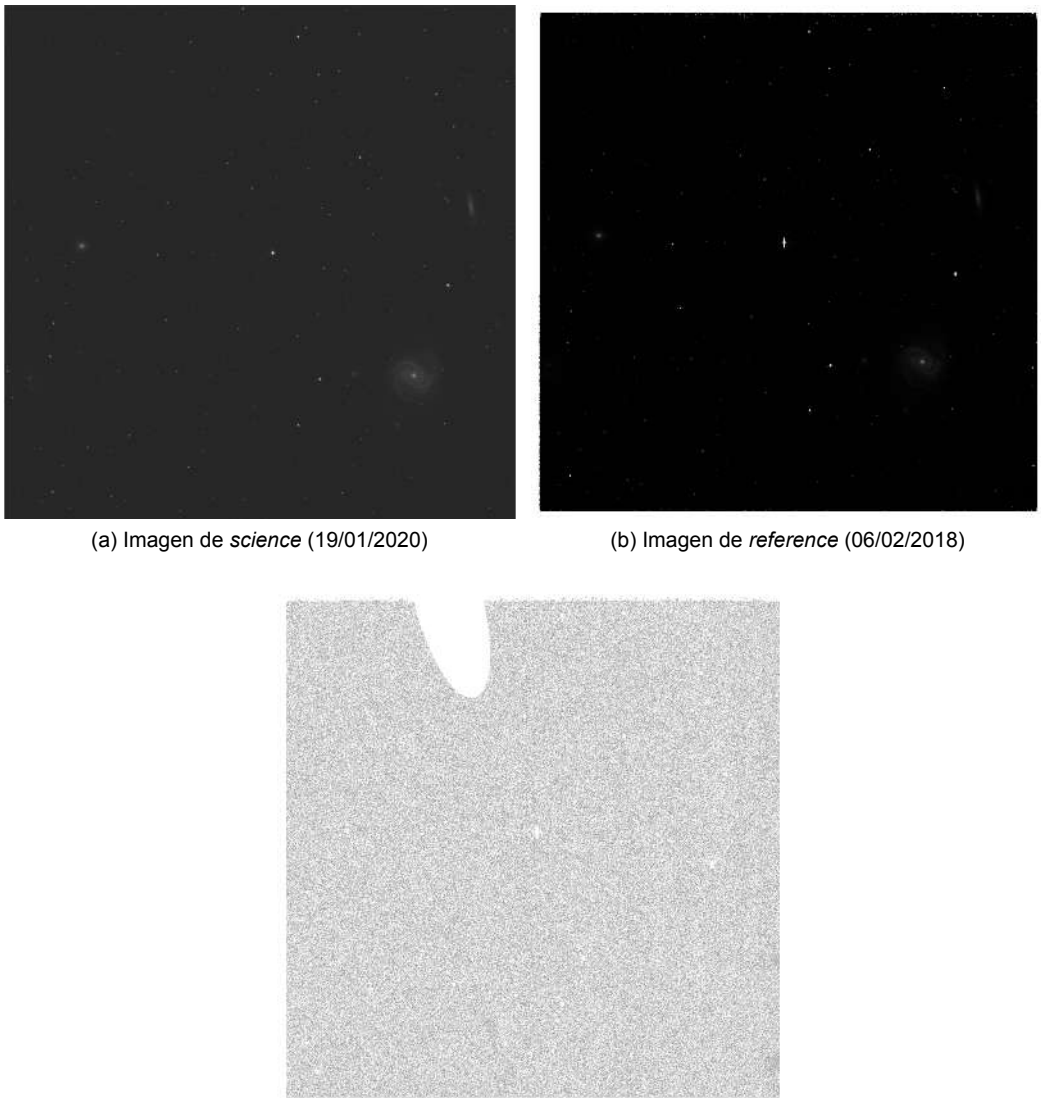
Dentro de las imágenes obtenidas, se hacen recortes centrados en eventos de interés que se identifican en la imagen de *reference*, por lo tanto es posible encontrar varios objetos en cada exposición de los CCDs. Para determinar que ha existido un evento interesante, ZTF utiliza la definición de alerta de LSST: si existe una diferencia mayor o igual de 5σ en las coordenadas ecuatoriales (RA, Dec) o en el flujo con respecto al cielo de referencia (Graham et al., 2019). El criterio considera σ como la desviación estándar de la respectiva medición.

Los recortes o estampillas, también conocidos como *stamps*, están centrados en el evento de interés. Cada recorte es de 63×63 píxeles y se le asocia un identificador al objeto avistado. En la Figura 2.11 se muestra la supernova ZTF20aaelulu[8], que está en la galaxia que se presenta en la Figura 2.10.

Una vez que se tiene las *stamps*, ZSDS ocupa un componente llamado ZTF Alert Distribution System (ZADS) que se encarga de distribuir en tiempo real las alertas generadas. En esta se empaquetan los recortes, posición en el cielo, magnitud, filtro de banda, entre otros. ZADS

información de una observación (imágenes, cubos de datos, metadatos, etc).

[8] `https://alerce.online/object/ZTF20aaelulu`



(a) Imagen de *science* (19/01/2020)

(b) Imagen de *reference* (06/02/2018)

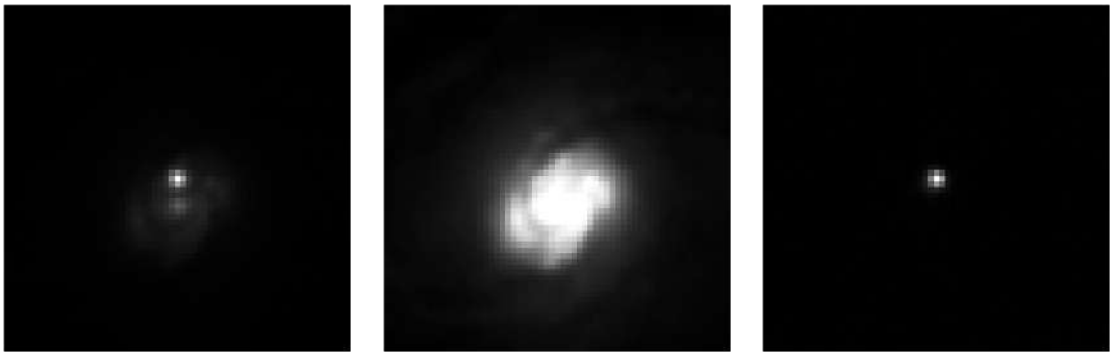
(c) Imagen de *difference* (19/01/2020)

Figura 2.10: Imágenes de *science*, *reference* y *difference* capturadas por ZTF. Fuente: Zwicky Transient Facility

provee la transmisión de una cantidad relativamente grande de datos: aproximadamente 1 millón de alertas por noche y cada una con un tamaño de mensaje de alrededor de 60 KB Patterson et al. (2018).

2.2.4 Distribución de datos públicos

Las alertas son retransmitidas por internet mediante Apache Kafka a distintos *brokers* astronómicos a lo largo del mundo. Apache Kafka es un *software* de código abierto que



(a) Recorte *science* (b) Recorte *reference* (c) Recorte *difference*

Figura 2.11: Recorte de ejemplo de objeto ZTF20aaelulu (supernova tipo Ic) captada por ZTF. Notar que la imagen de *science* la explosión de la supernova cambia la escala, dejando en negro gran parte de la galaxia que la aloja. Fuente: Zwicky Transient Facility

permite transmitir datos en tiempo real siguiendo la lógica productor/consumidor (Garg, 2013). Los **productores** encolan mensajes en distintas **particiones** de **tópicos**, en el caso de ZTF cada tópico corresponde a una noche de datos (e.g. ztf_public_20210811). Luego los *brokers* despliegan **consumidores** que desencolan los mensajes y post-procesan según sus intereses.

Es importante mencionar que sólo un subconjunto de datos son transmitidos públicamente. Una porción de observaciones de la banda *g* y *r* son transmitidos de manera pública, mientras que el resto de datos de la banda *i* se utilizan de manera privada por parte de ZTF (de vez en cuando realizan *data releases* donde ponen a disposición datos privados).

2.3 REDES NEURONALES ARTIFICIALES

Las redes neuronales artificiales (ANN) son modelos matemáticos que son utilizados para el problema de clasificación o regresión inspirados en el cerebro humano. El modelo matemático de una neurona fue propuesto por McCulloch & Pitts (1943), el cual sirvió como fuente de inspiración para otros modelos. Rosenblatt (1962) propone el perceptrón como algoritmo de aprendizaje para el modelo simplificado de una neurona, véase Figura 2.12.

La interconexión de unidades de perceptrón permite la generalización y construcción de modelos más complejos y sofisticados como lo son las redes neuronales superficiales, o también conocidas como redes neuronales poco profundas. Además de poseer una capa de entrada y salida, tienen capas intermedias llamadas capas ocultas. Para expresar matemáticamente una capa de una ANN, se utiliza la notación de matrices, como muestra la Ecuación 2.1.

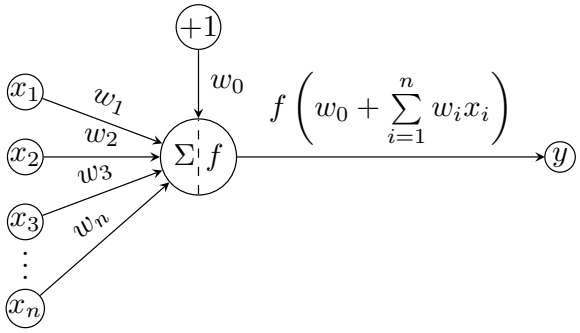
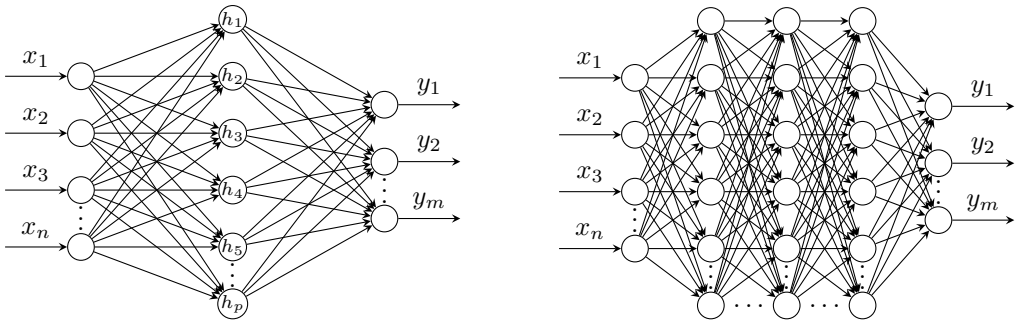


Figura 2.12: Representación gráfica de un perceptrón. Donde w_i corresponde a los pesos sinápticos que son ponderados con las entradas x_i, el resultado del producto es sometido a una suma más un término de sesgo w_0. Posteriormente el resultado pasa por una función de activación f dando como salida un valor y. Elaboración propia, 2021

$$\mathbf{y} = f(W^T\mathbf{x} + \mathbf{b}) \tag{2.1}$$

Donde $\mathbf{x} \in R^n$ representa las n entradas, $\mathbf{y} \in R^m$ corresponde a las m salidas, los pesos de cada neurona están en cada columna de $W \in R^{m \times n}$ y el sesgo de cada neurona estaría representado por $\mathbf{b} \in R^m$. La función de activación f se aplica a cada valor del vector resultante. En la Figura 2.13a se muestra una representación gráfica de una red neuronal con una capa oculta.



(a) *Single layer perceptron* (b) *Multi layer perceptron*

Figura 2.13: Representación gráfica de una MLP. En (a) se puede apreciar un esquema que posee una capa de entrada **x** con n neuronas, una capa oculta con p neuronas y una capa de salida **y** neuronas, por otro lado en (b) se tiene una generalización de una MLP. Fuente: Elaboración propia, 2021

Para la construcción de modelos más complejos, basta con apilar una cantidad determinada de capas, las capas que no reciben las entradas y no retornan una salida son llamadas capas ocultas. Éstas poseen unidades de neuronas no observables. Esta manera de conformar conexiones se realiza de tal forma, que las unidades de una capa se conectan con las unidades de las capas siguientes. Esta estructura recibe el nombre de *multi layer perceptron* (MLP), véase Figura 2.13b, cuya expresión matemática para un modelo con 2 capas ocultas se

puede apreciar en la Ecuación 2.2

$$\mathbf{y_1} = f_1(W_1^T \mathbf{x_1} + \mathbf{b_1}) \quad \wedge \quad \mathbf{y_2} = f_2(W_2^T \mathbf{y_1} + \mathbf{b_2}) \tag{2.2}$$

2.3.1 Funciones de activación

Anteriormente se ha mencionado el uso de funciones de activación en redes neuronales, éstas están encargadas de modificar o limitar la salida del valor adquirido en la neurona. Las funciones de activación tienen el deber de transmitir la información contenida en la neurona, que corresponde a la combinación lineal de los pesos y las entradas. Usualemente se utilizan funciones con forma de escalón suave, tales como la función sigmoide o la tangente hiperbólica, que se caracterizan por ser una aproximación diferenciable de la función escalón, por lo que se puede aplicar el algoritmo de *backpropagation* (Cybenko, 1989).

El problema de las funciones sigmoidales suele traer el efecto de *vanishing gradients*, ya que dichas funciones toman valores dentro de un rango acotado. Generalmente se da cuando los modelos tienen muchas capas y parámetros, donde el gradiente tiene que ser propagado a través de muchas neuronas, provocando que el aprendizaje se vuelva inviable ya que los parámetros al inicio de la red toman valores muy pequeños.

La función rectificadora o *Rectified Linear Units* (*ReLU*) disminuye el efecto de *vanishing gradients*, donde los valores negativos de la entrada los transforma en cero y tiene el comportamiento de la función identidad para valores positivos (véase Ecuación 2.3) (Maas et al., 2013).

$$f(x) = max(0, x) \tag{2.3}$$

Existen variantes de la función *ReLu*, como por ejemplo *Leaky ReLU* (Maas et al., 2013) cuya expresión está en la Ecuación 2.4, donde α es un valor positivo y pequeño (generalmente 0.01), o por otro lado *Parametric Relu* (He et al., 2015) donde el valor α no es un valor constante e igual para todas la unidades, sino que es un parámetro a optimizar en el entrenamiento de la red.

$$f(x) = max(\alpha x, x) \tag{2.4}$$

2.3.2 Ajuste de una red neuronal

Los valores de W y **b** son modificados durante la etapa de entrenamiento. Esta etapa corresponde a la búsqueda de los mejores valores para los parámetros W y **b**. Para realizar el entrenamiento de un modelo de redes neuronales, se requiere un conjunto de datos de entrenamiento, el cual permite ajustar los parámetros de la red. Para encontrar los mejores parámetros de la red se debe resolver el problema de optimización que se muestra en la Ecuación 2.5 mediante técnicas basadas en el descenso del gradiente; donde se tiene los parámetros θ^* a encontrar, el conjunto de datos de entrenamiento $\{(x_i, y_i)\}_{i=1}^N$ y una función de pérdida $\mathcal{L}$ la cual debe ser diferenciable.

$$\theta^* = argmin \quad \mathcal{L}(y_i, f(x_i)) \tag{2.5}$$

Los parámetros θ^* son ajustados iterativamente mediante el cálculo de la Ecuación 2.6; donde k es la iteración y α corresponde a la tasa de aprendizaje.

$$\theta_k = \theta_{k-1} - \alpha \nabla \mathcal{L}(y_i, f(x_i)) \tag{2.6}$$

La función de pérdida $\mathcal{L}$ es una función que permite evaluar la diferencia que existe entre los valores predichos $\hat{y}_i$ con los valores reales y_i de las observaciones utilizadas durante el entrenamiento. Dependiendo del problema a resolver, se escoge una función para encontrar los parámetros θ^*. En el caso del problema de regresión, usualmente se utiliza el error cuadrático medio (MSE), el cual se muestra en la Ecuación 2.7

$$\mathcal{L}_{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{2.7}$$

En un problema de clasificación se debe relacionar cada ejemplo recibido con una de las clases existentes en el conjunto de entrenamiento. Una de las formas para lidiar con la pertenencia de una observación a una clase es la codificación *one hot*. Para M clases se tiene un vector de largo M, donde cada posición indica la pertenencia a la m-ésima clase. En este vector todos los valores son ceros a excepción del m-ésimo valor el cual es un 1. Para este tipo de problemas resulta útil la función objetivo de entropía cruzada. A partir de la codificación *one hot* resulta simple definir la función de pérdida, véase Ecuación 2.8.

$$\mathcal{L}_{crossentropy} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{M} (y_{ic} \cdot log(\hat{y}_{ic})) \tag{2.8}$$

Además el uso de una función *softmax* en la salida de la red, permite forzar las salidas

como una distribución de probabilidades. Solo se debe aplicar al vector de salida $z_i \in \mathcal{R}$, veáse Ecuación 2.9.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}} \tag{2.9}$$

Al poseer una mayor cantidad de neuronas y capas el cálculo del gradiente puede ser costoso, sin embargo se puede calcular eficientemente utilizando *backpropagation* (respetando la regla de la cadena para redes neuronales), el cual es un algoritmo que propaga el error desde la salida del modelo hasta la primera capa de la red neuronal (Werbos, 1994).

Para modelos que tengan más de una capa, la función de pérdida es no convexa en los parámetros de la red neuronal, es decir, la función puede tener más de un mínimo o máximo. Esto provoca que la solución al problema de optimización converja a un mínimo o máximo local. Es por esta razón que existen otros métodos que permiten acelerar este proceso de cómputo, como lo es *stochastic gradient descent with momentum* (Qian, 1999). Por otro lado existen variaciones que poseen una tasa de aprendizaje α adaptativo, como lo son Adagrad (Duchi et al., 2011), Adam (Kingma & Ba, 2014), entre otras.

2.3.3 Redes neuronales profundas

Hochreiter (1998) presentó uno de los problemas más significativos de las ANN. Lo definió como el problema de desvanecimiento de gradiente. Este dice que en algunos casos el gradiente se va desvaneciendo a valores muy pequeños, de tal manera que el peso no cambia de valor. Esto impide que ANN con más de 3 capas entrenen eficazmente, ya que los pesos no serían actualizados a medida que avance el entrenamiento, convergiendo rápidamente a valores que provoquen un mal desempeño del modelo. Este hecho provocó que el campo de las ANN se estancara por años.

Posteriormente Hinton et al. (2006) propuso un modelo de red que usa una *Restricted Boltzmann Machine* (RBM) para aprender una jerarquía de características de un nivel a la vez, por lo tanto sus modelos son extractores de características efectivos sobre datos estructurados de alta dimensión. Esto permitió que los pesos de la red neuronal en un comienzo estén configurados de manera inteligente. Lo propuesto por Hinton fue un gran avance para el aprendizaje profundo, ya que en su investigación logró entrenar modelos con más de 3 capas de manera exitosa; esto significó la apertura para nuevos desafíos en el campo del *deep learning*, debido a que prueba que es posible entrenar arquitecturas profundas de manera efectiva (Bengio et al., 2013).

Los métodos de aprendizaje profundo son útiles cuando se dispone de una gran

cantidad de datos y un gran poder de cómputo. Por otro lado, cuando la etapa de extracción de características de los datos no está muy clara, las arquitecturas profundas permiten aprender las características directamente de los datos. Actualmente existen diferentes métodos para lidiar con el problema de desvanecimiento del gradiente, también para soportar distintos formatos de entrada, además de poder entrenar modelos a partir de condiciones propuestas por los investigadores. Es por estas razones la popularidad de este tipo de método para resolver distintos problemas como la clasificación, generación de datos, agrupamiento, entre otros.

2.3.4 Redes neuronales convolucionales

Uno de los grandes desafíos en el aprendizaje de máquina es la recuperación de información útil contenida en imágenes, es por esta razón que diversos científicos han colaborado para comprender como funciona la visión y dar paso a lo que es la visión computacional.

En un principio Hubel & Wiesel (1959) estudiaron cómo funciona la corteza visual de un cerebro animal, demostrando que existen células que son responsables de la detección de bordes y otras de la selectividad de orientación, además propusieron un modelo basado en estas células para el reconocimiento de patrones. Este estudio inspiró la investigación de Fukushima & Miyake (1982), el cual propone una red neuronal multi-capa jerárquica llamada Neocognitrón para el reconocimiento de patrones visuales.

Durante esa misma época se comenzó a aplicar la operación de convolución para analizar y encontrar patrones en señales digitales (Atlas et al., 1987). Años más tarde LeCun et al. (1989) incorporan el método de *backpropagation* para entrenar el sistema de manera correcta, la cual pudo reconocer dígitos escritos a mano de manera efectiva. Este hallazgo da paso a las redes neuronales convolucionales (CNN). A pesar de tener una estructura robusta y buen desempeño para algunos problemas, las CNN no podían escalar, es decir, se necesitaba de una gran cantidad de datos y recursos informáticos para funcionar de manera eficiente para imágenes de mayor resolución, durante años funcionaron solo para imágenes pequeñas.

No fue hasta el año 2010 en que Cireşan et al. (2010) demostraron que las CNN se pueden entrenar correctamente y rápidamente utilizando GPU, su modelo obtuvo resultados impresionantes en comparación a modelos propuestos en años anteriores para el problema de clasificación del conjunto de datos de MNIST[9]. Un par de años después, Krizhevsky et al. (2012) demostró el poder de las CNN ganando un desafío de clasificación de imágenes de gran resolución llamado *“ImageNet Large Scale Visual Recognition Challenge 2012”*.

[9]Modified National Institute of Standards and Technology es un gran conjunto de datos de imágenes de dígitos en manuscrita.

Por lo tanto las CNN se originan a partir de la mejora de los primeros modelos inspirados en la corteza visual de los animales, cuyo funcionamiento se basa en limitar el funcionamiento de la arquitectura de las MLP, acotando la conectividad de cada neurona a una pequeña porción de la capa anterior (Hubel & Wiesel, 1959; Fukushima & Miyake, 1982; LeCun et al., 1989).

Las CNN han demostrado funcionar de manera exitosa en la clasificación de datos que están espacialmente correlacionados como lo son las imágenes (LeCun et al., 1989; Krizhevsky et al., 2012) o para la clasificación de datos volumétricos (Qi et al., 2016; Bogacz & Mara, 2020); además de reconocer patrones en datos que están temporalmente correlacionados como los es el audio (Abdel-Hamid et al., 2014) o la predicción en series de tiempo (Tsantekidis et al., 2017).

Generalmente las arquitecturas de las CNN se organizan de la siguiente forma: (1) Capas convolucionales que se encargan de realizar la operación de convolución, obteniendo un vector que contiene la información relevante de la entrada; (2) Capas de *pooling* que reducen progresivamente el tamaño espacial de la representación, reduciendo así la cantidad de parámetros y cómputo de la red; (3) Finalmente con la entrada reducida a un vector, éste pasa por una capa *fully connected*[10] (FC) para buscar una solución al problema de clasificación, regresión, segmentación, etc (Cireşan et al., 2010; Krizhevsky et al., 2012).

En la Figura 2.14 se muestra un ejemplo de CNN que recibe como entrada una imagen, ésta se introduce por una serie de filtros de la capa convolucional, para luego pasar por una capa de *pooling* que reduce aún más la entrada, al final de la CNN se encuentra una capa densa (o FC) que se encarga de realizar la predicción.

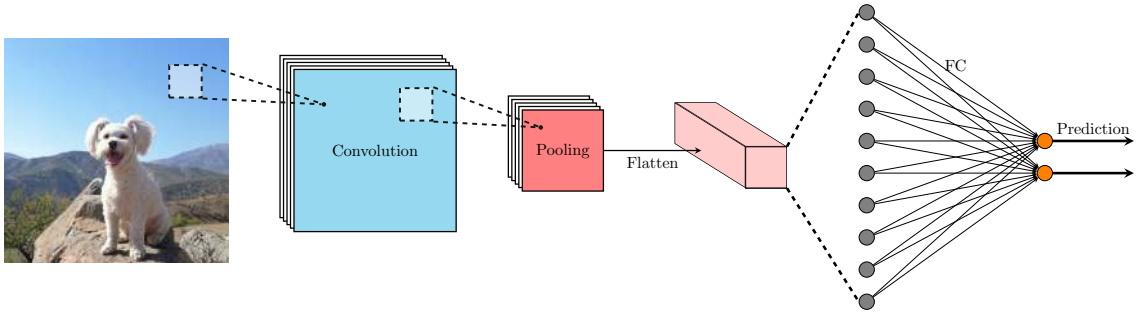


Figura 2.14: Ejemplo de arquitectura de una red convolucional. Generalmente la entrada corresponde a una imagen o audio, ésta se opera mediante la convolución con diferentes filtros que detectan bordes, líneas, relieves, etc. Posteriormente se reduce la información mediante una capa de *pooling*, la cual recupera valores representativos de la entrada convolucionada. Finalmente, el resultado se transforma en un vector, donde éste último es la entrada a una capa FC para resolver alguna tarea de clasificación, regresión o segmentación. Fuente: Elaboración propia, 2021

Las capas de convolución se encargan de aplicar distintos filtros de un tamaño

[10]Las capas FC conectan cada neurona de una capa con una capa subsiguiente. En la practica es una MLP tradicional

determinado a la entrada. En la Ecuación 2.10 se aprecia el cálculo en una capa de este tipo:

$$y = f(W^T * x + b) \tag{2.10}$$

donde $*$ es el operador de convolución, $x \in \mathcal{R}$ es la entrada a la capa, $y \in \mathcal{R}$ es la salida de la capa, W es un conjunto de filtros que se utilizan para realizar la convolución con la entrada, $b \in \mathcal{R}$ corresponde a el *bias* de cada filtro y $f()$ es una función de activación. El filtro se desplaza con un valor definifo, llamado *stride*. En la Figura 2.15 se muestra la convolución a una matriz I de 7×7 con un *kernel* K de tamaño 3×3 y un *stride* de 1. La salida $I * K$ es conocida como *feature map*, es más pequeña que los datos de entrada y posee información combinada de la entrada (Zhang et al., 1990).

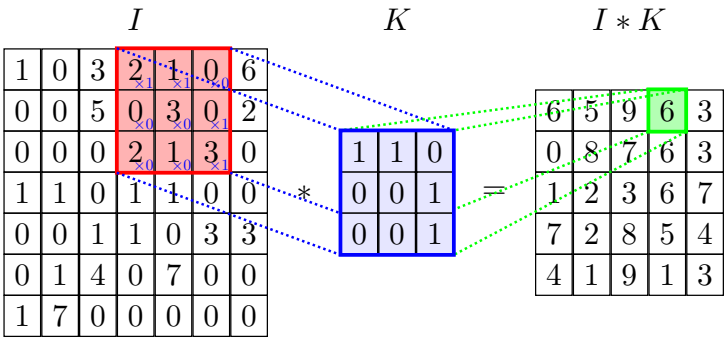


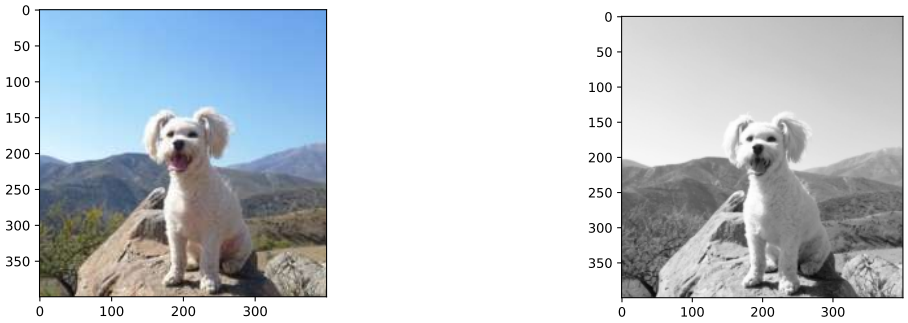
Figura 2.15: Operación de convolución en una matriz. Fuente: Elaboración propia, 2021

En el caso de las imágenes digitales es posible aplicar la operación de convolución para filtrar información relevante que se quiera rescatar. En la Figura 2.16 se muestra un ejemplo de imagen digital, donde se presenta una imagen en RGB y ésta misma en escala de grises. Generalmente se utilizan tensores[11] para representar y operar cada píxel de la imagen.

El funcionamiento de las CNN indica que en la etapa de entrenamiento, el modelo realiza varias convoluciones simultáneamente con distintos *kernels*, de manera que dichos *kernels* reconozcan características espaciales que sean relevantes. Finalmente se obtiene una serie de *feature maps*, donde cada uno recupera características particulares de la entrada (por ejemplo bordes, formas, eliminación de detalles, etc).

Una propiedad que permite el buen funcionamiento de la convolución en el área de las CNN, corresponde a la invariancia a las traslaciones, es decir, si se tiene cierto elemento en una imagen y ésta es convolucionada con un filtro en específico, el resultado será un *feature map* particular. Si los píxeles de la imagen original se trasladan a otro sector, ocupando otra posición en el espacio de la imagen, al convolucionar la imagen modificada con el filtro anterior, el *feature map* será el mismo, con la diferencia que va a estar trasladado al igual que la entrada modificada.

[11] Un tensor corresponde a la representación vectorial de n-dimensiones. Éstos son capaces de realizar un gran cantidad de operaciones entre sí y son utilizados en distintas ramas de la ingeniería.



(a) Imagen en RGB

(b) Imagen en escala de grises

Figura 2.16: Ejemplo de imagen digital de 400×400 píxeles. En (a) se puede apreciar una imagen en el estándar RGB (sigla en inglés de *red*, *green*, *blue* que representan 3 canales, por lo tanto el tensor que representa la imagen es de $3 \times 400 \times 400$), cada píxel corresponde a 3 valores enteros de 8 bits (256 niveles), se pueden combinar entre ellos y generar hasta 16 millones de colores. En (b) se encuentra una imagen en escala de grises, donde cada píxel consiste de un número que varía entre cero y uno (se representa en un solo canal, por lo tanto el tensor que representa la imagen es de $1 \times 400 \times 400$). Fuente: Elaboración propia, 2021

En la Figura 2.17 y Figura 2.18 se muestran ejemplos de distintos *kernels* que convolucionan la imagen de entrada I, es posible observar que dependiendo de los valores del *kernel* la salida resultante tendrá cierta información. Por ejemplo en la Figura 2.17, la salida muestra algunos bordes de los elementos de la imagen, eliminando las partes más planas de la imagen. Por otro lado el *kernel* K_2 de la Figura 2.18 solo suaviza la imagen, funcionando como un filtro pasa bajo.

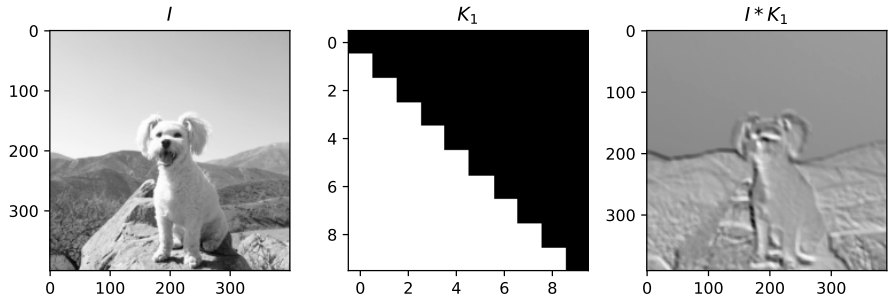


Figura 2.17: Ejemplo de convolución con *kernel* de 10×10 diagonal. Los píxeles negros y blancos tienen los valores -1 y 0 respectivamente. Fuente: Elaboración propia, 2021

El *pooling* por su parte se emplea para reducir la dimensión de la representación y se utiliza como regularizador. Esta operación se encarga de calcular un valor representativo para cada región de la imagen, si se toma el máximo valor de la región la operación se nombra *max pooling* y si se ocupa el valor promedio se llama *average pooling*. Generalmente se utilizan capas de *pooling* de tamaño 2×2 que no se solapen (vale decir un *stride* de 2). En la Figura 2.16 se

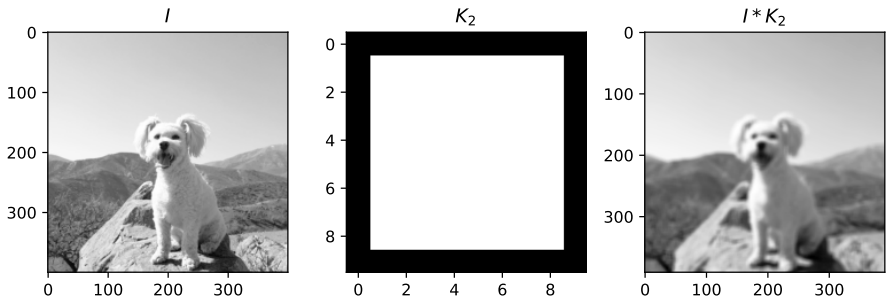


Figura 2.18: Ejemplo de convolución con *kernel* de 10×10 vertical. Los píxeles negros y blancos tienen los valores -1 y 1 respectivamente. Fuente: Elaboración propia, 2021

muestra un ejemplo de *max pooling*, donde para cada región de 2×2 de la imagen original se escoge el máximo valor.

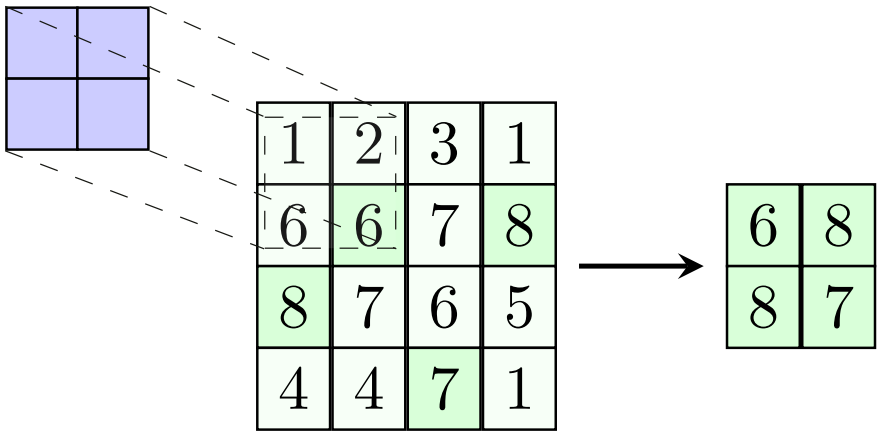


Figura 2.19: Ejemplo de *max pooling* de 2×2. Se utiliza una ventaan de 2×2 y un *stride* de 2. Al lado izquierdo se muestra la ventana que va pasando por la entrada, al centro se encuentra la entrada que corresponde a una matriz de 4×4 y al lado derecho se encuentra el resultado de la operación. Fuente: Elaboración propia, 2021

Después de las capas convolucionales y de *pooling*, están las capas *fully connected* que son las encargadas de realizar la predicción del problema a resolver. Las neuronas en una capa FC tienen conexiones con todas las activaciones de la capa anterior, al igual que en las MLP. Generalmente la mayor parte de los parámetros de una CNN se encuentran en las capas FC, por lo tanto es importante fijarse en la cantidad de neuronas en cada una de estas capas al momento de diseñar la arquitectura de la red, dado que un amplio número de neuronas podría incidir de manera directa en los tiempos de entrenamiento o en el uso de recursos computacionales.

2.3.5 Capas de regularización

Algunas técnicas que sirven para mejorar los resultados del modelo durante el entrenamiento corresponden al ***dropout***: apagar cierto porcentaje de neuronas mientras el modelo realiza el ajuste de sus parámetros (Srivastava et al., 2014); ***batch normalization***: realizar un escalado de los datos de entrada por lotes, el cual además permite realizar un entrenamiento más rápido (Ioffe & Szegedy, 2015); o también aplicar penalizaciones a la función de pérdida que se desea optimizar.

2.3.6 Métricas en el problema de clasificación

Para evaluar el desempeño de un modelo de clasificación, existen formas para medir la calidad de los resultados. La medida estándar para la clasificación es el ***accuracy*** que mide cuantos aciertos tiene el modelo en el total de datos, la ***precision*** indica la fracción de instancias relevantes entre las instancias recuperadas, midiendo la dispersión de los aciertos. El ***recall*** indica la sensibilidad del modelo, es decir, la proporción de casos positivos que fueron correctamente identificados y el ***f1 score*** combina la *precision* y *recall* para tener una media más general del modelo, y suele ser útil cuando el problema a resolver tiene distribución de clases desbalanceadas. Estas medidas varían entre 0 y 1, y mientras más cercano a 1 sea el valor, indica que el modelo es mejor. A continuación se presenta las expresiones de cada una de ellas:

$$\textit{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.11}$$

$$\textit{precision} = \frac{TP}{TP + FP} \tag{2.12}$$

$$\textit{recall} = \frac{TP}{TP + FN} \tag{2.13}$$

$$f_1 = 2 \cdot \frac{\textit{precison} \cdot \textit{recall}}{\textit{precison} + \textit{recall}} \tag{2.14}$$

Donde:

- TP: Corresponde a los verdaderos positivos (*true positive*), es decir, cuando la clase real es verdadera y la predicha también.

- TN: Corresponde a los verdaderos negativos (*true negative*), es decir, cuando la clase real es falsa y la predicha también.
- FP: Corresponde a los falsos positivos (*false positive*), es decir, cuando la clase real es falsa y la predicha es verdadera.
- FN: Corresponde a los falsos negativos (*false negative*), es decir, cuando la clase predicha es verdadera y la predicha es falsa.

Una de las herramientas visuales para evidenciar el desempeño de los modelos obtenidos, es la **matriz de confusión**, donde cada columna de la matriz representa la cantidad de predicciones de cada clase y cada fila representa las clases reales de las instancias. En términos prácticos permite ver la tasa de aciertos y errores que está teniendo el modelo.

2.4 ESTADO DEL ARTE: CLASIFICACIÓN UTILIZANDO IMÁGENES DE *ASTRONOMICAL SURVEYS*

La clasificación de datos astronómicos mediante métodos de aprendizaje automático es un área que ha sido explorada recientemente. Abundan modelos que utilizan diferentes mediciones u observaciones de un objeto para realizar la predicción (e.g curva de luz, características del objeto, estampillas, etc). Dado que el trabajo propuesto abarca el uso de estampillas, se ha realizado una revisión de la literatura que contempla principalmente este tipo de entrada.

Como se ha mencionado en la Subsección 2.3.3 las técnicas de aprendizaje profundo son un ejemplo de soluciones basadas en el análisis de datos, que han sido capaces de extraer características automáticamente y han demostrado ser exitosas para problemas de clasificación; en particular las redes neuronales convolucionales se han aplicado a datos espacialmente correlacionados como las imágenes (Fukushima & Miyake, 1982).

Recientemente las redes convolucionales se han aplicado en el área de la astronomía, por ejemplo para la detección de objetos reales o *bogus* (Cabrera-Vives et al., 2016), cómputo de fotometría (Kimura et al., 2017), detección de exoplanetas (Shallue & Vanderburg, 2018), etc.

El uso de las estampillas para resolver el problema de clasificación en tiempo real, tiene la ventaja de que se entrega toda la información del evento y del contexto al modelo, lo que resulta más conveniente en comparación con los modelos que reciben como entrada la curva de luz de algún objeto, dado que el cálculo que permite obtener características de ésta son costosos

computacionalmente (Charnock & Moss 2017; Martínez-Palomera et al. 2018; Sánchez-Sáez et al. 2021).

Uno de los enfoques que iniciaron el uso de métodos de aprendizaje automático en la clasificación de imágenes astronómicas, corresponde al trabajo realizado por Bailey et al. (2007), el cual aplica técnicas a las imágenes de diferencia para encontrar supernovas. Los modelos propuestos corresponden a árboles de decisión, *Random Forest* (RF), *Support Vector Machine* (SVM) y *Artificial Neural Network* (ANN), los que son capaces de discriminan ciertas clases con mejor rendimiento que el filtrado de algunos metadatos de la observación. Los resultados de estos modelos proporcionan menos falsos positivos y reducen el número de objetos que nos son supernovas en un factor 10 en el programa *Nearby Supernova Factor*.

Posteriormente Bloom et al. (2012) realiza un modelo de RF para datos de *Palomar Transient Factory* (PTF), el cual se caracteriza por ser de los primeros modelos en funcionar en el contexto de clasificación en tiempo real y capaz de descubrir diversos candidatos a supernovas.

Wright et al. (2017) presenta un modelo basado en *deep learning*, específicamente una CNN que es capaz de distinguir entre asteroides, *bogus* y supernovas. El trabajo realizado disponibiliza todo un sistema integrado para la detección de supernovas en datos de PanSTARRS1, en donde se combinan resultados de humanos con los del modelo de aprendizaje profundo.

Cabrera-Vives et al. (2017) propone una arquitectura basada en CNN llamada Deep-HiTS, un modelo que aplica la técnica de ***rotational invariance*** a la entrada. Esta técnica permite lidiar con la orientación aleatoria de las estampillas. Básicamente *rotational invariance* rota la entrada original en $90°$, $180°$ y $270°$ (ver Figura 2.20). El modelo fue entrenado con datos reales proveniente de HiTS. El uso de esta arquitectura redujo las supernovas mal clasificadas en aproximadamente un factor de $1/5$. Además, este trabajo corresponde a la primera vez que se utilizan CNN para detectar eventos astronómicos transientes.

Para el año 2019, Carrasco-Davis et al. (2019) propone un modelo RCNN el cual utiliza datos simulados de HiTS para realizar la predicción de objetos astronómicos. Este enfoque evita el cálculo de curvas de luz o imágenes de diferencia. El modelo de RCNN presenta varias ventajas en un escenario de clasificación de *streaming* de alertas astronómicas, por ejemplo la reducción del preprocesamiento de datos y evaluación más rápida. Este modelo, es el primer acercamiento al uso de una secuencia de estampillas para la tarea de clasificación.

Recientemente Carrasco-Davis et al. (2020), desarrolla un modelo que combina la arquitectura de Deep-HiTS con el uso de metadatos de los objetos astronómicos (véase Figura 2.21). En esta ocasión se trabaja con alertas de ZTF y se disponibiliza el modelo en la *pipeline* de ALeRCE, por lo tanto, es capaz de clasificar en tiempo real. Los resultados obtenidos han sido sorprendentes y ha permitido a ALeRCE reportar una gran cantidad de supernovas a TNS.

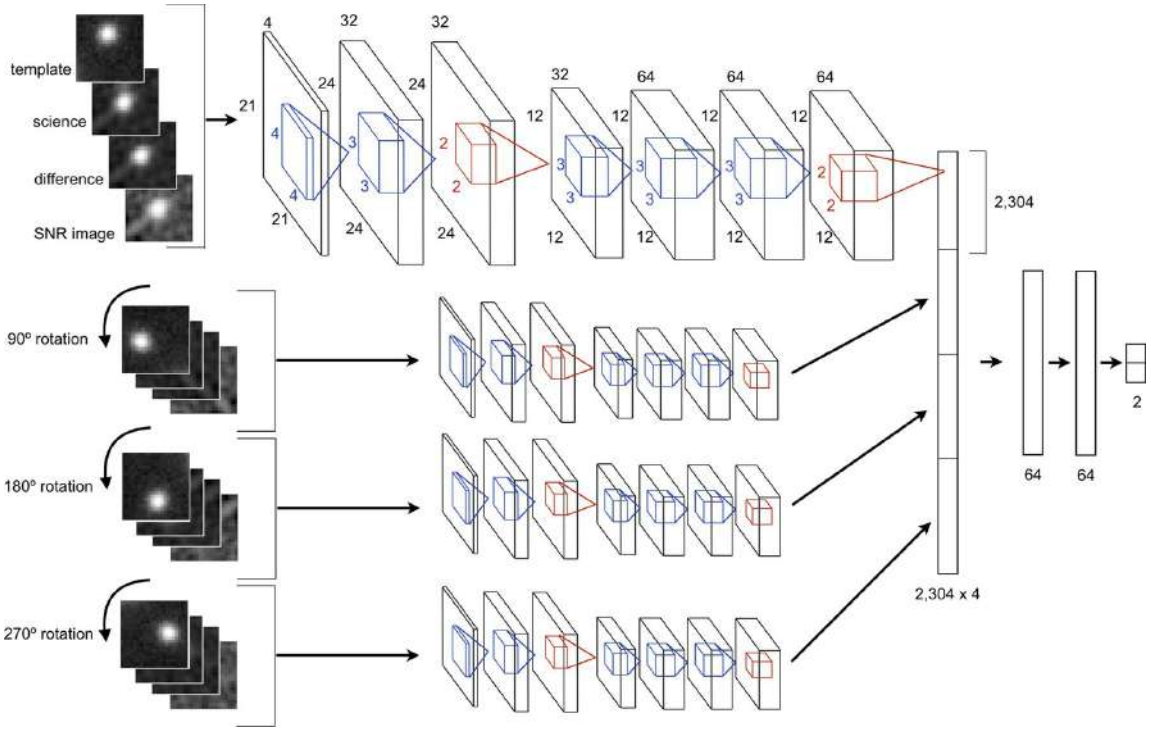


Figura 2.20: Arquitectura de Deep-HiTS. El modelo recibe como entrada una imagen de 4 canales (*template, science, difference* y *SNR image*) en donde se rota la entrada en $0°$, $90°$, $180°$ y $270°$. Posteriormente se concatenan los resultados y se realiza la predicción entre 2 clases - Fuente: Cabrera-Vives et al. (2017)

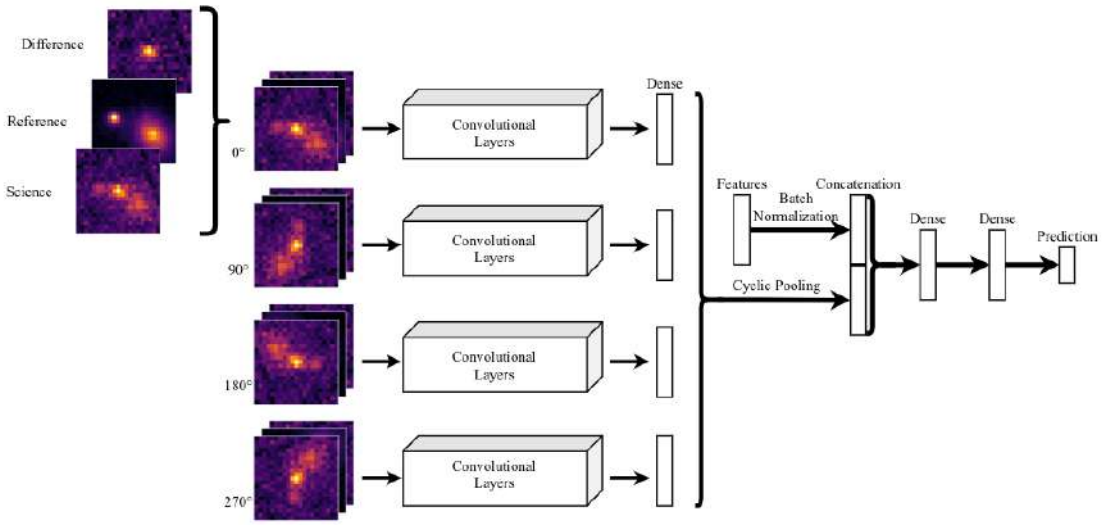


Figura 2.21: Arquitectura de *stamp classifier*. El modelo recibe como entrada una imagen de 3 canales (*difference, reference* y *science*) en donde rota se la entrada en $0°$, $90°$, $180°$ y $270°$. Posteriormente se concatenan los resultados con los metadatos de las alertas y se realiza la predicción entre 5 clases - Fuente: Carrasco-Davis et al. (2020)

Los modelos de Cabrera-Vives et al. (2017) y Carrasco-Davis et al. (2020) presentan ciertas debilidades en cuanto a los datos utilizados, recordar que los *astronomical surveys* son

capaces de recolectar desde decenas a centenas de alertas por objeto, lo que provoca que dichos modelos no complementen el conocimiento con alertas que no sean la primera por objeto. Al utilizar más de una alerta para clasificar, permitiría obtener conocimiento de la evolución en el tiempo de ciertos objetos o eventos en el universo.

Si bien la arquitectura de RCNN de Carrasco-Davis et al. (2019) utiliza más de una estampilla por objeto astronómico, éste fue entrenado con datos simulados de HiTS, por lo que los datos cumplen con ciertas condiciones ideales que no necesariamente se dan en la realidad. Sin embargo logra demostrar que el uso de una secuencia de imágenes permite distinguir entre una mayor cantidad de clases.

La revisión de la literatura indica que una arquitectura que incluya la evolución en el tiempo de las estampillas, implica un buen rendimiento en cuanto a la tarea de clasificación de objetos astronómicos, ya que en las estampillas se aprecia el contexto y forma del astro. La ausencia de modelos que utilicen más de una estampilla y datos asociados a cada alerta astronómica señala ser un campo inexplorado, que puede ser un gran aporte a la comunidad astronómica y de alto impacto.

En base a los artículos científicos revisados, se ha construido una tabla que expone los avances respectos a esta área astronómica (véase Tabla 2.1). En ésta se aprecia como al pasar el tiempo los modelos son más robustos y se van acomodando a las necesidades de los astrónomos.

Referencia	Año	Modelo	Fuente de datos	# clases	Clases	Comentarios
Bailey et al. (2007)	2007	ANN, SVM, RF	PTF	2	*bogus*, *real*	Abarca la clasificación a partir de la diferencia entre la imagen de ciencia y referencia. Básicamente se utilizan parámetros como importancia estadística, forma, movimiento, etc. Aplica diversas formas de poder clasificar, utilizando cortes en umbrales, métodos clásicos y ensambles de éstos. Se llega a la conclusión de que la utilización de métodos como ANN, SVM y RF funciona mejor que cortes de umbrales en ciertos parámetros.
Bloom et al. (2012)	2012	RF	PTF	2	*bogus*, *real*	Se entrena un RF con 43 características disponibles. Se experimenta la clasificación entre 5 clases y se propone el uso de métodos de *machine learning* para la automatización de *follow up*. Se reporta la importancia de las característica de entrada para cada clase.
Wright et al. (2017)	2017	CNN	PanSTARRS1	3	*asteroid*, *bogus*, *real*	Se propone un enfoque que mezcla el uso de aprendizaje automático e inspección humana, dado que aún existe la supervisión para declarar un objeto como interesante. El propósito de este enfoque, es preparar modelos para el descubrimiento de transientes en tiempo real.
Cabrera-Vives et al. (2017)	2017	CNN	HiTS	2	*bogus*, *real*	Corresponde a un modelo de red convolucional que incorpora *rotation invariant* a las imágenes de entrada, se denomina Deep-HiTS. El modelo recibe imágenes de 21×21 píxeles en 4 canales. Se obtienen resultados de 0.9945 ± 0.003 en *accuracy* para el conjunto de prueba. Por otro lado, se comparan los resultados con un modelo de RF, se concluye que CNN muestra mejores resultados.
Carrasco-Davis et al. (2019)	2019	RCNN	HiTS	7	*asteroid*, CEP, E, Galaxy, Non Variable, RRL, SN	Corresponde a una red recurrente convolucional, en donde básicamente utiliza una secuencia de imágenes para poder funcionar. Además realiza una simulación de datos mediante los datos de HiTS, con estos datos sintéticos entrena el modelo y obtiene un *accuracy* de 0.9530 ± 0.002 en un conjunto de prueba.
Carrasco-Davis et al. (2020)	2020	CNN	ZTF	5	AGN, *asteroid*, *bogus*, VS, SN	Este modelo está basado en Deep-HiTS, por lo tanto utiliza la técnica de *rotation invariant* además de metadatos de la alerta. Los resultados obtenidos llegan a 0.9410 ± 0.004 en *accuracy* en un conjunto de prueba balanceado.

Tabla 2.1: Cuadro comparativo de artículos revisados. Fuente: Elaboración propia, 2021

CAPÍTULO 3. CONJUNTO DE DATOS

ZTF entró en funcionamiento el 01/06/2018 (UTC) y cada noche ha transmitido cientos de miles de alertas que son visibles desde el hemisferio norte. A la fecha se han avistado más de 61 millones de objetos y se cuenta con más de 222 millones de alertas transmitidas en tiempo real.

Las alertas astronómicas de ZTF están serializadas en formato AVRO[1] y son transmitidas mediante Apache Kafka. En este capítulo se presenta cómo son los datos de ZTF y cómo se agruparon para conformar un conjunto de datos para el desarrollo de esta tesis. Los datos utilizados corresponden los datos disponibles a partir del *stream* público de ZTF[2].

3.1 DESCRIPCIÓN Y CARACTERÍSTICAS DE LOS DATOS

3.1.1 Contenido de una alerta

Las alertas que transmite ZTF tienen un esquema[3] establecido, es decir, los datos y tipos de datos que se transmiten ya están definidos (esto permite una rápida serialización/deserialización).

Tabla 3.1: Contenidos generales de las alertas de ZTF

Campo	Descripción
schemavsn	Versión del esquema de la alerta
publisher	Origen de la alerta
objectId	Identificador único del objeto de la alerta
candid	Identificador único de la alerta
candidate	Metadatos de la alerta
prv_candidates	Registros de posibles candidatos del objeto (últimos 30 días)
cutoutScience	Recorte de imagen científica
cutoutTemplate	Recorte de la imagen de referencia
cutoutDifference	Recorte de imagen de diferencia

Fuente: Elaboración propia, 2021

El esquema de las alertas se muestra en la Tabla 3.1, de la cual se puede destacar que:

[1] Serialización de la estructura de datos en binario, sigue la lógica llave/valor. Es parecido al formato JSON pero más liviano.

[2] Los datos históricos del *stream* público se encuentran en `https://ztf.uw.edu/alerts/public/`.

[3] Esquema de ZTF: `https://zwickytransientfacility.github.io/ztf-avro-alert/schema.html`

- La versión del esquema de la alerta mediante el campo *schemavsn*, ya que con el tiempo el esquema ha ido cambiando de acuerdo a los avances por parte de ZTF. Actualmente van en la versión v3.3, esto quiere decir que las primeras alertas no tienen los mismos campos que las alertas más recientes.
- Los identificadores *objectId* permite reconocer el objeto avistado, mientras que *candid* permite identificar la alerta distribuida.
- Los recortes *cutoutScience*, *cutoutTemplate* y *cutoutDifference* que corresponden a las imágenes astronómicas.
- El campo *candidate* que corresponde a otro segmento de llave/valor que posee información de la alerta; más detalle en Subsección 3.1.2.

3.1.2 Metadatos

En el campo *candidate* se tiene la información de más de 100 campos que están asociados a la observación realizada: condiciones de observación, cuándo se obtuvo la alerta, filtro utilizado, etc. Para Carrasco-Davis et al. (2020) resultó útil utilizar los campos que se muestran en la Tabla 3.2 para la realización de *stamp classifier*. Es importante mencionar, que no se utilizan todos los campos de las alertas, dado que pueden introducir algún sesgo a los resultados del modelo (e.g el metadato *rb* indica un puntaje de si un objeto es un *bogus*, esto está calculado por ZTF e implicaría depender del cálculo que ellos proveen).

A partir de los metadatos es posible calcular nuevas características que ayudan a discriminar entre clases, como las siguientes (Carrasco-Davis et al., 2020):

- Coordenadas eclípticas: Latitud y longitud de la eclíptica calculadas a partir de las coordenadas *RA* y *Dec* del candidato.
- Coordenadas galácticas: Latitud y longitud galácticas calculadas a partir de las coordenadas *RA* y *Dec* del candidato.
- Cantidad aproximada de *non-detections*: Número aproximado de observaciones en la posición del candidato, con una señal inferior a $signal/noise \approx 3$. Corresponde a la resta de *ncovhist* y *ndethist*.

Tabla 3.2: Lista de metadatos utilizados para el desarrollo de *stamp classifier* de ALeRCE

Metadato	Descripción
sgscore{1, 2, 3}	*Score* de estrella/galaxia de la {primera, segunda, tercera} fuente más cercana del catálogo de PanSTARRS1 $0 \leq sgscore \leq 1$, donde un valor cercano a 1 implica una mayor probabilidad de ser estrella, se asigna -999 cuando no hay fuente.
distpsnr{1, 2, 3}	Distancia de la {primera, segunda, tercera} fuente más cercana del catalógo de PanSTARRS1, si existe una dentro de 30 arcseg. Se asigna -999 cuando no hay fuente.
isdiffpos	*flag* de valor binario. 1 si el candidato es de resta positiva (ciencia menos referencia); 0 si el candidato es de resta negativa (referencia menos ciencia).
fwhm	*Full Width Half Max* asumiendo un núcleo gaussiano del candidato en la imagen de ciencia mediante SExtractor (Bertin, E. & Arnouts, S., 1996).
magpsf	Magnitud de la fotometría de ajuste PSF del candidato en la imagen de diferencia.
sigmapsf	Incertidumbre 1-sigma en *magpsf*.
ra, dec	Coordenadas ecuatoriales medidas en grados.
diffmaglim	límite de magnitud en la imagen de diferencia basado en fotometría de ajuste PSF.
classtar	*Score* de clasificación de estrella/galaxia del candidato en la imagen de ciencia mediante SExtractor.
ndethist	Número de detecciones coincidentes espacialmente que caen dentro de 1.5 segundos de arco y se remontan al comienzo del sondeo; sólo se cuentan las detecciones que cayeron en el mismo campo y el ID del canal de lectura donde se observó el candidato de entrada. Se incluyen todas las detecciones sin procesar hasta un $signal/noise$ fotométrico ≈ 3.
ncovhist	Número de veces que la posición del candidato de entrada cayó en cualquier campo y canal de lectura que se remonta al inicio del telescopio.
chinr, sharpnr	Parámetros detectados por DAOPhot (Stetson, 1987). Corresponden al *chi*, *sharp* de la fuente más cercana en la imagen de referencia PSF-catalog dentro de 30 segundos de arco. Son métricas asociadas que aseguran que sea una fuente puntual.

Fuente: Carrasco-Davis et al. (2020)

3.1.3 Estampillas

Como se ha mencionado en la Sección 2.2.3 cada alerta contiene tres estampillas que contienen la información de la observación. Cada estampilla es de 63×63 píxeles. Cada estampilla tiene centrado el objeto/evento avistado.

Hay ocasiones en que las estampillas vienen con dimensiones erróneas, es decir, tienen una dimensión menor a la esperada. Aún así, ZTF entrega las coordenadas *x* e *y* de donde se encuentra el candidato (en la alerta se define como *xpos* e *ypos* respectivamente), por lo tanto es posible reestructurar las estampillas; centrando el candidato en las coordenadas *x* e *y* y luego rellenar con algún valor definido hasta tener una imagen de 63×63 píxeles.

3.1.4 Etiquetas

Förster et al. (2020) propone un conjunto de etiquetas basado en el *crossmatch* de fuentes de ZTF con diferentes catálogos disponibles en la literatura o compilados por ALeRCE. La construcción de este conjunto de etiquetas está diseñado a partir del conocimiento experto y logra conformar una taxonomía de clasificación (véase Anexo A).

Sánchez-Sáez et al. (2021) para el modelo basado en un *Hierarchical Random Forest* (HRF) construye dos niveles de clasificación a partir del conjunto de etiquetas de ALeRCE, por un lado puede clasificar entre si un objeto es transiente, estocástico o periódico y entre otras 15 clases correspondiente a una categoría más específica.

Carrasco-Davis et al. (2020) para entrenar el modelo *stamp classifier* realiza un mapeo de las etiquetas del conjunto de etiquetas de Sánchez-Sáez et al. (2021), quedando solo con supernovas (SN), galaxias con un núcleo activo (AGN) y estrellas variables (VS). En la Tabla 3.3 se muestra el mapeo realizado entre las clases de los modelos mencionados anteriormente.

Actualmente el conjunto etiquetado está en su versión v8.0.3 y posee más de 250.000 registros, este subconjunto solo representa el 0.004% del total de objetos avistados por ZTF, esta cifra sirve para comprender la cantidad de datos que es capaz de recolectar este *survey*.

Los asteroides son objetos que se mueven dentro del sistema solar, por esta misma razón solo deberían aparecer una vez en una posición determinada, por lo tanto mostrarían flujo en las imágenes de *science* y *difference*. A partir de la instrumentación de ZTF el valor de *ssdistnr*[4] debería variar entre 0 y 10 (Carrasco-Davis et al., 2020). A pesar de que no deberían existir asteroides con más de una detección, en la base de datos de ALeRCE existen más de 590.000

[4]Distancia al objeto del sistema solar conocido más cercano. Si no posee un valor, se define como -999. Se mide en arco segundos.

Tabla 3.3: Etiquetas de ALeRCE

HRF primer nivel	HRF segundo nivel	*stamp classifier*
Transient	SNIa	SN
	SNIbc	
	SNII	
	SLSN	
Stochastic	AGN	AGN
	QSO	
	Blazar	
	CV/Nova	-
	YSO	VS
Periodic	LPV	
	RRL	
	CEP	
	E	
	DSCT	
	Periodic-Other	

Fuente: Elaboración propia con datos de ALeRCE (Förster et al., 2020)

objetos que cumplen la condición del *ssdistnr* para ser un asteroide y además tienen más de dos alertas. Para agregar una muestra de 60.000 asteroides al conjunto de etiquetas se ha realizado la siguiente consulta a la base de datos de ALeRCE:

```
WITH asteroids AS (
    SELECT sz.oid, sz.ssdistnr, o.ndet FROM ss_ztf sz
    INNER JOIN "object" o ON sz.oid = o.oid
    WHERE ssdistnr >= 0 and ssdistnr < 10 and ndet >=2
)
SELECT * FROM asteroids ORDER BY random() LIMIT 60000;
```

Dicha consulta selecciona de manera aleatoria 60.000 objetos de la intersección de la tabla *ss_ztf* (tabla que contiene objetos del sistema solar) y tabla *object* (tabla con información agregada de los objetos), que cumplan con la condición de tener más de 2 detecciones y una distancia entre 0 y 10 arcosegundos al objeto conocido más cercano del sistema solar.

Los *bogus* al igual que los asteroides suelen tener una detección, pero existen casos en que se repiten los *bogus* en una misma región del cielo, esto se da por zonas muy brillantes que saturan los píxeles de la cámara o simplemente son artefactos repetitivos en el tiempo. La clase *bogus* se agrega a partir de:

- Etiquetas *bogus* a partir del trabajo realizado por Carrasco-Davis et al. (2020).
- Conocimiento experto plasmado en la base de datos de reportes de ALeRCE. ALeRCE cuenta con un sistema web que permite a los astrónomos reportar si un objeto es un

candidato a supernova o es un *bogus*. Se han seleccionado los *bogus* que han sido reportados por al menos dos astrónomos de ALeRCE.

Finalmente el conjunto de etiquetas suma un total de 350.951 registros, en la Sección 3.3 se detalla más acerca de la distribución de las etiquetas.

3.2 OBTENCIÓN DE LOS DATOS

Luego de unir el conjunto de etiquetas de ALeRCE, asteroides y *bogus*, se tiene un archivo CSV con las columnas *objectId*, *classALeRCE* y *subclassALeRCE*, tal como se ve en la Tabla 3.4. Este archivo se utiliza como entrada para la obtención de estampillas y metadatos para el desarrollo de esta tesis.

Tabla 3.4: Muestra de archivo CSV de etiquetas

objectId	classALeRCE	subclassALeRCE
ZTF17aaaaasi	VS	LPV
ZTF17aaaaast	VS	LPV
ZTF17aaaaasx	VS	LPV
ZTF17aaaaatf	VS	LPV
ZTF17aaaaavq	VS	LPV
ZTF17aaaabfs	VS	LPV
ZTF17aaaabpo	VS	EB/EW
ZTF17aaaabrg	VS	EB/EW
ZTF17aaaabrl	VS	EB/EW
ZTF17aaaabte	VS	EB/EW

Fuente: Elaboración propia, 2021

Para la recuperación de las alertas se utiliza Elastic Map Reduce (EMR), servicio de AWS que permite ejecución distribuida de código a través de Apache Spark. Claramente el uso de este servicio tiene un costo monetario asociado y para este trabajo se dispone del respaldo de ALeRCE para la obtención de las alertas.

Otro de los motivos de porque utilizar el servicio de EMR corresponde a que las alertas están en Simple Storage Service (S3), servicio de AWS que permite almacenar de manera elástica archivos (cada archivo se denomina objeto) en la nube. ALeRCE tiene creado un *bucket* de objetos donde están las alertas y EMR puede leerlo sin un costo asociado, es decir, se pueden leer y procesar los 7.5 TB de alertas de manera distribuida con Apache Spark.

El flujo de trabajo es el siguiente:

1. A partir del archivo CSV de entrada, se buscan todas las alertas de todos los *objectId* que están en el *bucket* de alertas.

2. Del conjunto de alertas se recuperan las dos primeras alertas por objeto (si un objeto tiene solo una alerta, ésta se recupera también).
3. Se escriben los datos recuperados en un *bucket* de salida.

Para el conjunto de datos recuperados, se tiene la siguiente información:

- Tamaño del conjunto de datos recuperado: 24.1 GB
- Total de objetos recuperados: 350.951
- Total de objetos con al menos dos alertas: 289.077

Dentro de los costos asociados a la búsqueda y recuperación de los datos, se puede calcular el costo monetario, el cual asciende a un total de 4.93 USD. Para esto se debe considerar lo siguiente:

- Costo por una unidad de instancia m5.xlarge *Master* de EMR: $0.192 USD por hora de la instancia.
- Costo por 30 unidades de instancia m5.2xlarge *Core* de EMR (de tipo *Spot*[5]): $0.185 USD por hora de cada instancia.
- Tiempo de ejecución total: 36 minutos[6].

Como se ha mencionado al inicio de este capítulo, actualmente se tienen más de 61 millones de objetos en formato AVRO y para la conformación del conjunto de datos solo se han recuperado las primeras dos alertas de 289.077 objetos, es decir, solo se utiliza el 0.004% de los objetos para el desarrollo de esta tesis.

3.3 ANÁLISIS DE LOS DATOS

Un análisis previo de los datos permite comprender el dominio del problema y cómo atacarlo, evidenciando qué características podrían ser fundamentales. En primer lugar, se puede apreciar la distribución de las clases del conjunto de datos. Tal como se ve en la Figura 3.1, se logra apreciar un claro desbalance en el conjunto de datos.

En la Figura 3.2 se puede evidenciar la distribución de sub-etiquetas de los datos que se tiene. Es posible observar cierto dominio de sub-etiquetas en las clases a trabajar, dicho

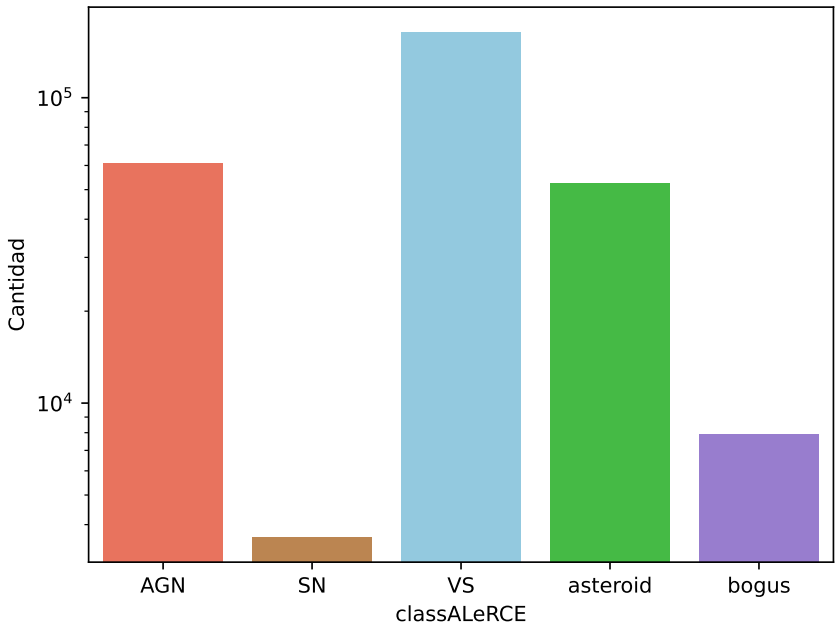


Figura 3.1: Distribución de clases en el conjunto de datos. Se presentan la cantidad de objetos de las 5 clases que involucra esta investigación: agujeros negros supermasivos (AGN), supernovas (SN), estrellas variables (VS), asteroides y *bogus*. Fuente: Elaboración propia, 2021

dominio podría provocar un posible sobreajuste a las sub-clases mayoritarias. En la Sección 3.5 se describe un método de estratificación para el beneficio de las sub-etiquetas minoritarias.

Para realizar el análisis de los metadatos, se toma una muestra de 2000 objetos por cada clase, esto permite tener una noción de las distribuciones de los datos. En el Anexo C se presenta un breve análisis de los metadatos que no se exponen a lo largo de esta sección.

Una de las características que podría resultar un buen discriminador entre clases es el *deltajd*, una característica calculada que indica la diferencia de días en que fue obtenida la segunda alerta respecto a la primera. En la Figura 3.3 se aprecia como la clase asteroide y supernova tienden a tener un *deltajd* menor a un día, es decir ocurren la misma noche (10^0 en el eje de las abscisas). Por otro lado, la clase AGN y VS suelen tener una distribución de *deltajd* parecida luego de un día. El *deltajd* puede ser una característica sensible a agentes externos, como por ejemplo malas condiciones climáticas o rutinas de operación en la cámara de ZTF, que podrían dejar inoperativo por un tiempo al *survey*.

Por otro lado, existen otras características que podrían ser un buen discriminador que no han sido utilizados por Carrasco-Davis et al. (2020), como por ejemplo *elong*, que indica la elongación de la imagen, prácticamente es la relación de píxeles entre las longitudes de los ejes semi-mayor y semi-menor respectivamente, asumiendo que el objeto detectado está descrito de

[5]Es un tipo de instancia de AWS más barata que una instancia normal. Ocupan tiempo de cómputo sobrante de otras instancias de la región.

[6]Considerar que no se ha tomado en cuenta el tiempo de configuración de la máquina.

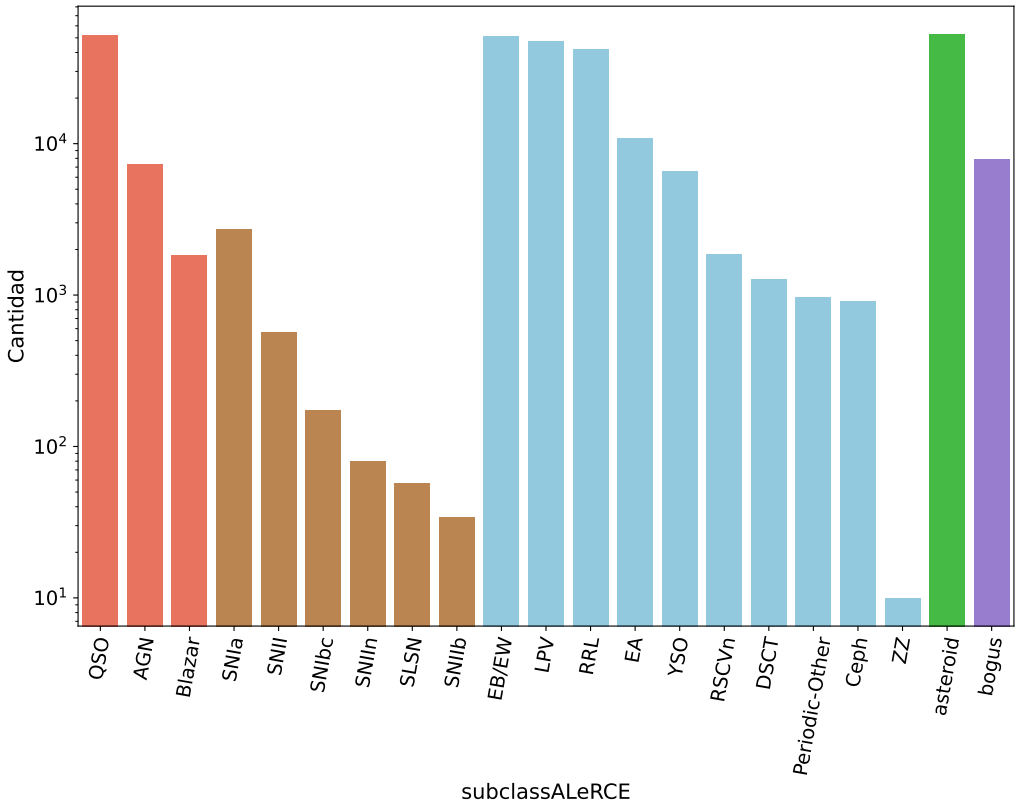


Figura 3.2: Distribución de sub-clases en el conjunto de datos. Las clases macro se encuentran del mismo color y las sub-clases están en orden descendente respecto a la cantidad de elementos. Fuente: Elaboración propia, 2021

forma elíptica (Bertin, E. & Arnouts, S., 1996). En la Figura 3.4 se aprecia la distribución de *elong* para la primera y segunda detección, en donde se aprecia una leve diferencia entre la distribución de *bogus* y el resto de clases.

Hay otros metadatos que permitirían discriminar bien entre clases, pero introducirían un sesgo a los modelos que se entrenen con estos datos, como por ejemplo: (a) la magnitud de apertura e incertidumbre de dicha magnitud, representada por los metadatos *magap* y *sigmagap* ; (b) magnitud e incertidumbre a la fuente más cercana del catálogo PS1; (c) el filtro *real-bogus* (*rb*) que corresponde a un *score* que otorga ZTF para indicar que el objeto en cuestión es un *bogus*, este valor varía entre 0 y 1, mientras más cercano sea a 1, indicaría que el objeto es real y por el contrario, si es más cercano a 0 sería *bogus*. En la Figura 3.5 se ve como actúa dicho filtro y se puede ratificar que existe una tendencia de las etiquetas *bogus* a un *rb* inferior a 0.5.

En la Figura 3.6 se observa los filtros utilizados para las dos primeras alertas, en el eje x se aprecia el par de filtros utilizados, por ejemplo *gr* indica que la primera alerta fue con el filtro *g* y la segunda con el filtro *r*. Uno de los aspectos importantes de esta figura, es que en las supernovas se tiende a utilizar ambos filtros en las dos primeras alertas, es decir, se encuentra en

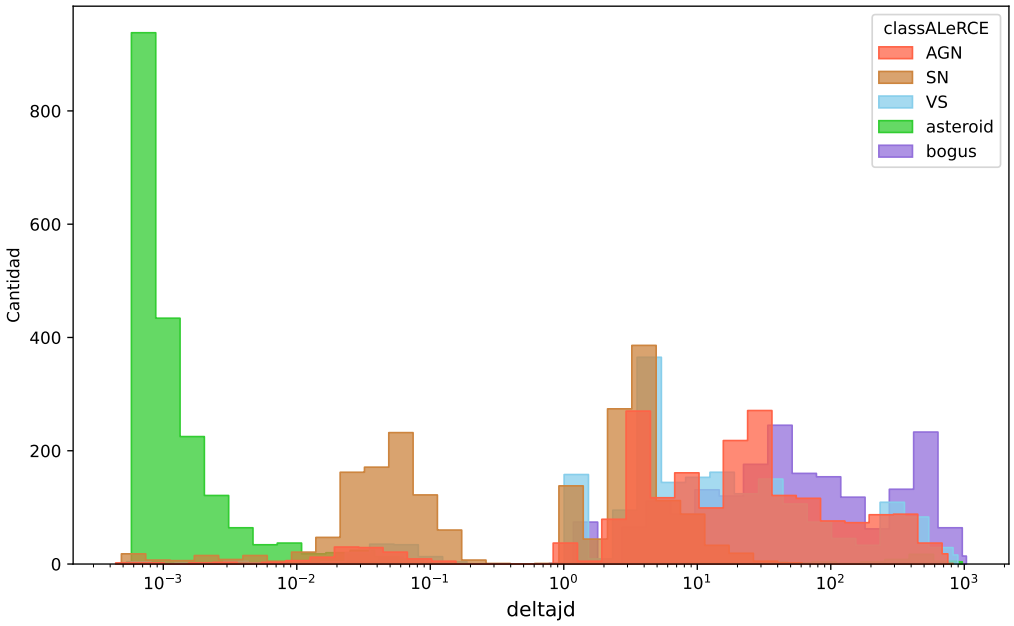


Figura 3.3: Histograma de *deltajd* por clase - Fuente: Elaboración propia, 2021

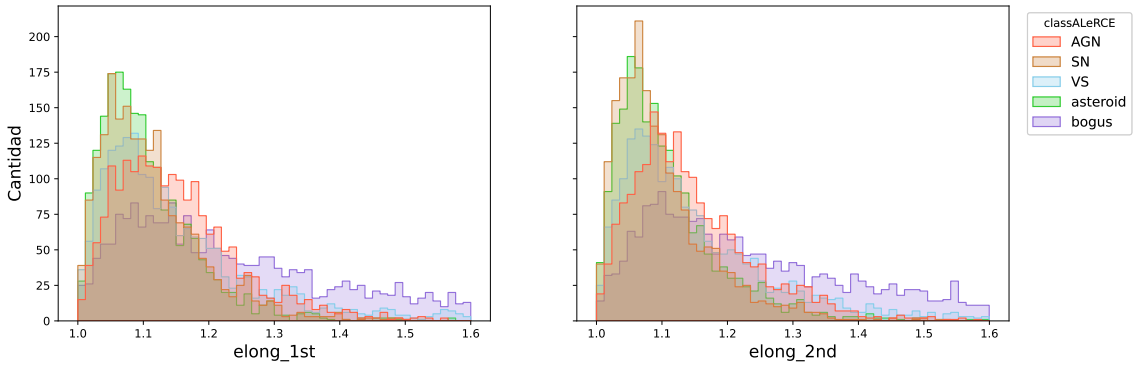


Figura 3.4: Histograma de *elong* por clase - Fuente: Elaboración propia, 2021

mayor proporción la relación *gr* y *rg* que *gg* y *rr*.

Debido a que existen metadatos que pueden introducir un sesgo a modelos de aprendizaje de máquina, el equipo de expertos de astrónomos de ALeRCE llegaron al consenso de utilizar sólo los metadatos que se muestran en la Tabla 3.2, además del *deltajd*, los filtros utilizados en cada alerta y el metadato *elong*.

Del total de 289.077 objetos, el 98.46% de la primera alerta y el 98.42% de la segunda alerta tienen estampillas cuadradas, es decir, las dimensiones corresponde a 63×63 píxeles. El 97.59% de los objetos tienen dimensiones correctas en la primera y segunda alerta. Las estampillas que no son de las dimensiones esperadas se corrigen pero llegan a ser menos del 3%, en la Sección 3.5 se detalla la reconstrucción de las estampillas con dimensiones incorrectas.

Existen casos en que ZTF calcula y compone una máscara en zonas muy densas

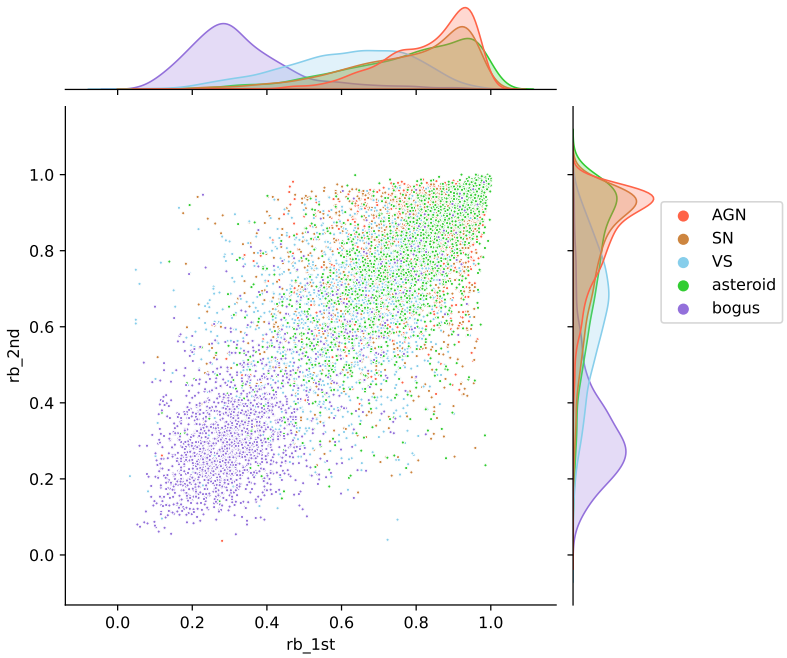


Figura 3.5: Puntuación de calidad de *real bogus* entregado por ZTF para cada alerta. El rango es de 0 a 1, donde más cerca de 1 indica ser una observación confiable - Fuente: Elaboración propia, 2021

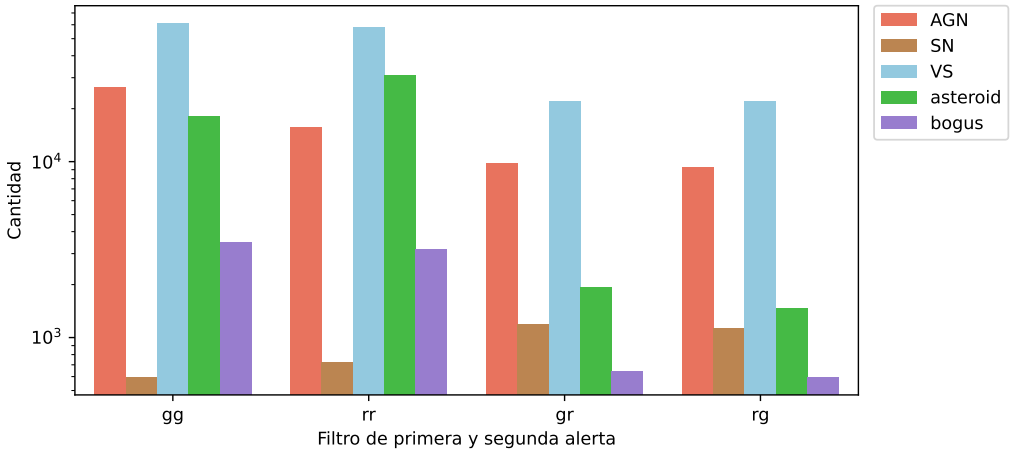


Figura 3.6: Cambio de filtros de las dos primeras alertas. Fuente: Elaboración propia, 2021

del cielo, quedando una forma blanca (generalmente rectángulos) que cubren una porción de las estampillas. En la Figura 3.7 se aprecia una muestra de las estampillas de 5 objetos por clase, cada clase tiene ciertas características representativas; por ejemplo en las supernovas generalmente se aprecia una galaxia anfitriona, los *bogus* suelen ser artefactos en zonas muy saturadas respecto al brillo en el cielo, etc.

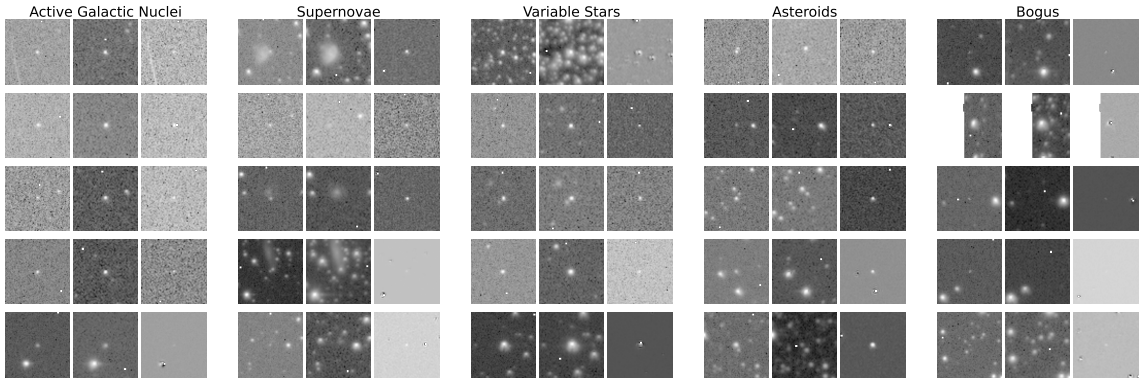


Figura 3.7: Muestra de estampillas por clase. Cada fila por clase muestran los recortes de *science*, *reference* y *difference* respectivamente - Fuente: Elaboración propia, 2021

3.4 CONJUNTO DE ENTRENAMIENTO, VALIDACIÓN Y PRUEBA

Dado que el conjunto de datos está desbalanceado, para la construcción del conjunto de datos a trabajar se tomó la decisión de limitar a un máximo de 30.000 objetos por clase y limitar a un máximo de 100 días la característica *deltajd*, ya que el caso de uso del modelo desarrollado en esta tesis contempla objetos que varíen y tengan alertas en épocas cortas.

Para el corte de las 30.000 muestras por clase, se realiza un proceso de limitación a aquellas sub-clases que sobrepasan este valor. Posteriormente, por fuerza bruta se busca la combinación de sub-etiquetas que sumen 30.000 en su etiqueta macro, pero penalizando a la sub-etiqueta mayoritaria. En Anexo B se describe el algoritmo utilizado.

En la Figura 3.8, se logra apreciar que varias sub-etiquetas sobrepasan el limite establecido. Una vez realizado el proceso repartición de las sub-etiquetas, el conjunto de datos queda como se muestra en la Figura 3.9, donde se demuestra que se favorece a las sub-etiquetas minoritarias.

En el trabajo de Carrasco-Davis et al. (2020) se limitó a un máximo de 15.000 objetos por clase, pero no se realizó un procedimiento de corte por sub-clase, lo cual puede indicar que este trabajo de tesis tenga un conjunto de datos más representativo en cuanto a las sub-etiquetas, ya que se pudo haber dado el caso que en los datos de Carrasco-Davis et al. (2020) una sub-clase predominara en la clase (e.g. toda la clase AGN fue representada por la sub-etiqueta QSO).

Posteriormente, se construye el conjunto de entrenamiento, validación y prueba. Para esto se toma una muestra de 200 objetos por clase para el conjunto de validación y una muestra de 100 objetos por clase para el conjunto de prueba. La selección de objetos para los conjuntos de validación y prueba, está estratificado según la sub-clase, quedando distribuidos como lo muestra la Tabla 3.5

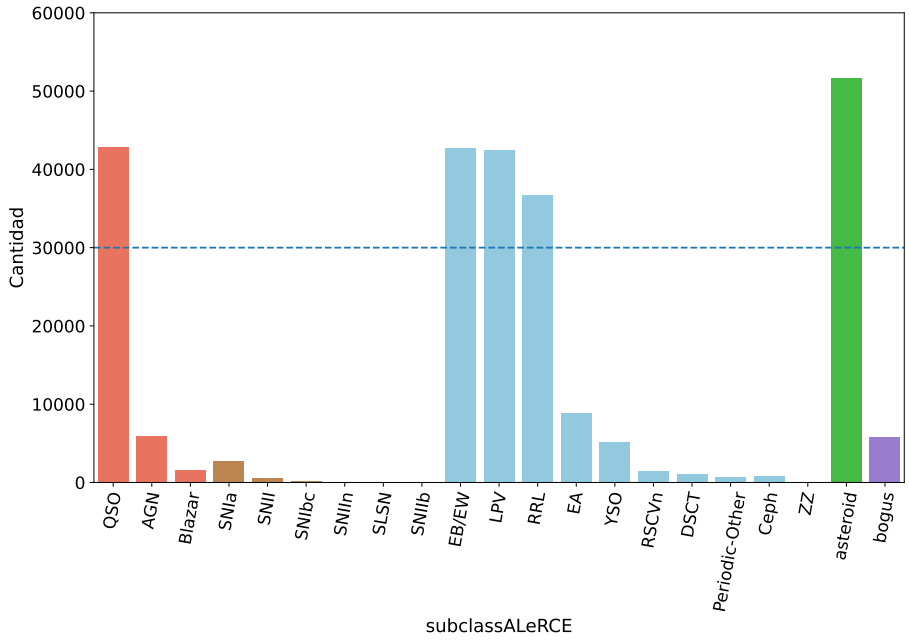


Figura 3.8: Sub-clases del conjunto de datos. Fuente: Elaboración propia, 2021

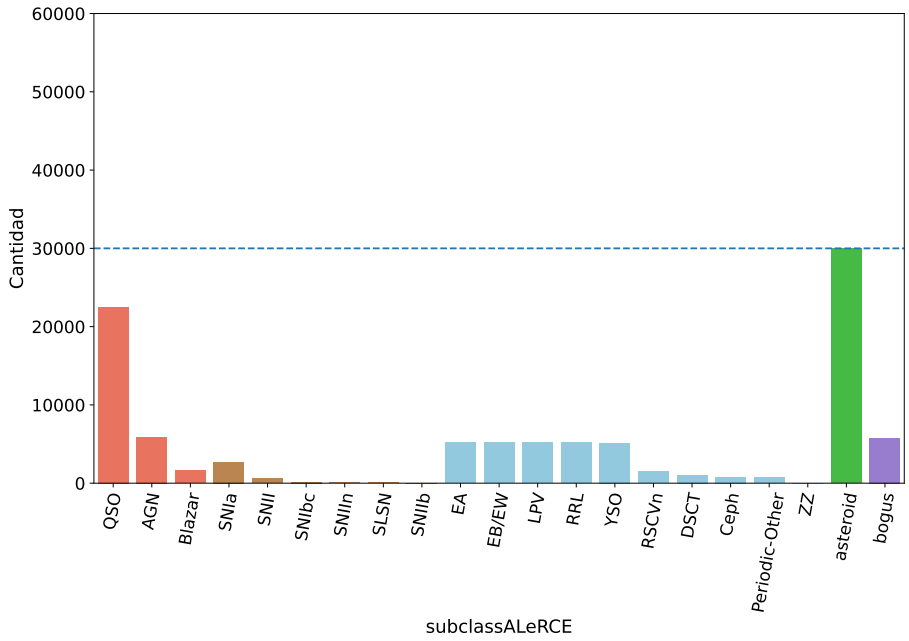


Figura 3.9: Sub-clases limitadas en 30.000. Notar que la suma de sub-clase por cada clase da el total de 30.000. Fuente: Elaboración propia, 2021

Tabla 3.5: Conjunto de entrenamiento, validación y prueba

Clase	Sub-clase	Cant. entrenamiento	Cant. validación	Cant. prueba
AGN	QSO	22288	150	75
	AGN	5844	39	20
	Blazar	1568	11	5
SN	SNIa	2504	150	75
	SNII	523	31	16
	SNIbc	159	10	5
	SNIIn	74	4	2
	SLSN	52	3	1
	SNIIb	31	2	1
VS	EB/EW	5153	34	17
	EA	5152	35	17
	LPV	5152	35	17
	RRL	5152	35	17
	YSO	5071	34	17
	RSCVn	1451	10	5
	DSCT	1050	7	4
	Ceph	773	5	3
	Periodic-Other	740	5	3
	ZZ	6	-	-
	Total	62743	600	300
	%	98.58%	0.94%	0.47%

Fuente: Elaboración propia, 2021

3.5 PREPROCESAMIENTO

3.5.1 Estampillas

El preprocesamiento de las estampillas realiza las siguientes acciones:

- Reparar estampillas que no son de 63×63 píxeles. En el caso de que las estampillas no sean de las dimensiones correctas, se centra el evento astronómico de acuerdo a los metadatos *xpos* e *ypos*, luego se rellena con NaN hasta tener una estampilla de 63×63.
- Se normaliza cada estampilla de 63×63 píxeles. Para esto se calcula un máximo y mínimo de acuerdo al q-ésimo cuantil de las estampillas ignorando los valores NaN. Para el caso del máximo se utiliza el cuantil 0.9 y 0.1 para el mínimo. Luego cada píxel de la imagen se resta con el mínimo y el resultado se divide por el máximo. Posteriormente se limitan los valores de los píxeles en el rango $[-1, 1]$ y los valores NaN se establecen como 2. Cabe destacar que los valores de los píxeles son del tipo punto flotante.

3.5.2 Metadatos

En primer lugar se utilizan los metadatos utilizados por Carrasco-Davis et al. (2020), los cuales se muestran en la Tabla 3.2 para las primeras dos alertas. En resumen se realizan las siguientes operaciones:

- Cálculo de coordenadas eclípticas.
- Cálculo de coordenadas galácticas.
- Cálculo de días transcurridos entre una alerta y otra (*deltajd*).
- Recuperación de filtros utilizados para cada alerta. Es posible que ambas alertas tengan el mismo filtro, aunque también puede ocurrir lo contrario. Para el uso de los filtros (que corresponde a un número entero que puede ser 1 ó 2 en el caso de ZTF) se utiliza la codificación *one hot*[7].

Los valores que son NaN, se establecen como 0.0. Luego se concatenan los vectores de los metadatos de la primera alerta con la segunda, quedando un solo vector de tamaño 49 por cada objeto en el conjunto de datos.

Finalmente, se estandarizan los valores de los conjuntos de entrenamiento, validación y prueba. Para esto se calcula la media $\bar{x_e}$ y desviación estándar σ_e por metadato del conjunto de entrenamiento, luego se aplica la Ecuación 3.1 a cada metadato de cada conjunto. Estos valores de $\bar{x_e}$ y σ_e por metadato deben ser almacenados, ya que éstos permiten estandarizar nuevos valores de acuerdo al conjunto con que se va a entrenar el modelo.

$$\frac{X - \bar{x_e}}{\sigma_e} \tag{3.1}$$

3.6 SERIALIZACIÓN EN TFRECORDS

Los TFRecords[8] corresponde a una serialización para el almacenamiento secuencial en binario de datos. Esto permite una serie de ventajas para grandes conjuntos de datos, ya que está diseñado para operaciones exigentes en disco.

Por otro lado, los TFRecords están optimizados para su uso con TensorFlow. Se tiene una serie de operaciones que permiten lectura y preprocesamiento en paralelo. El uso

[7] La codificación en *one hot* permite representar categorías o números nominales en un vector de tamaño n. Por ejemplo para el caso de los filtros de ZTF se tiene un $n = 2$, donde el vector $[0, 1]$ indicaría que el filtro utilizado es el 2.

[8] `https://www.tensorflow.org/tutorials/load_data/tfrecord`

de TFRecords es de bastante utilidad para conjuntos de datos grandes, donde éstos últimos sobrepasen la memoria de la máquina a utilizar, ya que permite la lectura que se requiera en el momento (lectura por lotes, mientras se está procesando un lote de *records* también se está preparando la lectura del próximo lote, y así sucesivamente).

La serialización que se realiza se basa en la lógica llave/valor, donde cada *record* correspondería a un objeto del conjunto de datos, con sus respectivos valores. Los conjuntos de entrenamiento, validación y prueba se serializan por separado para una mayor organización. Cada registro en los TFRecord contiene:

- *oid*: Identificación del objeto.
- *deltajd*: Días transcurridos entre la primera y segunda alerta.
- *classALeRCE*: Clase a la que pertenece el objeto.
- *first_stamp*: Tripleta de *science*, *reference* y *difference* de la primera alerta.
- *second_stamp*: Tripleta de *science*, *reference* y *difference* de la segunda alerta.
- *first_metadata*: Metadatos utilizados de la primera alerta.
- *second_metadata*: Metadatos utilizados de la segunda alerta.

La organización de los registros de TFRecord queda con los siguientes directorios, donde cada directorio contiene un conjunto de TFRecords. Esto permite acceder de manera más rápida a los registros, ya que se encuentran organizados por nombre en el conjunto de entrenamiento. La idea de esta organización, es que cada directorio actúa como la fuente de datos de una *pipeline* de lectura.

```
dataset_folder/
  |
  |--- training/
  |     |
  |     |--- AGN/
  |     |--- asteroid/
  |     |--- bogus/
  |     |--- SN/
  |     |--- VS/
  |--- test/
  |--- validation/
```

CAPÍTULO 4. *TWO STAMP CLASSIFIER*

Para resolver el problema de la clasificación de objetos astronómicos mediante sus primeras dos alertas se propone el uso de una CNN. En este capítulo se describe las arquitecturas propuestas, como es el proceso de entrenamiento y los experimentos a realizar.

4.1 ARQUITECTURAS

La arquitectura de la CNN propuesta está basada en el modelo *stamp classifier* de Carrasco-Davis et al. (2020), ya que los resultados presentados por el autor han demostrado que el uso de las estampillas y metadatos de la primera alerta permiten clasificar objetos astronómicos de manera exitosa. En la Figura 2.21 se puede apreciar las capas que realizan la operación de convolución.

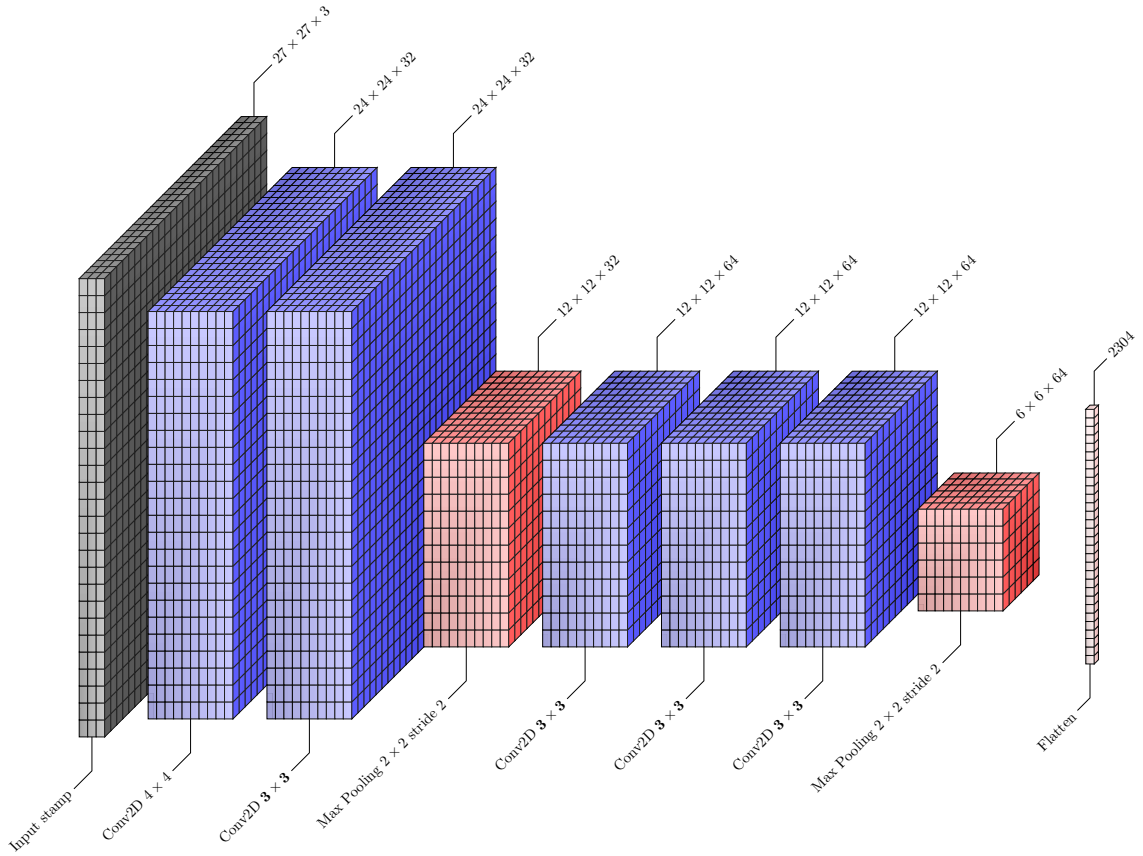


Figura 4.1: Capas convolucionales de *stamp classifier*. Notar que las convoluciones de 3×3 son un hiper-parámetro del modelo, donde dicho tamaño del *kernel* entrego mejores resultados. Fuente: Elaboración propia, 2021

En la Figura 4.1 se aprecian las capas convolucionales del modelo de Carrasco-Davis et al. (2020). La entrada corresponde a estampillas cortadas y centradas, dejando la entrada de 21×21 píxeles y aplicado un *zero padding*[1] de 3. Por lo tanto, la entrada queda de 3 canales, con matrices de 27×27 píxeles. Este bloque de capas convolucionales y de *max pooling* terminan en un vector con la información que las capas han obtenido. Por simplicidad, en este trabajo se nombra como capa *DeepHiTS* a los bloques de capas convolucionales y capas densas que se encargan de reunir la información recuperada por las convoluciones. En la Tabla 4.1 se aprecian los parámetros y salida de la capa *DeepHiTS*. En la Tabla 4.2 se puede observar como el modelo de Carrasco-Davis et al. (2020) usa la capa de *DeepHiTS* y realiza la predicción.

Tabla 4.1: Bloque de convoluciones que realiza DeepHiTS

Capa	Parámetros de la capa	Tamaño de salida
Input	-	$21 \times 21 \times 6$
Zero padding	-	$27 \times 27 \times 6$
Rotation augmentation	-	$27 \times 27 \times 6$
Convolution	4×4, 32	$24 \times 24 \times 6$
Convolution	3×3, 32	$24 \times 24 \times 32$
Max-pooling	2×2, *stride* 2	$12 \times 12 \times 32$
Convolution	3×3, 32	$12 \times 12 \times 64$
Convolution	3×3, 32	$12 \times 12 \times 64$
Convolution	3×3, 32	$12 \times 12 \times 64$
Max-pooling	2×2, *stride* 2	$6 \times 6 \times 64$
Flatten	-	2304
Fully connected	2304×64	64
Rotation concatenation	-	4×64
Cyclic pooling	-	64

Fuente: Basado en Carrasco-Davis et al. (2020).

Tabla 4.2: Arquitectura de red neuronal de *stamp classifier*

Capa	Parámetros de la capa	Tamaño de salida
Stamp input	-	$21 \times 21 \times 3$
DeepHits layer	*kernel*: 3×3	64
Concat with batch normalizated features	-	$64 + 23$
Fully connected with dropout	87×64	64
Fully connected	64×64	64
Output softmax	64×5	5

Fuente: Basado en Carrasco-Davis et al. (2020).

[1] Rellenenar una matriz (o imagen) con una cierta cantidad de ceros en los margenes.

En el caso de la variación de *two stamp classifier*, se propone dos variantes del modelo de Carrasco-Davis et al. (2020) que puede usar las dos primeras alertas; en primer lugar una versión llamada *stacked two stamp classifier* que apila la entrada y sigue la lógica del *stamp classifier* y por otro lado, se propone la variación *separated two stamp classifier* en donde se usan dos *stamp classifiers* por separado, donde cada uno opera una alerta y posteriormente hace la predicción.

4.1.1 *Stacked two stamp classifier*

La arquitectura *stacked two stamp classifier* es la misma arquitectura de Carrasco-Davis et al. (2020), con la salvedad de que la entrada a las capas convolucionales están apiladas, se juntan las estampillas de $3 \times 63 \times 63$ de la primera y segunda alerta, quedando un tensor [2] de $6 \times 63 \times 63$, donde las primeras tres matrices son los recortes *cutoutScience*, *cutoutTemplate* y *cutoutDifference* respectivamente de la primera alerta, y las últimas tres matrices corresponden a los mismos recortes pero para la segunda alerta. Adicionalmente, se utilizan los metadatos concatenados de las primeras dos alertas con el logaritmo neperiano de *deltajd* [3].

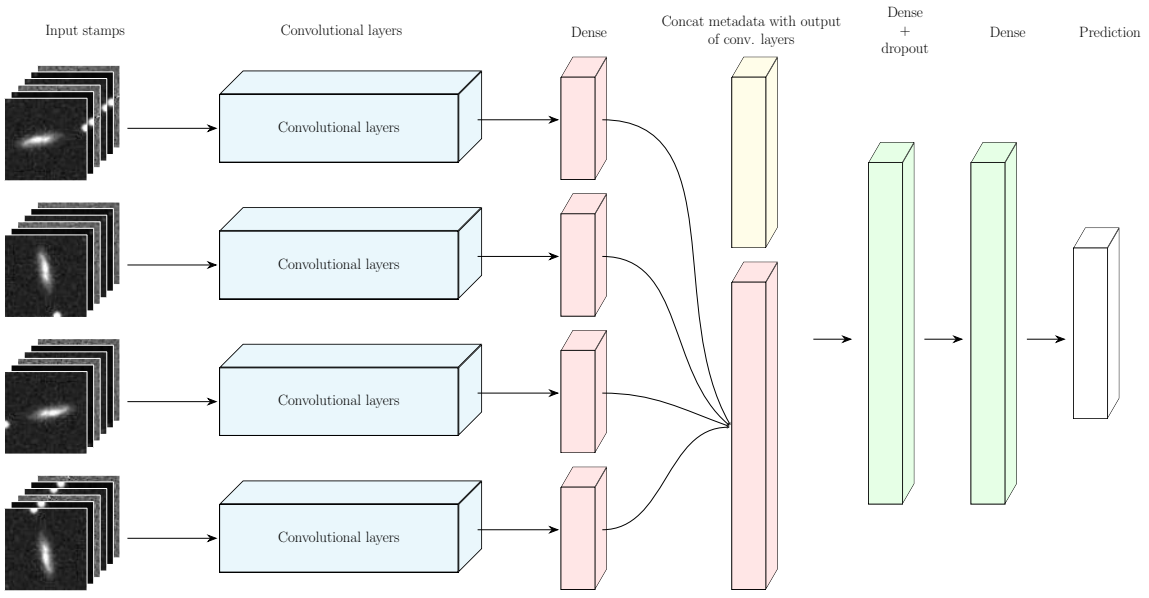


Figura 4.2: *Stacked two stamp classiffier*. Esta versión presenta una arquitectura similar a *stamp classifier*, donde la entrada de las imágenes corresponden a las estampillas de las dos primeras alertas apiladas y se agrega el metadato *deltajd* como entrada a las capas densas. Fuente: Elaboración propia, 2021

[2] Durante la experimentación se realizan cortes centrados en el objeto, es decir la entrada puede quedar de dimensiones más pequeñas en el ancho y largo de las imágenes (e.g $6 \times 21 \times 21$)

[3] Se aplica el logaritmo neperiano a *deltajd* por la escala de los valores de la característica. Gracias a esto se obtiene una representación en escala logarítmica que logra distribuir de mejor forma el *deltajd*.

En la Figura 4.2, el bloque amarillo representa los metadatos que han pasado por una capa de *batch normalization*. Posteriormente se concatena con la salida de las capas convolucionales y el resultado pasa por dos capas densas (bloques verdes de la figura). Finalmente, el tensor resultante pasa por una capa densa (bloque blanco) que tiene una salida de tamaño 5 (por las 5 clases), en donde cada valor del vector de tamaño 5 indica la posibilidad de ser alguna de esas clases.

En la Tabla 4.3 se aprecian los parámetros y salidas de las capas utilizadas en *stacked two stamp classifier*, donde la mayor diferencia está presente en las dimensiones de la entrada, tanto estampillas como metadatos.

Tabla 4.3: Arquitectura de red neuronal *stacked two stamp classifier*

Capa	Parámetros de la capa	Tamaño de salida
Stamp input	-	$21 \times 21 \times 6$
DeepHits layer	kernel: 3×3	64
Concat with batch normalizated features + deltajd	-	$64 + 53$
Fully connected with dropout	117×64	64
Fully connected	64×64	64
Output softmax	64×5	5

Fuente: Elaboración propia, 2021

4.1.2 *Separated two stamp classifier*

La variación *separated two stamp classifier*, corresponde a tener dos modelos de *stamp classifier* para cada tripleta de estampillas y metadatos, con la salvedad que cada modelo devuelve los resultados de la última capa densa, es decir, no devuelven un predicción sino que un conjunto de neuronas con sus pesos asociados. En la Figura 4.3 se muestra la composición y disposición del modelo, en donde se concatenan las salidas de cada *stamp classifier* y se agrega el *deltajd*. Posteriormente pasa por una capa densa y luego realiza la predicción.

En la Tabla 4.4 se aprecian los bloques de capas utilizados en la variante *separated two stamp classifier*, prácticamente se utilizan dos *stamp classifier* por separado, donde luego se concatenan los resultados de sus últimas capas densas con el metadato *deltajd*. Finalmente se obtiene la predicción mediante la MLP que se encuentra al final de la red.

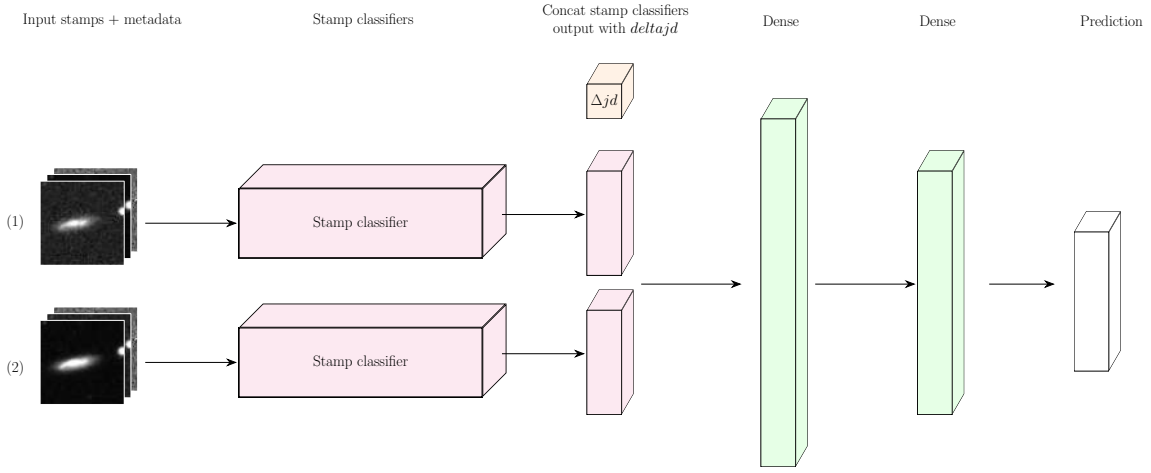


Figura 4.3: *Separated two stamp classiffier*. Esta variante presenta dos arquitecturas de *stamp classifier* que operan por separado las estampillas de la primera y segunda alerta y sus respectivos metadatos. Posteriormente se concatenan los resultados, incluyendo el metadato *deltajd*. Fuente: Elaboración propia, 2021

Tabla 4.4: Arquitectura de red neuronal *separated two stamp classifier*. El número entre paréntesis indica si la entrada corresponde a la (1) primera o (2) segunda alerta.

	Capa	Parámetros de la capa	Tamaño de salida
(1)	*Stamp input*	-	$21 \times 21 \times 3$
(1)	*DeepHiTS layer*	kernel: 3×3	64
(1)	*Concat with batch normalizated features*	-	$64 + 26$
(1)	*Fully connected with dropout*	90	64
(1)	*Fully connected*	64×64	64
(2)	*Stamp input*	-	$21 \times 21 \times 3$
(2)	*DeepHiTS layer*	kernel: 3×3	64
(2)	*Concat with batch normalizated features*	-	$64 + 26$
(2)	*Fully connected with dropout*	90	64
(2)	*Fully connected*	64×64	64
	*Concat **(1)** and **(2)** outputs with deltajd*	-	$64 + 64 + 1$
	Fully connected	128×128	128
	Fully connected	128×64	64
	Output softmax	64×5	5

Fuente: Elaboración propia, 2021

CAPÍTULO 5. METODOLOGÍA Y EXPERIMENTOS

Los experimentos a realizar están diseñados con la finalidad de comparar los resultados de los modelos propuestos y el *stamp classifier* de Carrasco-Davis et al. (2020), por lo cual también se ha implementado dicho modelo[1]. En el entrenamiento se configuran 50 épocas y en cada época se toma una cantidad determinada de *batches*. Estos *batches* están balanceados, es decir se toma la misma cantidad por clase. Puede ocurrir que algunas observaciones de las clases desbalanceadas (SN y *bogus*) se repitan en los *batches* tomados.

Cada modelo ha sido entrenado con el mismo conjunto de entrenamiento, además de evaluarlo y tomar decisiones con los mismos conjuntos de validación y prueba. Por otro lado, la búsqueda de hiper-parámetros se hizo a través de un muestreo aleatorio de 70 combinaciones posibles, de los parámetros que se encuentran a continuación:

- *kernel size*: $[3, 5]$. Hace referencia al *kernel* de las capas convolucionales. En la Figura 4.1 el *kernel size* definido es de 3×3.
- *crop size*: $[21, 42, 63]$. Hace referencia al corte que se puede realizar a las estampillas. Para esto la estampilla se centra y se cortan los bordes.
- *batch size*: $[64, 128, 256]$. Corresponde al tamaño de los *batches*. Este tamaño incide en las capas de *batch normalization*.
- *learning rate*: $[1^{-4}, 1^{-3}, 5^{-3}]$: Es la tasa de aprendizaje, indica que tan grande son los pasos en el período de entrenamiento.
- *dropout rate*: $[0.3, 0.5]$: Tasa de neuronas que se apagan en las capas de *dropout*.

Cada configuración del modelo se entrena 5 veces para asegurar resultados con una incertidumbre asociada. Por lo tanto, hay 70 combinaciones posibles que se repiten 5 veces por arquitectura.

La función de pérdida que se utiliza para encontrar los parámetros del modelo, corresponde a la función de entropía cruzada categórica. En TensorFlow la función se llama *CategoricalCrossEntropy* y requiere que las clases estén codificadas en *one hot*. Para encontrar los mínimos de la función de pérdida, se utiliza el optimizador de Adam. Estas configuraciones están basadas en el trabajo de Carrasco-Davis et al. (2020) y lo único que se varía en este trabajo al respecto, es el *learning rate* del optimizador.

El *learning rate* utilizado tiene el carácter de no ser un valor fijo, es decir, puede cambiar a medida que el optimizador se utilice en cada época, el que se utiliza en este tesis se

[1]El modelo de *stamp classifier* desarrollado en esta investigación está escrito TensorFlow 2.4 y la versión publicada está en TensorFlow 1.4. Esta actualización es importante dado que las nuevas versiones de TensorFlow permiten una puesta en producción más sencilla.

denomina *linear warmup*. Este permite mayor flexibilidad a medida que el modelo se entrena, ya que se parte con un *learning rate* un poco más pequeño que el establecido y a medida que se avanza el entrenamiento se aumenta linealmente el ritmo del aprendizaje hasta llegar a un ritmo constante. Esto reduce la volatilidad en las primeras etapas del entrenamiento.

Para monitorizar el estado del entrenamiento, se utiliza una herramienta llamada Tensorboard que permite visualizar las curvas de *loss* y *accuracy* por cada época en tiempo real, además de poder monitorizar recursos utilizados en el entrenamiento.

Finalmente se comparan los resultados de los modelos utilizando la prueba estadística de Welch (Welch, 1947), puesto que permite comparar medias entre dos grupos independientes cuando no se supone que los grupos tienen varianzas iguales. De esta forma es posible comparar los resultados de los distintos modelos entrenados, otorgando evidencia estadística para concluir acerca de estos.

CAPÍTULO 6. RESULTADOS

Cada arquitectura fue entrenada a partir de lo definido en el diseño experimental. Para esto se ocuparon recursos del Departamento de Ingeniería en Informática, específicamente la máquina *beam* que posee 4 GPUs Nvidia Tesla P100 (solo se ocuparon 3 para esta tesis). Posteriormente, para realizar el análisis se transfirieron los datos a un computador local y se obtuvieron las métricas y comparaciones pertinentes. A continuación se exponen los resultados obtenidos.

6.1 *STAMP CLASSIFIER*

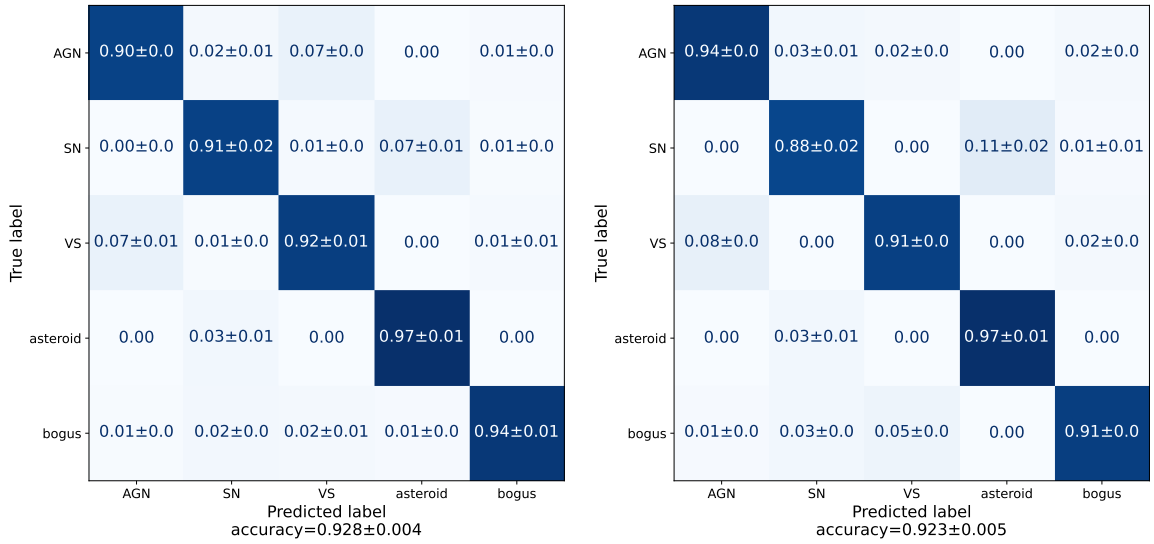
En la Tabla 6.1 se puede ver los mejores 10 modelos de *stamp classifier* ordenados de manera descendente según el *accuracy* en el conjunto de validación. Donde *dr* es el *dropout rate*, *ks* es el *kernel size*, *lr* el *learning rate*, *cs* el *crop size* y *bs* el *batch size*.

Tabla 6.1: Mejores 10 *stamp classifier* en *accuracy* de validación

dr	*ks*	*lr*	*cs*	*bs*	*accuracy* en validación	*accuracy* en prueba
0.3	3	0.0001	21	256	$\mathbf{0.928 \pm 0.004}$	0.923 ± 0.005
0.5	3	0.0001	21	256	0.927 ± 0.003	0.928 ± 0.003
0.3	3	0.001	21	256	0.927 ± 0.004	0.923 ± 0.004
0.3	5	0.0001	21	256	0.926 ± 0.006	0.922 ± 0.006
0.3	5	0.001	21	256	0.926 ± 0.005	0.923 ± 0.005
0.3	3	0.001	42	64	0.921 ± 0.003	0.923 ± 0.007
0.3	5	0.001	42	64	0.920 ± 0.003	0.922 ± 0.004
0.3	5	0.001	63	64	0.918 ± 0.006	0.916 ± 0.007
0.3	5	0.001	21	64	0.918 ± 0.004	0.915 ± 0.003
0.5	3	0.001	42	64	0.918 ± 0.003	0.912 ± 0.004

Fuente: Elaboración propia, 2021

En la Figura 6.1 se muestra la matriz de confusión para el modelo que tiene mayor *accuracy* en el conjunto de validación. El *accuracy* en el conjunto de prueba es de 0.923 ± 0.005, donde para la clase SN llega a acertar en una proporción de 0.88 ± 0.02 y suele clasificar un 0.11 ± 0.02 de asteroides como supernovas.



(a) Matriz de confusión para conjunto de validación

(b) Matriz de confusión para conjunto de prueba

Figura 6.1: Matriz de confusión para mejor *stamp classifier*. Fuente: Elaboración propia, 2021

6.2 *STACKED TWO TAMP CLASSIFIER*

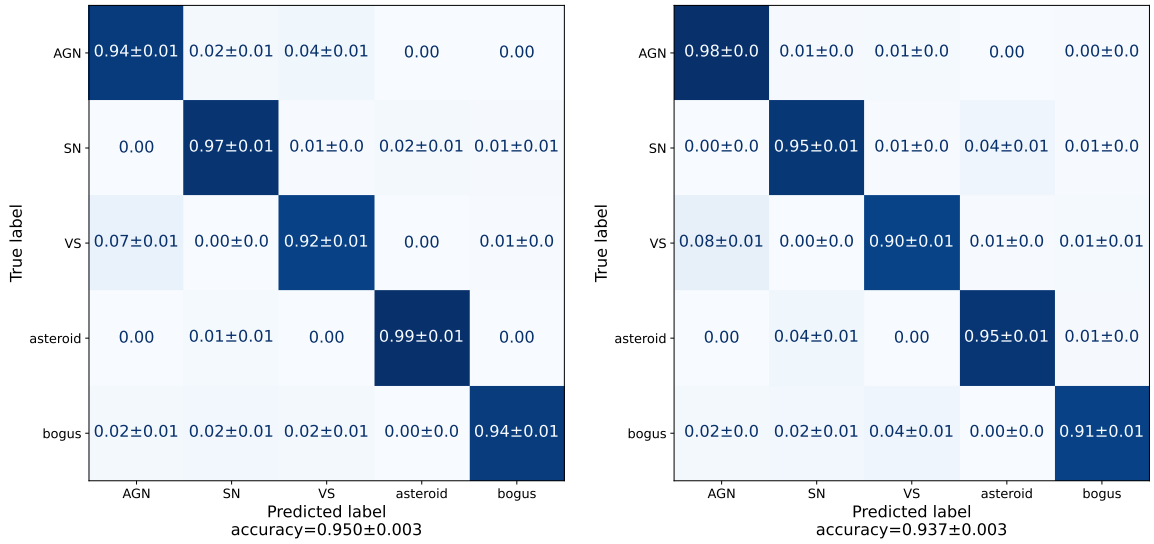
En la Tabla 6.2 se aprecia los mejores 10 *stacked two stamp classifier*, según el *accuracy* de validación. Tener en cuenta que esta arquitectura es la misma que *stamp classifier*, sólo se cambian las entradas y se incluye el *deltajd*.

Tabla 6.2: Mejores 10 *stacked two stamp classifier* en *accuracy* de validación

dr	*ks*	*lr*	*cs*	*bs*	*accuracy* en validación	*accuracy* en prueba
0.3	3	0.001	21	256	$\mathbf{0.950 \pm 0.003}$	0.937 ± 0.003
0.3	5	0.001	21	256	0.948 ± 0.005	0.935 ± 0.002
0.3	3	0.001	21	64	0.946 ± 0.003	0.935 ± 0.005
0.3	5	0.001	21	64	0.946 ± 0.002	0.936 ± 0.004
0.3	3	0.0001	21	256	0.946 ± 0.003	0.935 ± 0.006
0.5	3	0.001	21	64	0.943 ± 0.002	0.930 ± 0.005
0.3	5	0.001	42	64	0.943 ± 0.003	0.931 ± 0.005
0.3	3	0.001	42	64	0.941 ± 0.002	0.939 ± 0.002
0.3	5	0.001	63	64	0.941 ± 0.006	0.933 ± 0.006
0.5	5	0.001	21	64	0.941 ± 0.004	0.926 ± 0.005

Fuente: Elaboración propia, 2021

La Figura 6.2 muestra la matriz de confusión para el modelo con mayor *accuracy* en validación. De ésta se puede destacar que en la matriz de confusión para el conjunto de prueba se mejora el puntaje de clasificación de SN, los falsos positivos para esta clase son en mayor medida los asteroides.



(a) Matriz de confusión para conjunto de validación

(b) Matriz de confusión para conjunto de prueba

Figura 6.2: Matriz de confusión para mejor *stacked two stamp classifier*. Fuente: Elaboración propia, 2021

6.3 *SEPARATED TWO STAMP CLASSIFIER*

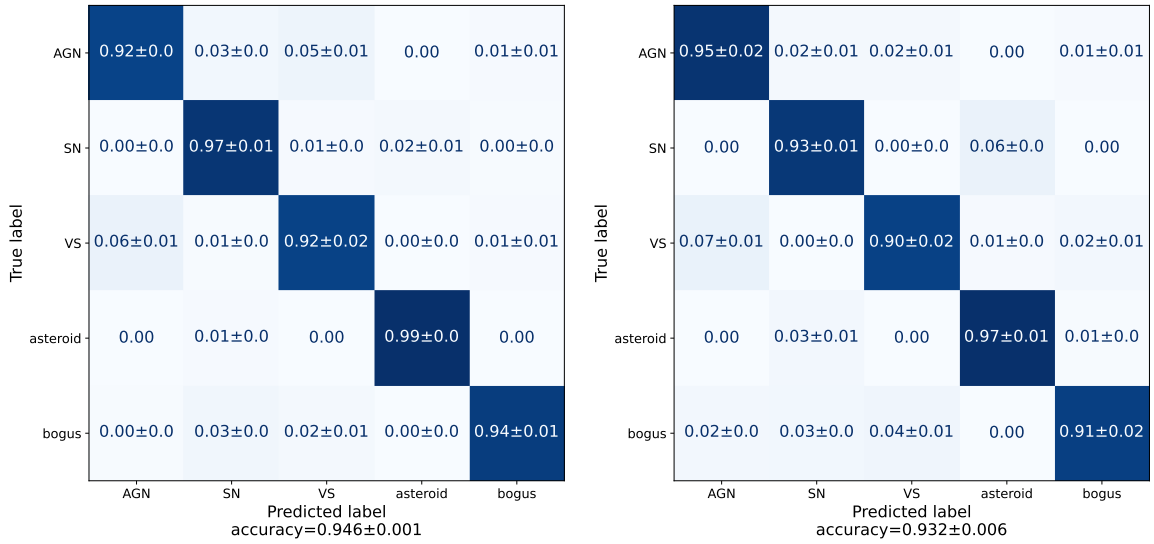
Al igual que para las arquitecturas anteriores, se presentan los 10 mejores modelos para *separated two stamp classifier* en la Tabla 6.3.

Tabla 6.3: Mejores 10 *separated two stamp classifier* en *accuracy* de validación

dr	*ks*	*lr*	*cs*	*bs*	accuracy en validación	accuracy en prueba
0.3	5	0.001	21	64	$\mathbf{0.946 \pm 0.001}$	0.932 ± 0.006
0.3	3	0.001	21	64	0.946 ± 0.002	0.938 ± 0.003
0.5	5	0.001	21	64	0.943 ± 0.001	0.932 ± 0.003
0.3	3	0.001	42	64	0.941 ± 0.004	0.929 ± 0.006
0.3	5	0.001	42	64	0.939 ± 0.005	0.935 ± 0.004
0.3	5	0.0001	21	64	0.938 ± 0.003	0.937 ± 0.007
0.3	3	0.005	63	64	0.938 ± 0.001	0.930 ± 0.001
0.3	3	0.005	42	64	0.938 ± 0.002	0.932 ± 0.003
0.5	3	0.001	21	64	0.937 ± 0.004	0.938 ± 0.005
0.3	5	0.005	21	64	0.937 ± 0.003	0.935 ± 0.005

Fuente: Elaboración propia, 2021

Analizando la matriz de confusión para el conjunto de prueba de la Figura 6.3, se ve un comportamiento similar al mejor modelo de *stacked two stamp classifier*. A grandes rasgos, la predicción de la clase asteroide también suele confundirse con la clase SN.



(a) Matriz de confusión para conjunto de validación

(b) Matriz de confusión para conjunto de prueba

Figura 6.3: Matriz de confusión para mejor *separated two stamp classifier*. Fuente: Elaboración propia, 2021

6.4 COMPARACIÓN ENTRE MODELOS

Un análisis general para los resultados expuestos es que existe una tendencia a que los modelos confunde la clase AGN con VS, véase Figura 6.2 y Figura 6.3. Coincide con el conocimiento de los astrónomos expertos, que indican que es una tarea compleja identificar tales clases con sólo dos observaciones, ya que se necesita observar la evolución en el tiempo del objeto y caracterizar mediante más observaciones (e.g. utilizando la curva de luz).

La selección del mejor modelo por arquitectura mediante la búsqueda de hiper-parámetros es comparando el rendimiento en el conjunto de validación. Para comparar los resultados de cada arquitectura, se ha realizado una prueba de hipótesis de Welch (Welch, 1947) entre los mejores modelos de cada una. En la prueba de hipótesis se compara el rendimiento en el conjunto de validación de:

- *stamp classifier* vs *stacked two stamp classifier*: Según la hipótesis nula de que las medias de ambos *accuracies* es igual y la hipótesis alternativa de que la media del *accuracy* del modelo *stamp classifier* es menor que la del *stacked two stamp classifier*, se procede a hacer la prueba estadística. Con una confianza del 0.05 y el *p-value* obtenido de $1.668e^{-5}$, se rechaza la hipótesis nula y se concluye que existen diferencias significativas, además de poder decir que el modelo *stacked two stamp classifier* es mejor que el *stamp classifier*.

- *stamp classifier* vs *separated two stamp classifier*: Según la hipótesis nula de que las medias

de ambos *accuracies* es igual y la hipótesis alternativa de que la media del *accuracy* del modelo *stamp classifier* es menor que la del *separated two stamp classifier*, se procede a hacer la prueba estadística. Con una confianza del 0.05 y el *p-value* obtenido de $1.817e^{-4}$, se rechaza la hipótesis nula y se concluye que existen diferencias significativas, además de poder decir que el modelo *separated two stamp classifier* es mejor que el *stamp classifier*.

La evidencia indica que tanto la versión *stacked* y *separated* del modelo *two stamp classifier* es mejor que el modelo *stamp classifier*. Sin embargo, al comparar las arquitecturas propuestas para saber si existen diferencias significativas entre la versión *stacked* y *separated*, se realiza otra prueba estadística de Welch. El *p-value* obtenido es de $2.749e^{-2}$, rechazando la hipótesis nula, por lo tanto existen diferencias significativas entre los mejores modelos de las arquitecturas *stacked* y *separated*, siendo la variante *stacked* mejor que *separated*.

En la Tabla 6.4 se aprecia un resumen comparativo entre las arquitecturas utilizadas en este estudio, donde se presenta el *accuracy* de validación y prueba, además del tiempo de inferencia de cada modelo (en la Sección 6.5 se profundiza más sobre la medición de tiempos de inferencia). Por otro lado, se indica el p-valor para la prueba estadística de Welch para la comparacione entre modelos, evidenciando que el mejor modelo corresponde a la variación *stacked two stamp classifier*.

Tabla 6.4: Resumen mejores modelos

modelo		*accuracy* en validación	*accuracy* en prueba	tiempo de inferencia [ms]
M0	*stamp classifier*	0.928 ± 0.004	0.923 ± 0.005	$\mathbf{0.970 \pm 0.099}$
M1	*stacked two stamp classifier*	$\mathbf{0.950 \pm 0.003}$	$\mathbf{0.937 \pm 0.003}$	1.050 ± 0.181
M2	*separated two stamp classifier*	0.946 ± 0.001	0.932 ± 0.006	1.740 ± 0.304
p-valor de la prueba de Welch en *accuracy* M0 vs M1 – M0 vs M2 – M1 vs M2		$1.668e^{-5}$	$1.817e^{-4}$	$2.749e^{-2}$

Fuente: Elaboración propia, 2021

6.5 TIEMPOS DE INFERENCIA DEL MODELO

La idea principal de este trabajo de tesis es que el modelo se ejecute en tiempo real, es decir, que el modelo esté noche a noche prediciendo qué tipo de objetos se ven en el cielo. Una medida que puede resultar de interés, es el tiempo de inferencia, es decir, cuanto tiempo se demora en predecir un conjunto de datos cualquiera. Es importante destacar, que no se toma en

cuenta los tiempos de transmisión y preprocesamiento de los datos, lo cual en un ambiente real debiese considerarse.

El tiempo de inferencia se ha medido utilizando los modelos ya entrenados, para esto se tomo un elemento cualquiera del conjunto de datos y se midió el tiempo de ejecución al momento de predecir la clase, este procedimiento se repitió 10 veces por modelo. En la Figura 6.4 se muestra el tiempo de ejecución en milisegundos y el *accuracy* en el conjunto de prueba para los 5 mejores modelos por arquitectura.

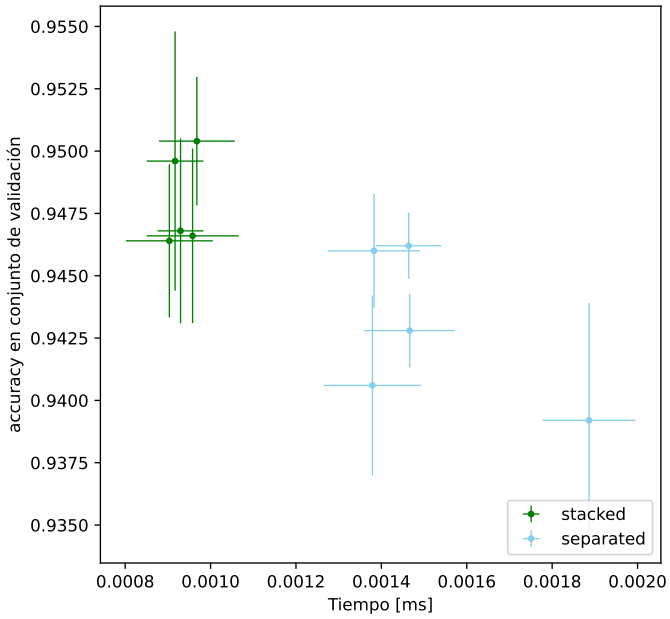


Figura 6.4: Tiempo de inferencia y *accuracy* según arquitectura. Los puntos indican la media de los 10 resultados por arquitectura y las barras de error muestran la desviación estándar de cada medida. Fuente: Elaboración propia, 2021

Se realiza una prueba estadística de Welch para comparar el tiempo de inferencia entre los modelos más rápidos y lentos por cada arquitectura. Para el modelo de *stacked two stamp classifier* se tiene un *p-value* de 0.075 por lo tanto no existen diferencias significativas. Sin embargo para el *separated two stamp classifier* se tiene un *p-value* de 0.026, el que indica que existen diferencias significativas entre el modelo que infiere más rápido y más lento de esta arquitectura.

Al comparar los tiempos de inferencia entre arquitecturas, se realiza otra prueba estadística de Welch que entrega un *p-value* de $5.251e-3$, por lo que se rechaza la hipótesis nula y se puede concluir que existe una diferencia estadísticamente significativa en las medias de los tiempos de inferencia entre las dos arquitecturas, vale decir que la arquitectura *stacked* es más rapída infiriendo que la variante *separated*.

La *pipeline* de ALeRCE funciona mediantes *batches* de datos, es decir, a través de

pequeños conjuntos de datos. Tomando el modelo *stacked two stamp classifier*, se ha realizado la estimación de inferencia a partir de diferentes tamaños de *batch*, el proceso se ha repetido 10 veces por tamaño. En la Figura 6.5 se puede apreciar que a un mayor tamaño del *batch* hay un mayor tiempo de inferencia.

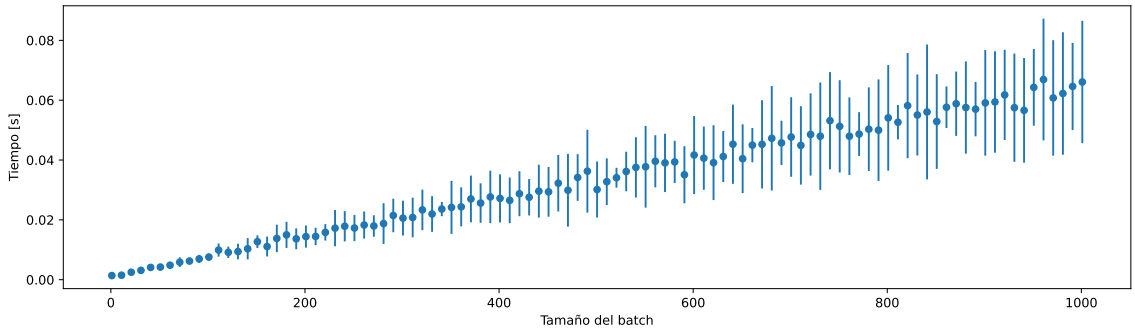


Figura 6.5: Tiempo de inferencia según tamaño de *batch*. Los puntos indican la media de los 10 resultados por tamaño de *batch* y las barras de error muestran la desviación estándar. Fuente: Elaboración propia, 2021

Los tiempos de respuesta son de gran importancia, ya que el modelo debe estar en un contexto de clasificación en tiempo real. Considerando que el menor tiempo de inferencia es de $1.378 \pm 0.067[ms]$ que corresponde a un *batch* que contiene solo un elemento, y el mayor tiempo de inferencia es de $66.078 \pm 20.451[ms]$ que es de un *batch* de 1000 objetos, se puede definir un problema sobre cuál es valor ideal para el tamaño de *batch*, en el cual se debe tomar en cuenta la tasa de alertas que se encolan en la *pipeline* en función del tiempo, ya que si se utiliza el *batch* de un sólo elemento, los tiempos de inferencia para 1000 objetos sería de $1.378 \pm 0.067[s]$.

6.6 DESEMPEÑO EN CONJUNTO NO ETIQUETADO

Otra manera de evidenciar que el modelo esté funcionando correctamente, es probarlo en un conjunto de datos no etiquetado. Según los expertos, hay algunas clases que deberían concentrarse en ciertas zonas del cielo, por ejemplo los asteroides deberían estar situados a lo largo del cinturón de asteroides en el sistema solar (estar cerca de la eclíptica), los *bogus* podrían encontrarse en zonas del cielo muy brillantes o distribuidos de manera aleatoria, por su lado las supernovas y AGN deberían estar fuera del plano galáctico y por el contrario las estrellas variables se concentran en el plano galáctico.

Para construir el conjunto de datos no etiquetado, se realizo una consulta espacial a la base de datos de ZTF que tiene ALeRCE. Para tomar zonas representativas del cielo se hicieron búsquedas en cono desde un RA $\in [0, 360]$ y Dec $\in [-90, 90]$ con saltos de 10 grados

en cada coordenada y un radio de $10 \cdot \sqrt{2}$ arcosegundos con la restricción de tener al menos 2 detecciones[1].

```
SELECT oid, meanra, meandec, ndet
FROM object
WHERE q3c_radial_query(meanra, meandec, <ra>, <dec>, <radius>) and ndet >= 2;
```

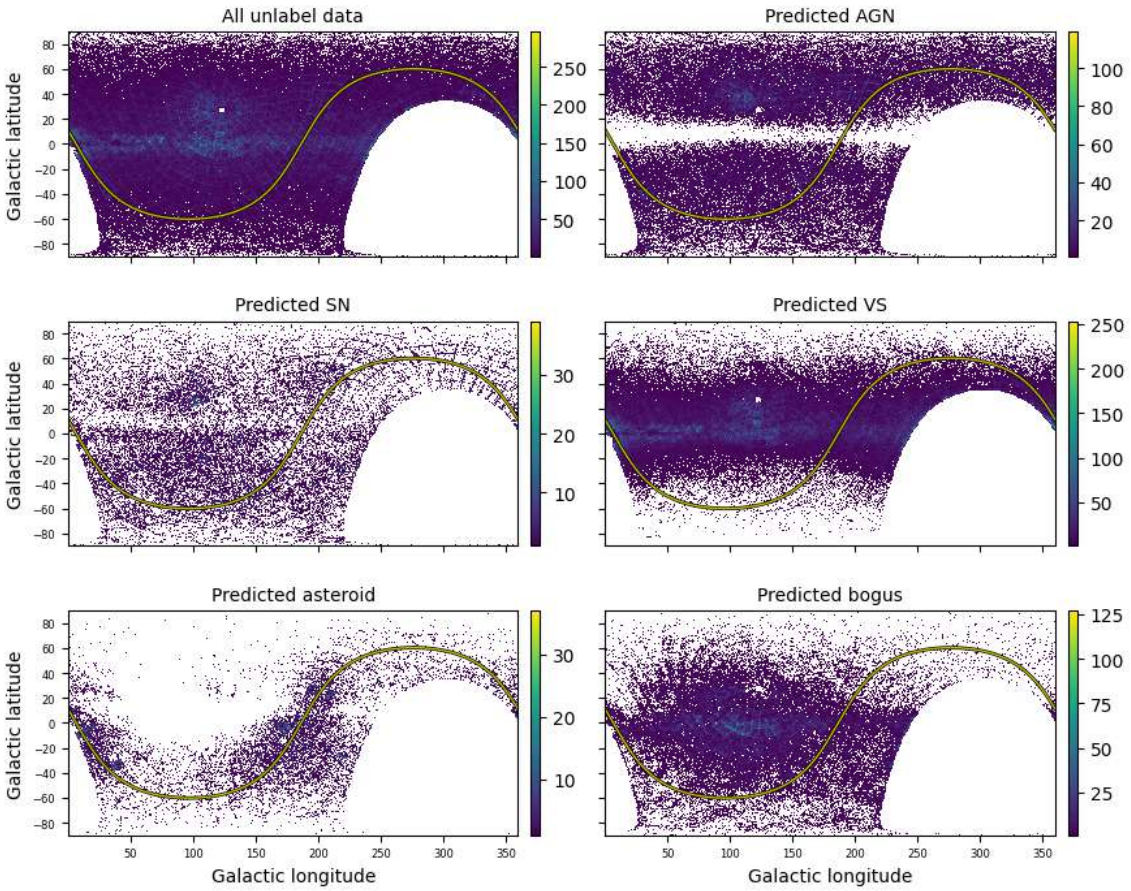


Figura 6.6: Distribución espacial para objetos en el conjunto no etiquetado por clase predicha. La barra de color indica la densidad de la zona. La eclíptica se muestra con una línea amarilla. Fuente: Elaboración propia, 2021

El conjunto de datos no etiquetado está compuesto por un total de 2.562.627 de objetos y sólo 1.445.667 cumplen con un *deltajd* menor a 100 días. Para evaluar el desempeño del modelo en este conjunto, se realiza la inferencia del modelo *stacked two stamp classifier* y se obtiene la clase de cada uno de estos objetos. Posteriormente, se realiza un gráfico de distribución espacial con las coordenadas galácticas de los objetos. En la Figura 6.6, se logra apreciar como se

[1]Hay que tener en cuenta, que no necesariamente las detecciones contienen estampillas. Esto es porque ZTF rectifica detecciones en una lista de candidatos previos, los cuales son detecciones y no detecciones que poseen solo ciertos metadatos y no contienen estampillas.

distribuyen todos estos objetos predichos dentro del campo de visión de ZTF. Además se observa si se cumple con la distribución esperada por clase.

Como es de esperar, la densidad de los objetos predichos como AGN y SN se muestran distantes del plano galáctico, lo que es correcto ya que son fuentes extragalácticas. Aún así, la clase SN tiene una distribución espacial con cierto ruido, esto puede deberse a la confusión con otras clases como asteroides y *bogus*, ya que el modelo no es capaz de discriminar propiedades de cada una de esas clases con los atributos con los que ha sido entrenado. Por otro lado, la distribución espacial de la clase VS está concentrada en el plano galáctico, mientras que la clase asteroide se encuentran cercanos a la eclíptica.

6.7 IMPORTANCIA DE METADATOS

Para corroborar el impacto de los metadatos utilizados en el modelo entrenado, se ha optado por entrenar un ***random forest*** (Breiman, 2001) solo con los metadatos. Este tipo de modelo, basado en árboles de decisión, permite obtener el grado de importancia que tienen las variables debido a la información contenida en cada árbol.

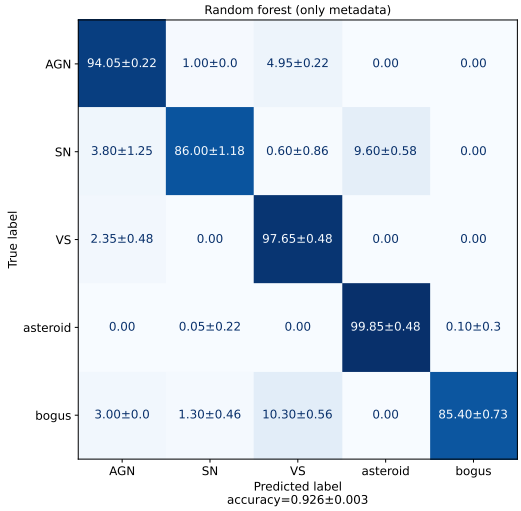


Figura 6.7: Matriz de confusión de *random forest* que solo usa metadatos en conjunto de prueba. Fuente: Elaboración propia, 2021

Para obtener la importancia, se entrenaron 20 *random forest* con 100 árboles cada uno. Luego, se calculó la media y desviación estándar de la importancia de cada característica. El modelo en sí tiene *accuracy* de 92.59 ± 0.29 en el conjunto de prueba. La matriz de confusión que encuentra en la Figura 6.7 es de los *random forests* entrenados. De esta se puede desprender

que aún se necesitan más datos o contexto que adhiera información para resolver el problema de clasificación, dado que el modelo tiende equivocarse en mayor medida en la clase SN y *bogus*.

En la Figura 6.8 se aprecia la importancia de cada metadato, dentro de los más influyentes se pueden destacar el *deltajd*, *distpsnr1* y la *magpsf* de la primera y segunda alerta. Esto concuerda con lo expuesto en la Sección 3.3, donde se presenta que el transcurso de días entre la primera y segunda alerta puede ser un buen discriminador entre clases. Por otro lado, se puede observar que los filtros utilizados no tienen una relevancia dentro de los *random forests* entrenados, siendo que para ambas alertas están al final del ranking.

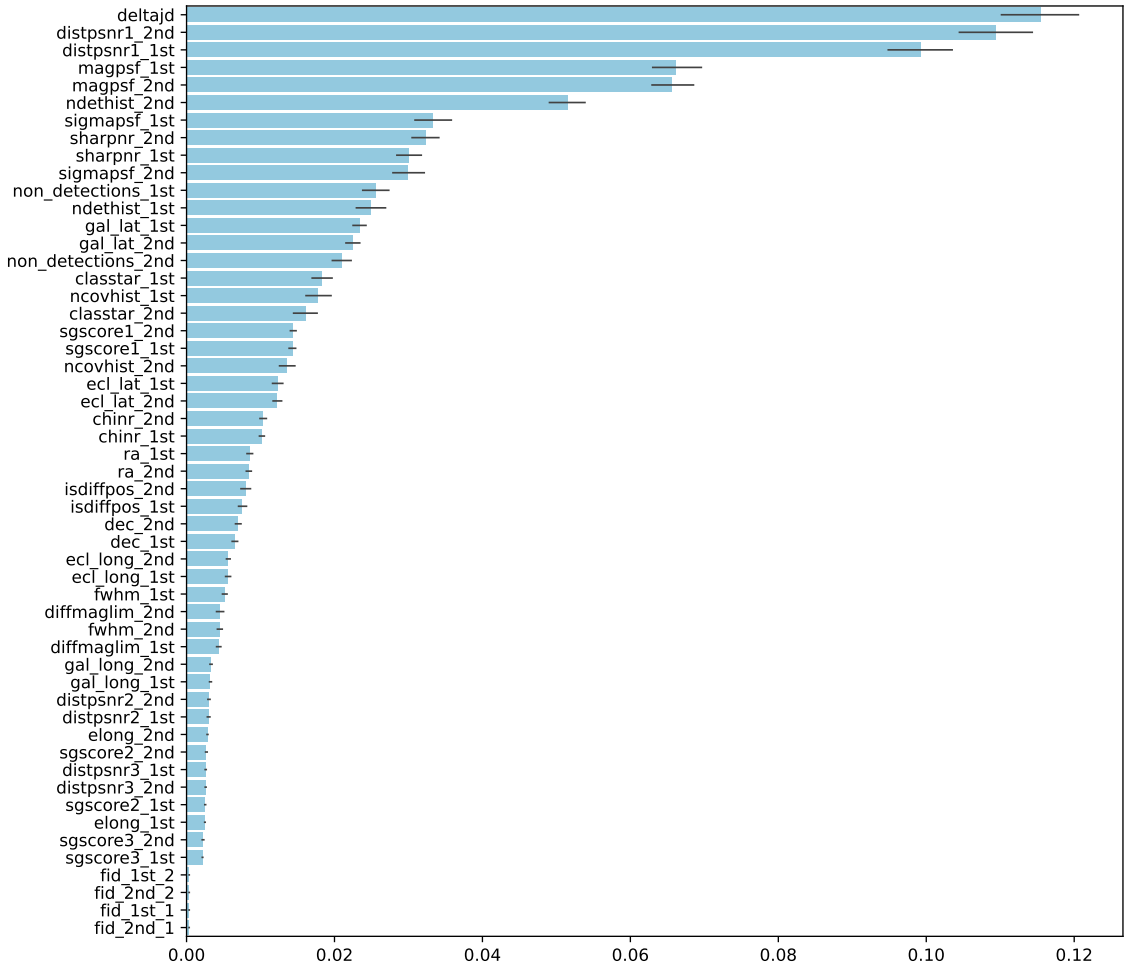
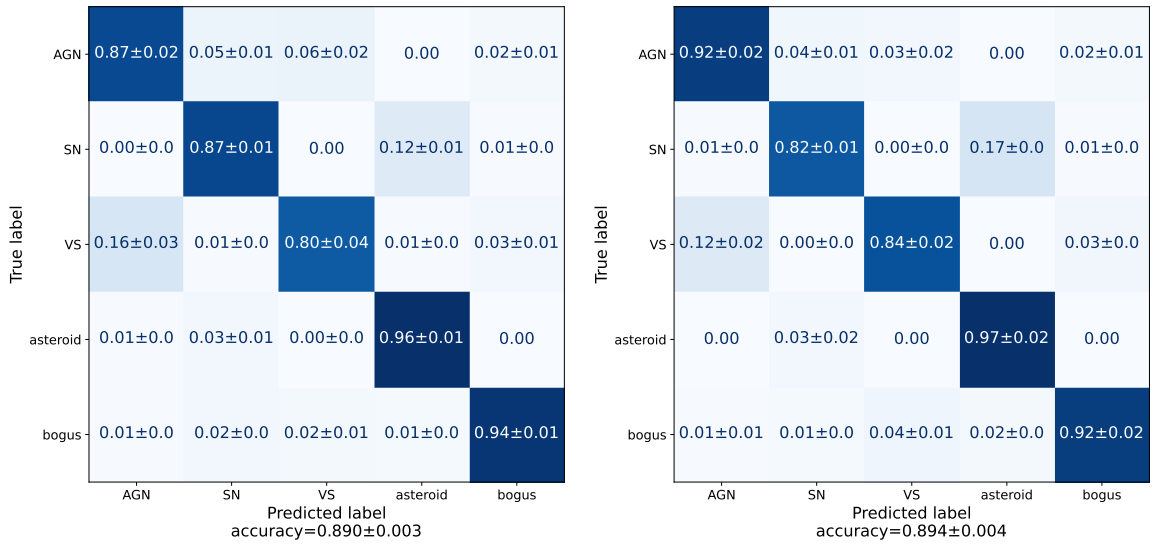


Figura 6.8: Importancia de metadatos según *random forest*. Fuente: Elaboración propia, 2021

6.8 *TWO STAMP CLASSIFIER* SIN METADATOS

Para evidenciar el impacto de los metadatos, se procede a entrenar un *stacked two stamp classifier* sin metadatos, es decir, solo utilizando las primeras dos tripletas de estampillas. Los hiper-parámetros del modelo entrenado corresponden a los que han entregado mejor resultados para esta arquitectura, vale decir los de la primera fila de la Tabla 6.2. El entrenamiento de esta variación se ha repetido 5 veces, en la Figura 6.9 se aprecia la matriz de confusión para el conjunto de validación y prueba. Como se evidencia en el conjunto de prueba se llega a 0.894 ± 0.004 de *accuracy*, los puntajes de aciertos para cada clase suelen ser peores que los modelos con metadatos.



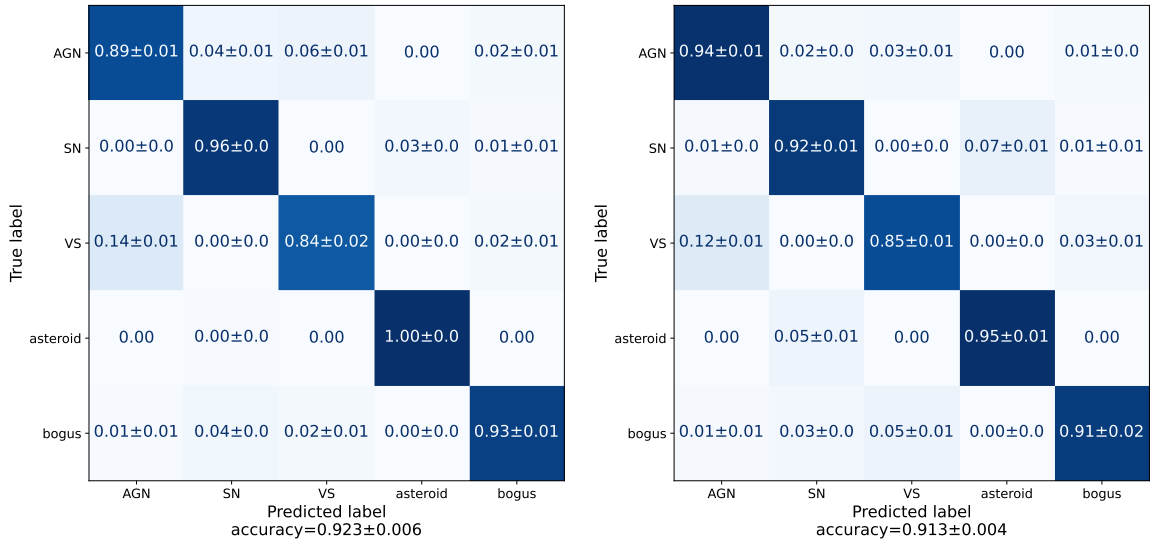
(a) Matriz de confusión para conjunto de validación

(b) Matriz de confusión para conjunto de prueba

Figura 6.9: *Stacked two stamp classifier* sin metadatos. Fuente: Elaboración propia, 2021

6.9 *TWO STAMP CLASSIFIER* CON ESTAMPILLAS Y *DELTAJD*

Al variar la arquitectura *stacked* para que tome como entrada las tripletas de estampillas y el *deltajd*, se puede apreciar como incide en la práctica dicha característica. Se ha entrenado 5 modelos con los hiper-parámetros de la primera fila de la Tabla 6.2. En la Figura 6.10 se nota que la puntuación general mejora a 0.913 ± 0.004 y se nota una clara mejoría en particular para la clase SN.



(a) Matriz de confusión para conjunto de validación

(b) Matriz de confusión para conjunto de prueba

Figura 6.10: *Stacked two stamp classifier* con estampillas y *deltajd*. Fuente: Elaboración propia, 2021

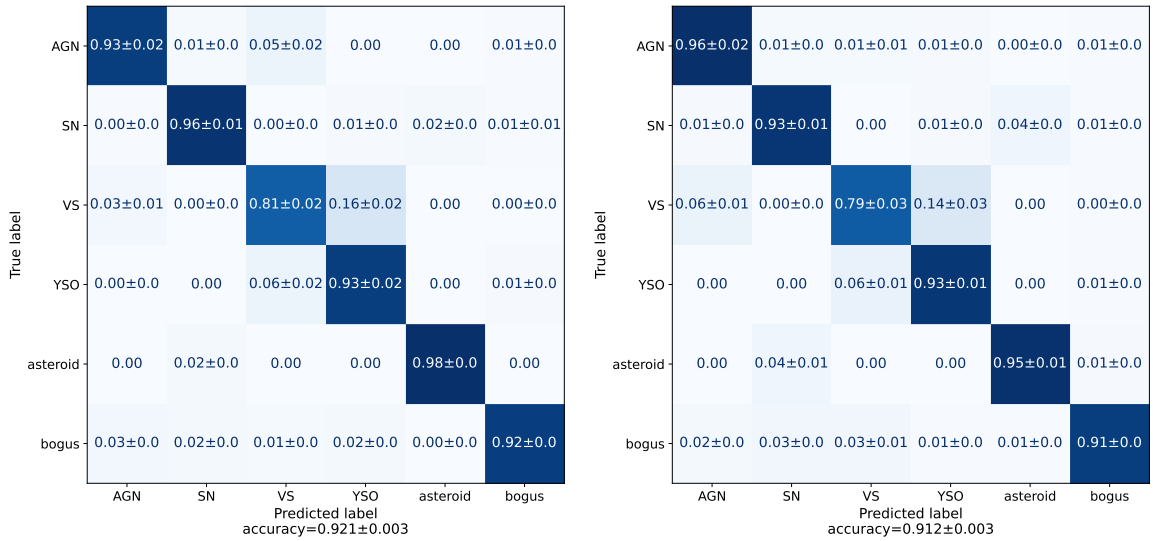
6.10 POSIBILIDAD DE AGREGAR MÁS CLASES

El mapeo de clases de Carrasco-Davis et al. (2020) se basa en la taxonomía de ALeRCE y en el trabajo realizado por Sánchez-Sáez et al. (2021). Es importante recordar que las clases propuestas por Carrasco-Davis et al. (2020) son las mismas trabajadas en esta tesis, principalmente porque es factible reconocer estas clases solo con una observación.

De manera experimental se ha tratado de ampliar la cantidad de clases a predecir, tomando las subclases mostradas en la Figura 3.8. Para esto se transformó la subetiqueta en etiqueta, posteriormente se realizó el proceso de construcción de conjunto de entrenamiento, validación y prueba de manera estratificada. Los resultados fueron prometedores principalmente para las clases asociadas a asteroides, *bogus* y supernovas, pero para clases relacionadas con AGNs y VS se tiene más problemas, ya que el modelo no logra identificar patrones que permitan una correcta clasificación. Para resolver este problema, se recomienda el uso de más observaciones o características para identificar las subclases.

Un resultado interesante, se muestra en la Figura 6.11, donde se desagrega la clase *young stellar object* (YSO) de la macro-clase VS. En la Figura A.1 del Anexo se puede evidenciar que la clase YSO se encuentra dentro del tipo de objeto estocástico y no periódico (como el resto de VS), esto puede indicar que el modelo pueda reconocer patrones que distinga YSO del resto de VS. El modelo logra un 0.912 ± 0.003 de *accuracy* en el conjunto de prueba, donde no pierde rendimiento en la clasificación de supernovas (0.93 ± 0.01). Es probable, que al agregar más datos

y/o estampillas se gane puntos de *accuracy* para la clase VS y YSO, ya que para dichas clases la evolución en el tiempo puede aportar más información que solo utilizar las estampillas de las dos primeras alertas.



(a) Matriz de confusión para conjunto de validación

(b) Matriz de confusión para conjunto de prueba

Figura 6.11: *Stacked two stamp classifier* prediciendo 6 clases. Fuente: Elaboración propia, 2021

6.11 DISCUSIÓN

Como se ha mostrado en la Subsección 6.4, los resultados expuestos por ambas arquitecturas propuestas resultan en una mejoría estadísticamente significativa en comparación al modelo *stamp classifier*. Los modelos propuestos son particularmente mejores en la clasificación de supernovas, tal como se ve en las Figuras 6.2 y 6.3. Estos resultados indican que los modelos son capaces de recuperar información del par de tripletas de estampillas, y mezclarlo con datos temporales como el *deltajd* para resolver el problema de clasificación.

Las arquitecturas propuestas tienen tiempos de inferencia que difieren entre sí, siendo el *stacked two stamp classifier* más rápido que *separated two stamp classifier*. Esto se debe principalmente a la cantidad de parámetros de la red. La versión *separated* es una arquitectura que posee dos *stamp classifier*, que operan cada alerta por separado, mientras que la versión *stacked* es un *stamp classifier* que opera las alertas de manera apilada. Por otro lado, para funcionar de manera óptima en la clasificación en tiempo real de objetos astronómicos, se debe resolver un problema de teoría de colas, donde se debe considerar el tamaño de *batch* a predecir, tasa de alertas en el tiempo y tolerancia del tiempo de respuesta.

A pesar de que los modelos entrenados tienen un buen *accuracy* en un conjunto de prueba, se ha evidenciado que la característica *deltajd* tiene una gran relevancia. Esto implica que el modelo sea sensible a la cantidad de días transcurridos entre una alerta y otra. En la Figura 6.12, se logra ver el tamaño en GB de datos transmitidos diariamente por año. Se puede observar que hay varios días hasta semanas, en que ZTF no produce alertas. Por lo tanto, la característica *deltajd* se vería adulterada e implicaría que el modelo no funcione como se espera.

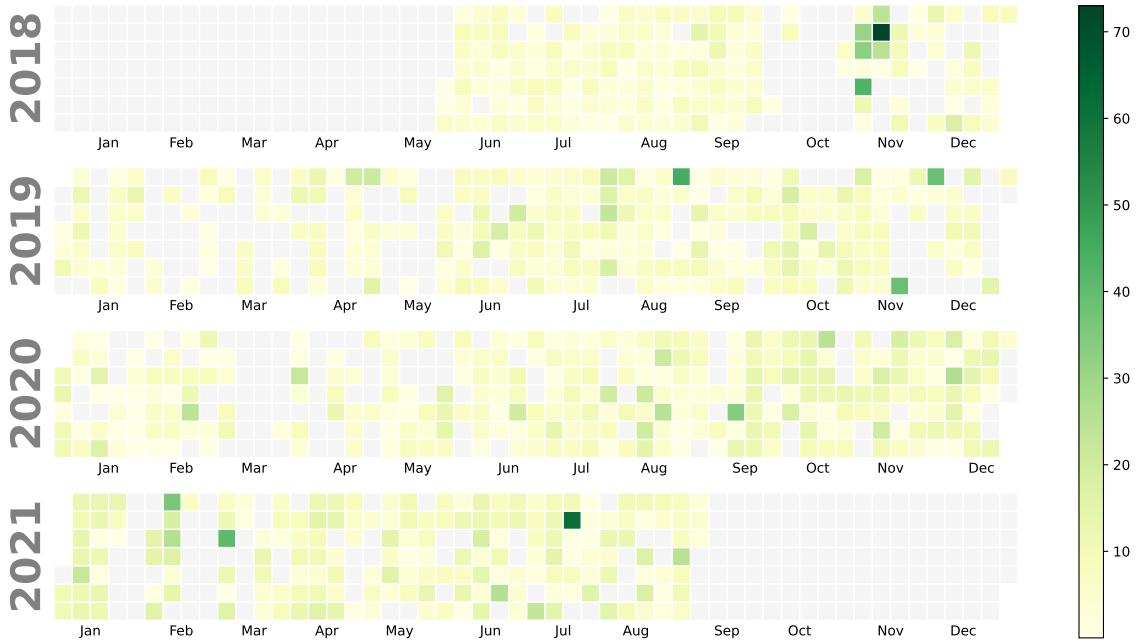


Figura 6.12: Tamaño de conjunto de alertas diarias de ZTF. Fuente: Elaboración propia, 2021

El mejor modelo escogido fue una configuración de la versión *stacked*, el cual se ocupó para la inferencia en un conjunto de datos no etiquetado. A partir de un histograma 2d expuesto en la Figura 6.6, es posible apreciar cómo se distribuyen en el espacio las predicciones para más de un millón de objetos. Como era de esperar los asteroides tienden a concentrase en la eclíptica, las VS están en mayor medida en el plano galáctico y los objetos extra-galácticos como las AGNs y supernovas se encuentran fuera del plano galáctico.

Al entrenar un *stacked two stamp classifier* sin los metadatos como entrada, el performance del modelo empeora en la detección de algunas clases (AGN, SN y VS) (véase Figura 6.9). Si el modelo se entrena con las estampillas y el *deltajd* como entradas, el rendimiento de clasificación de SN vuelve mejorar, aunque para las clases AGN y VS no parece haber afectado.

Para comprobar de manera empírica la importancia de los metadatos, el modelo de *random forest* que ha sido entrenado solo con los metadatos sugiere que las características *deltajd* en conjunto con *dispsnr1*, *magpsf* y *ndethist* de ambas alertas, son relevantes para resolver este problema. El resultado del modelo de *random forest* para dos alertas puede ser tomado como un

modelo base a resolver este problema, ya que tiene un performance aceptable para la clasificación de ciertos objetos, aunque no resultaría útil para la clasificación en tiempo real, ya que la idea principal de los clasificadores que no necesitan de muchas alertas es detectar supernovas jóvenes.

Respecto a si se pueden identificar más clases, la experimentación indica que no es suficiente sólo con dos alertas. Aún así, en la Figura 6.11 se observa que el modelo suele predecir como VS objetos que realmente son YSO, esto puede deberse a que la clase YSO es una subclase de VS según Carrasco-Davis et al. (2020), por lo tanto pueden compartir propiedades que el modelo identifica. Es posible que agregando más alertas por objeto, el modelo pueda reconocer más patrones que diferencien a más clases, en este contexto donde hay más observaciones por objeto es probable que la clase *bogus* y asteroide esté menos representada o desaparezcan, dado que los asteroides no suelen encontrarse en las mismas zonas y los *bogus* no pasen los filtros del *survey*.

CAPÍTULO 7. CONCLUSIONES

En la última década el uso de modelos de aprendizaje automático han demostrado ser herramientas útiles, que han facilitado tareas que requieren de inspección humana. Gracias a la gran cantidad de datos que se ha almacenado a lo largo de la historia, se ha podido crear modelos para diferentes áreas que tienen un excelente desempeño. En el caso particular de las redes neuronales, que se popularizaron gracias a diferentes hitos, como por ejemplo el descubrimiento del método de *backpropagation*, el diseño de redes neuronales profundas, uso de GPU para acelerar el entrenamiento, entre otros, han permitido resolver problemas aún más complejos y con mejor desempeño. Las redes neuronales se han empĺeado en diferentes áreas, lo que ha plasmado un sólido estado del arte y nuevas oportunidades de uso.

Con la llegada de una nueva era de grandes *astronomical surveys*, que permitirán observar el cielo cada vez con más detalles y cadencia, se abren nuevas necesidades y objetivos en el área de la astronomía. Grandes volúmenes de datos estructurados son transmitidos en tiempo real, lo que permite hacer ciencia sobre estos datos con modelos de aprendizaje automático. Los problemas a resolver como la detección de anomalías, *forecasting* y clasificación de objetos en categorías conocidas, se tendrán que automatizar e investigar. Este último problema ha sido abordado en este trabajo de tesis, y se resuelve con éxito mediante arquitecturas basadas en un modelo que actualmente se encuentra operativo.

Por medio del presente trabajo se ha podido construir un conjunto de datos con las etiquetas de ALeRCE, cuyos datos tienen como insumo principal las primeras dos detecciones con estampillas por objeto. Incluso se estratificaron los datos para balancear las subclases dentro de cada clase trabajada. Además se ha implementado un modelo de clasificación de objetos astronómicos a partir de las dos primeras alertas, el cual ha sido desarrollado para clasificar específicamente datos reales de ZTF. Las arquitecturas propuestas corresponden a una variación de *stamp classifier* (Carrasco-Davis et al., 2020).

7.1 SOLUCIÓN AL PROBLEMA PLANTEADO

Las arquitecturas *stacked two stamp classifier* y *separated two stamp classifier* se han desempeñado con éxito en clasificar objetos de ZTF. El *accuracy* obtenido en el conjunto de prueba es de 0.937 ± 0.003 para la variante *stacked* y 0.932 ± 0.006 para *separated*. Ambas arquitecturas resultan ser mejor que *stamp classifier*, ya que se presentan diferencias estadísticamente significativas.

El objetivo de un clasificador temprano, es decir, que necesite de pocas observa-

ciones para clasificar, es categorizar supernovas jóvenes. En este sentido, ambas arquitecturas propuestas han resultado ser mejor que la arquitectura de *stamp classifier*.

En cuanto a los tiempos de inferencia, se recomienda el uso de *stacked two stamp classifier* si se requiere de una rápida predicción, esto debido a que no existen diferencias significativas respecto al *accuracy* con la arquitectura *separated two stamp classsifier*, pero resulta ser más rápido en inferencia la arquitectura *stacked* para un *batch* de un solo objeto.

Prediciendo en un conjunto de datos no etiquetado, se ha podido construir un gráfico de distribución espacial en coordenadas galácticas, donde cada clase predicha se concentra en donde debería estar en la práctica.

La característica *deltajd* ha tenido gran relevancia en los resultados expuestos, dado que al quitar dicho dato el modelo empeora su *performance*. Aún así, el modelo que utiliza solo las estampillas y el *deltajd* tiene un rendimiento aceptable de 0.913 ± 0.004 de *accuracy* en el conjunto de prueba. Es importante mencionar que en términos de clasificación de supernovas este modelo resulta útil, dado que se disminuye el sesgo que podría existir en el uso del resto de metadatos.

Finalmente se ha corroborado la relevancia de la característica *deltajd* mediante el uso de un *random forest*, el cual indica que dicha característica es la más importante para clasificar solo usando metadatos.

7.2 ANÁLISIS DE LOS OBJETIVOS

El objetivo general de este trabajo de tesis se ha cumplido en su totalidad, pues se ha podido diseñar y desarrollar dos arquitecturas basadas en el modelo *stamp classifier*, con la salvedad de utilizar dos alertas con estampillas además de agregar componentes temporales.

En cuanto a los objetivos específicos, se ha cumplido a cabalidad cada uno de ellos. Se ha construido un conjunto de datos etiquetado y no etiquetado de las primeras dos alertas con estampillas. Se han diseñado arquitecturas que permitan resolver el problema de clasificación de las dos primeras alertas de un objeto, además de entrenar dichos modelos. Los modelos han sido evaluados y se ha podido comparar con el modelo de *stamp classifier*. También se ha medido la importancia del componente temporal añadido al problema, analizando su influencia en los resultados expuestos.

7.3 ACIERTOS Y DESACIERTOS

Para el desarrollo de este trabajo de tesis se ha obtenido nuevo conocimiento acerca de varias tecnologías y librerías de procesamiento masivo de datos. Uno de los aciertos en el transcurso de esta tesis ha sido el uso de servicios de AWS, principalmente el procesamiento distribuido de EMR, el cual permitió la construcción del conjunto de datos etiquetado y no etiquetado, donde la estimación del costo de este servicio ha sido notablemente barato, ya que con menos de $5USD se ha podido utilizar un clúster de 30 nodos.

Por otro lado, la serialización de TFRecords ha sido un total acierto en términos de recursos, puesto que ha permitido entrenar el modelo de manera eficaz y eficiente, ya que la lectura de este formato no sobre-utiliza recursos, dado que mientras una porción de datos está siendo procesada en GPU, un proceso libera memoria RAM y carga otra porción de datos, que se va a transferir a la GPU para ser procesado y así sucesivamente.

En términos de esperar que el modelo solo con dos alertas con estampillas sea capaz de predecir más clases, se puede decir, que no es factible. Esto último se debe a que se necesita más información del momento e historial del objeto, dado que con pocos antecedentes es difícil identificar clases que su principal característica es variar en el tiempo.

7.4 TRABAJO FUTURO

En primer lugar, la puesta en producción y optimización de recursos del modelo, resulta ser un trabajo futuro, puesto que el número de *batches* a utilizar no es trivial. En términos operativos, la puesta en producción considera la compatibilidad con la *pipeline* de ALeRCE, es decir, respetar los esquemas y las tecnologías que utilizan.

Con los antecedentes de que el componente temporal influye en la clasificación de objetos astronómicos e influye en mayor medida para objetos que varían en un corto período de tiempo, como lo son las supernovas, es posible extender el número de entradas a n alertas con estampillas. En cierta medida el problema se vuelve complejo en términos de la construcción del conjunto de datos para entrenar y evaluar el modelo, ya que los tiempos de obtención y uso recursos podría incrementar a lo más en un factor n. Por otro lado, emplear otras técnicas para clasificar datos como lo puede ser *attention* o técnicas que se han mostrado con éxito en datos simulados como la red recurrente convolucional (RCNN) de Carrasco-Davis et al. (2019) puede suponer nuevas formas de clasificar.

Otro trabajo futuro que se puede realizar, es poder clasificar satélites cercanos a la

tierra (como los de SpaceX). ZTF ya ha avistado satélites[1] anteriormente, y se espera que para un futuro la cantidad de satélites incremente sustancialmente, podría ser útil poder filtrarlos con un mecanismos de clasificación.

[1] Ejemplo de satélite: `https://alerce.online/object/ZTF21aaodknp`

GLOSARIO

ANN: Por su sigla en inglés *Artificial Neural Network*, son modelos matemáticos inspirados en el funcionamiento de las neuronas biológicas que permiten resolver problemas de aprendizaje automático.

API: Por su sigla en inglés *Application Programming Interface*, son interfaces que permiten ejecutar servicios o rutinas para cierto lenguajes de programación. Generalmente son servicios web, pero también pueden ser librerías de terceros. CNN: Por su sigla en inglés *Convolutional Neural Network*, son ANN que utilizan la operación de convoloción para extraer características elementales de las entradas.

CPU: Por su sigla en inglés *Central Processing Unit*, es un *hardware* que permite procesar datos.

GPU: Por su sigla en inglés *Graphics Processing Unit*, corresponde a un *hardware* diseñado para ejecutar cálculos complejos.

Cross match: Cruzar información entre catálogos astronómicos mediante su posición.

Hardware: Componentes físicos y tangibles de un computador.

Software: Rutinas diseñadas por personas para efectuar una tarea en específico en un sistema digital.

REFERENCIAS BIBLIOGRÁFICAS

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.
URL `https://www.tensorflow.org/`

Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, *22*(10), 1533–1545.

Aihara, H., Arimoto, N., Armstrong, R., Arnouts, S., Bahcall, N. A., Bickerton, S., Bosch, J., Bundy, K., Capak, P. L., Chan, J. H., et al. (2018). The hyper suprime-cam ssp survey: overview and survey design. *Publications of the Astronomical Society of Japan*, *70*(SP1), S4.

Anaconda Inc. (2020). Anaconda Software Distribution.
URL `https://docs.anaconda.com/`

Atlas, L., Homma, T., & Marks, R. (1987). An artificial neural network for spatio-temporal bipolar patterns: Application to phoneme classification. In *Neural Information Processing Systems*, (pp. 31–40).

Bailey, S., Aragon, C., Romano, R., Thomas, R. C., Weaver, B. A., & Wong, D. (2007). How to find more supernovae with less work: Object classification techniques for difference imaging. *The Astrophysical Journal*, *665*(2), 1246–1253.
URL `https://doi.org/10.1086/519832`

Beck, K. (2000). *Extreme programming explained: embrace change*. addison-wesley professional.

Bellm, E. C., Kulkarni, S. R., Graham, M. J., Dekany, R., Smith, R. M., Riddle, R., Masci, F. J., Helou, G., Prince, T. A., Adams, S. M., et al. (2018). The Zwicky Transient Facility: system overview, performance, and first results. *Publications of the Astronomical Society of the Pacific*, *131*(995), 018002.

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

Bertin, E., & Arnouts, S. (1996). Sextractor: Software for source extraction. *Astron. Astrophys. Suppl. Ser.*, *117*(2), 393–404.
URL `https://doi.org/10.1051/aas:1996164`

Bloom, J., Richards, J., Nugent, P., Quimby, R., Kasliwal, M., Starr, D., Poznanski, D., Ofek, E., Cenko, S., Butler, N., et al. (2012). Automating discovery and classification of transients and variable stars in the synoptic survey era. *Publications of the Astronomical Society of the Pacific*, *124*(921), 1175.

Bogacz, B., & Mara, H. (2020). Period classification of 3d cuneiform tablets with geometric neural networks. In *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, (pp. 246–251). IEEE.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Brough, S., Collins, C., Demarco, R., Ferguson, H. C., Galaz, G., Holwerda, B., Martinez-Lombilla, C., Mihos, C., & Montes, M. (2020). The Vera Rubin Observatory Legacy Survey of Space and Time and the Low Surface Brightness universe. *arXiv preprint arXiv:2001.11067*.

Cabrera-Vives, G., Reyes, I., Förster, F., Estévez, P. A., & Maureira, J.-C. (2016). Supernovae detection by using convolutional neural networks. In *2016 International Joint Conference on Neural Networks (IJCNN)*, (pp. 251–258). IEEE.

Cabrera-Vives, G., Reyes, I., Förster, F., Estévez, P. A., & Maureira, J.-C. (2017). Deep-HiTS: Rotation invariant convolutional neural network for transient detection. *arXiv preprint arXiv:1701.00458*.

Carey, S. S. (2011). *A beginner's guide to scientific method*. Nelson Education.

Carrasco-Davis, R., Cabrera-Vives, G., Förster, F., Estévez, P. A., Huijse, P., Protopapas, P., Reyes, I., Martínez-Palomera, J., & Donoso, C. (2019). Deep learning for image sequence classification of astronomical events. *Publications of the Astronomical Society of the Pacific*, *131*(1004), 108006.

Carrasco-Davis, R., Reyes, E., Valenzuela, C., Förster, F., Estévez, P. A., Pignata, G., Bauer, F. E., Reyes, I., Sánchez-Sáez, P., Cabrera-Vives, G., et al. (2020). Alert Classification for the ALeRCE Broker System: The Real-time Stamp Classifier. *arXiv preprint arXiv:2008.03309*.

Charnock, T., & Moss, A. (2017). Deep recurrent neural networks for supernovae classification. *The Astrophysical Journal Letters*, *837*(2), L28.

Cireşan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, *22*(12), 3207–3220.

Cutri, R. e., et al. (2014). Vizier online data catalog: Allwise data release (cutri+ 2013). *VizieR Online Data Catalog*, (pp. II–328).

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, *2*(4), 303–314.

Djorgovski, S. G., Mahabal, A., Drake, A., Graham, M., & Donalek, C. (2013). Sky surveys. *Planets, Stars, and Stellar Systems*, *2*, 223–281.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, *12*(7).

Förster, F., Cabrera-Vives, G., Castillo-Navarrete, E., Estévez, P., Sánchez-Sáez, P., Arredondo, J., Bauer, F., Carrasco-Davis, R., Catelan, M., Elorrieta, F., et al. (2020). The Automatic Learning for the Rapid Classification of Events (ALeRCE) Alert Broker. *arXiv preprint arXiv:2008.03303*.

Fukushima, K., & Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, (pp. 267–285). Springer.

Garg, N. (2013). *Apache kafka*. Packt Publishing Birmingham.

Graham, M. J., Kulkarni, S., Bellm, E. C., Adams, S. M., Barbarino, C., Blagorodnova, N., Bodewits, D., Bolin, B., Brady, P. R., Cenko, S. B., et al. (2019). The zwicky transient facility: Science objectives. *Publications of the Astronomical Society of the Pacific*, *131*(1001), 078001.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., & Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362.
URL `https://doi.org/10.1038/s41586-020-2649-2`

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, (pp. 1026–1034).

Hernández, R., Fernández, C., & Baptista, P. (2010). Metodología de la investigación 5ta, editor. *México DF: McGrawHill*.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, *18*(7), 1527–1554.

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

Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, *148*(3), 574–591.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Janesick, J. R. (2001). *Scientific charge-coupled devices*, vol. 83. SPIE press.

Kimura, A., Takahashi, I., Tanaka, M., Yasuda, N., Ueda, N., & Yoshida, N. (2017). Single-epoch supernova classification with deep convolutional neural networks. In *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, (pp. 354–359). IEEE.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kippenhahn, R., Weigert, A., & Weiss, A. (1990). *Stellar structure and evolution*, vol. 192. Springer.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. (2016). Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides, & B. Schmidt (Eds.) *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, (pp. 87 – 90). IOS Press.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*, 1097–1105.

Laher, R. R., Masci, F. J., Groom, S., Rusholme, B., Shupe, D. L., Jackson, E., Surace, J., Flynn, D., Landry, W., Terek, S., et al. (2017). Processing images from the zwicky transient facility. *arXiv preprint arXiv:1708.01584*.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541–551.

Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, vol. 30, (p. 3). Citeseer.

Martínez-Palomera, J., Förster, F., Protopapas, P., Maureira, J. C., Lira, P., Cabrera-Vives, G., Huijse, P., Galbany, L., De Jaeger, T., González-Gaitán, S., et al. (2018). The high cadence transit survey (hits): Compilation and characterization of light-curve catalogs. *The Astronomical Journal*, *156*(5), 186.

Masci, F. J., Laher, R. R., Rusholme, B., Shupe, D. L., Groom, S., Surace, J., Jackson, E., Monkewitz, S., Beck, R., Flynn, D., et al. (2018). The zwicky transient facility: Data processing, products, and archive. *Publications of the Astronomical Society of the Pacific*, *131*(995), 018003.

Matson, J. (2012). You are here: How astronomical surveys are pinpointing our place in the cosmos.
URL https://www.scientificamerican.com/article/gaia-galaxy-mapping/

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

Mertz, A., & Slough, W. (2007). Graphics with tikz. *The PracTEX Journal*, *1*.

Mickaelian, A. (2012). Large astronomical surveys, catalogs and databases. *Open Astronomy*, *21*(3), 331–339.

Mickaelian, A. M. (2016). Astronomical surveys and big data. *Open Astronomy*, *25*(1), 75–88.

Milone, E. F. (1993). *Light Curve Modeling of Eclipsing Binary Stars*. Springer.

Möller, A., Peloton, J., Ishida, E. E., Arnault, C., Bachelet, E., Blaineau, T., Boutigny, D., Chauhan, A., Gangler, E., Hernandez, F., et al. (2020). Fink, a new generation of broker for the LSST community. *Monthly Notices of the Royal Astronomical Society*.

Narayan, G., Zaidi, T., Soraisam, M., Collaboration, A., et al. (2018). Machine Learning-Based Transient Brokers for Real-Time Classification of the LSST Alert Stream. *AAS*, *231*, 332–08.

Patterson, M. T., Bellm, E. C., Rusholme, B., Masci, F. J., Juric, M., Krughoff, K. S., Golkhou, V. Z., Graham, M. J., Kulkarni, S. R., Helou, G., et al. (2018). The Zwicky Transient Facility Alert Distribution System. *Publications of the Astronomical Society of the Pacific*, *131*(995), 018001.

Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., & Guibas, L. J. (2016). Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 5648–5656).

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, *12*(1), 145–151.

Reyes Jainaga, I. A. (2019). Monitoreo y aprendizaje de redes neuronales utilizando medidas de información y su aplicación en detección de eventos astronómicos transitorios.

Rodrigo, C., & Solano, E. (2020). The SVO Filter Profile Service. In *Contributions to the XIV.0 Scientific Meeting (virtual) of the Spanish Astronomical Society*, (p. 182).

Rosenblatt, F. (1962). Principles of neurodynamics: Perceptions and the theory of brain mechanisms.

Sánchez-Sáez, P., Reyes, I., Valenzuela, C., Förster, F., Eyheramendy, S., Elorrieta, F., Bauer, F. E., Cabrera-Vives, G., Estévez, P. A., Catelan, M., Pignata, G., Huijse, P., Cicco, D. D., Arévalo, P., Carrasco-Davis, R., Abril, J., Kurtev, R., Borissova, J., Arredondo, J., Castillo-Navarrete, E., Rodriguez, D., Ruz-Mieres, D., Moya, A., Sabatini-Gacitúa, L., Sepúlveda-Cobo, C., & Camacho-Iñiguez, E. (2021). Alert classification for the ALeRCE broker system: The light curve classifier. *The Astronomical Journal*, *161*(3), 141.
URL https://doi.org/10.3847/1538-3881/abd5c1

Shallue, C. J., & Vanderburg, A. (2018). Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90. *The Astronomical Journal*, *155*(2), 94.

Smith, K. (2019). Lasair: the transient alert broker for LSST: UK. *eeu*, (p. 51).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Stetson, P. B. (1987). DAOPHOT - a computer program for crowded-field stellar photometry. *Publications of the Astronomical Society of the Pacific*, *99*, 191.
URL https://doi.org/10.1086/131977

Tonry, J., Denneau, L., Heinze, A., Stalder, B., Smith, K., Smartt, S., Stubbs, C., Weiland, H., & Rest, A. (2018). Atlas: a high-cadence all-sky survey system. *Publications of the Astronomical Society of the Pacific*, *130*(988), 064505.

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017). Forecasting stock prices from the limit order book using convolutional neural networks. In *2017 IEEE 19th Conference on Business Informatics (CBI)*, vol. 1, (pp. 7–12). IEEE.

Tyson, J. A. (2019). Cosmology data analysis challenges and opportunities in the LSST sky survey. In *Journal of Physics: Conference Series*, vol. 1290, (p. 012001). IOP Publishing.

Waskom, M. L. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021.
URL https://doi.org/10.21105/joss.03021

Welch, B. L. (1947). The generalization of 'student's'problem when several different population varlances are involved. *Biometrika*, *34*(1-2), 28–35.

Wenger, M., Ochsenbein, F., Egret, D., Dubois, P., Bonnarel, F., Borde, S., Genova, F., Jasniewicz, G., Laloë, S., Lesteven, S., et al. (2000). The SIMBAD astronomical database-The CDS reference database for astronomical objects. *Astronomy and Astrophysics Supplement Series*, *143*(1), 9–22.

Werbos, P. J. (1994). *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*, vol. 1. John Wiley & Sons.

Wes McKinney (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt, & Jarrod Millman (Eds.) *Proceedings of the 9th Python in Science Conference*, (pp. 56 – 61).

Wright, D. E., Lintott, C. J., Smartt, S. J., Smith, K. W., Fortson, L., Trouille, L., Allen, C. R., Beck, M., Bouslog, M. C., Boyer, A., et al. (2017). A transient search using combined human and machine classifications. *Monthly Notices of the Royal Astronomical Society*, *472*(2), 1315–1323.

York, D. G., Adelman, J., Anderson Jr, J. E., Anderson, S. F., Annis, J., Bahcall, N. A., Bakken, J., Barkhouser, R., Bastian, S., Berman, E., et al. (2000). The sloan digital sky survey: Technical summary. *The Astronomical Journal*, *120*(3), 1579.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., et al. (2016). Apache Spark: A unified engine for big data processing. *Communications of the ACM*, *59*(11), 56–65.

Zhang, W., Itoh, K., Tanida, J., & Ichioka, Y. (1990). Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Applied optics*, *29*(32), 4790–4797.

ANEXO A. TAXONOMÍA DE ALERCE

Los objetos astronómicos que se encuentran en el universo pueden categorizase según distintas características de interés. Förster et al. (2020) elabora una organización de tipos de objetos según características en común. La estructura de la taxonomía tiene forma de árbol, en donde los niveles superiores se ve el nivel de clasificación que distingue si una alerta es real o *bogus*. En los niveles inferiores se puede ver la rama de transientes (SNIa, SNIbc, SNII, SLSN), estocásticos (CV/Nova, Blazar, QSO, AGN, YSO) y periódicos (LPV, Cepheid, RRL, EB, *periodic-other*).

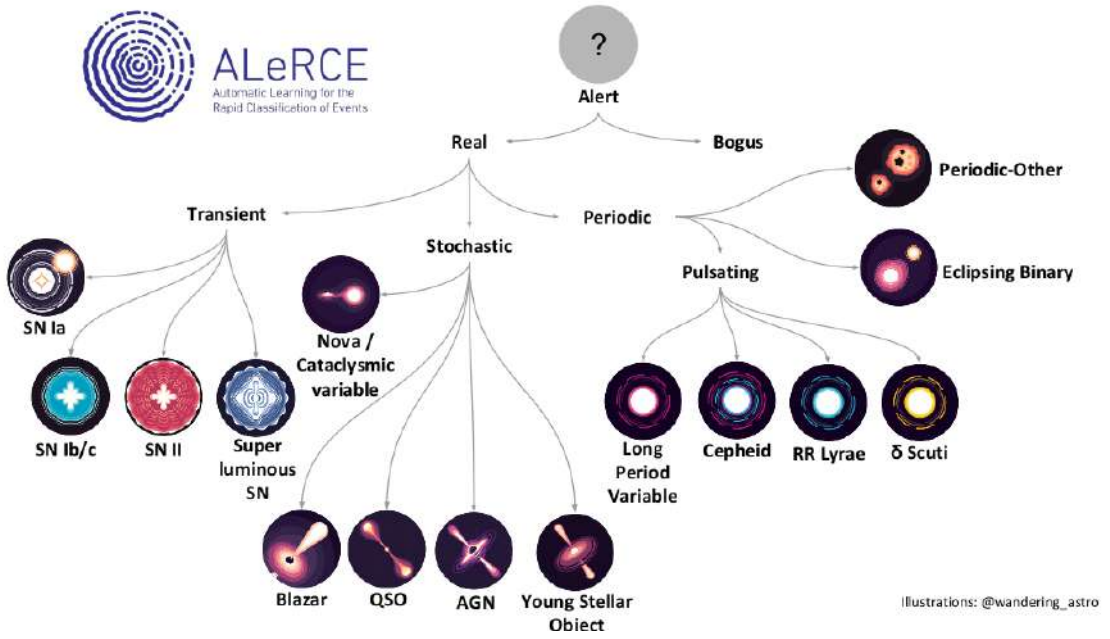


Figura A.1: Taxonomía de ALeRCE. Fuente: Förster et al. (2020)

Los trabajos de Carrasco-Davis et al. (2020) y Sánchez-Sáez et al. (2021) están basados en esta organización de categorías.

ANEXO B. ALGORITMO DE CORTE Y BALANCEO DE SUB-ETIQUETAS

Para conformar el conjunto de datos con clases, donde éstas sean representadas por una cantidad justa de subclases, se ha diseñado un algoritmo de fuerza bruta que va quitando muestras de clases hasta quedar con una cantidad de subclases que sumen un número arbitrario de corte. En este caso el número de corte es 30.000. En Algoritmo B.1 se muestra el pseudocódigo del algoritmo.

Algoritmo B.1: Algoritmo de corte de etiquetas.

Entrada: `número de corte`, `dataframe`

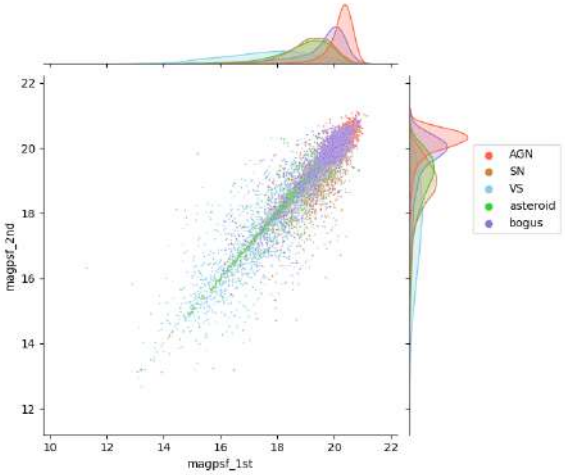
Salida: `dataframe`

```
for clase en dataframe do
    while Cantidad de elementos en clase > número de corte do
        obtener subclase mayoritaria
        eliminar de dataframe un elemento de subclase mayoritaria de clase
    end while
end for
return Retornar dataframe
```

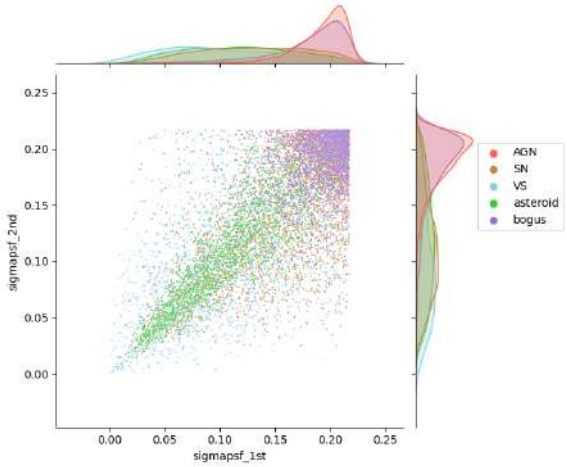
De esta forma se puede asegurar que cada clase va a tener un total de 30.000 y que no va a predominar ciertas subclases en ese número de corte.

ANEXO C. ANÁLISIS DEL RESTO DE LOS METADATOS

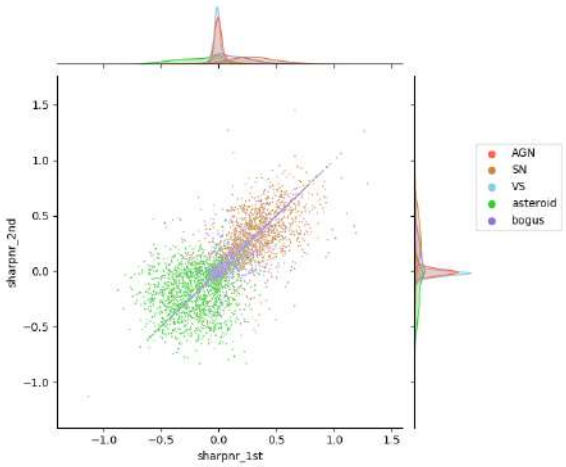
En la Tabla 3.2 se define cada metadato y se especifica los rangos de posibles valores. Para un análisis exploratorio de los datos, se realizan gráficos para cada metadato en la primera y segunda alerta, además de mostrar la distribución de los datos en los ejes x e y. En las Figuras C.1, C.2 y C.3 se logra apreciar dichos gráficos. En estos se aprecia, como algunos metadatos como *magpsf*, *sigmapsf*, *sharpnr*, entre otros pueden servir como discriminador de clases. Hay otros casos, como *ra*, *dec*, *dispsnr1*, *dispsnr2*, *dispsnr3*, *sgscore1*, *sgscore2* y *sgscore3* tienen alta correlación entre la primera y segunda alerta, lo que que tiene sentido ya que para objetos reales no deberían cambiar en el tiempo.



(a) *magpsf* para la primera y segunda alerta

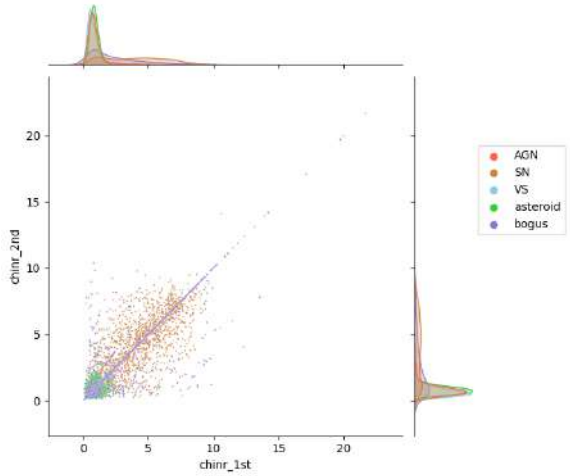


(b) *sigmapsf* para la primera y segunda alerta

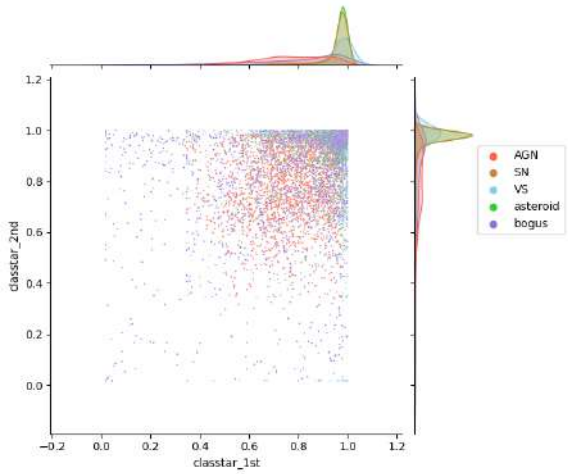


(c) *sharpnr* para la primera y segunda alerta

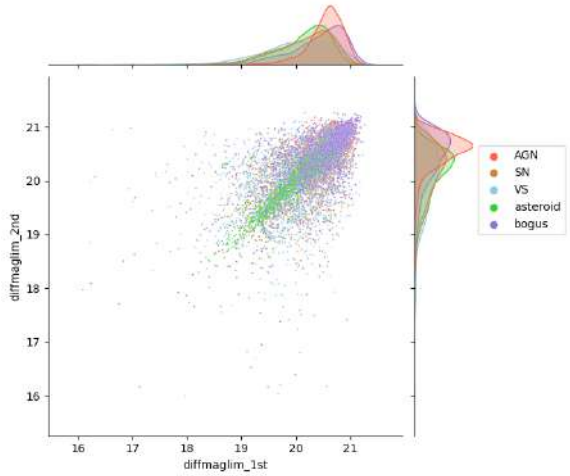
Figura C.1: Primera porción de metadatos utilizados. Fuente: Elaboración propia, 2021



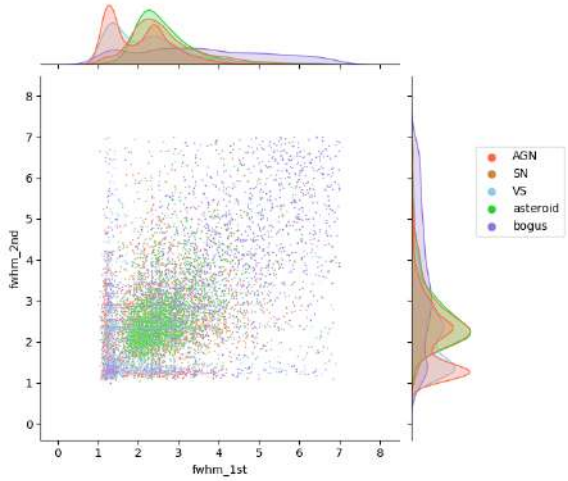
(a) *chinr* para la primera y segunda alerta



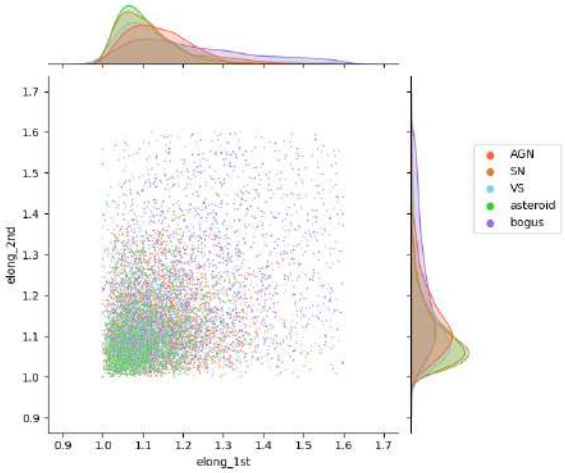
(b) *classtar* para la primera y segunda alerta



(c) *diffmaglim* para la primera y segunda alerta

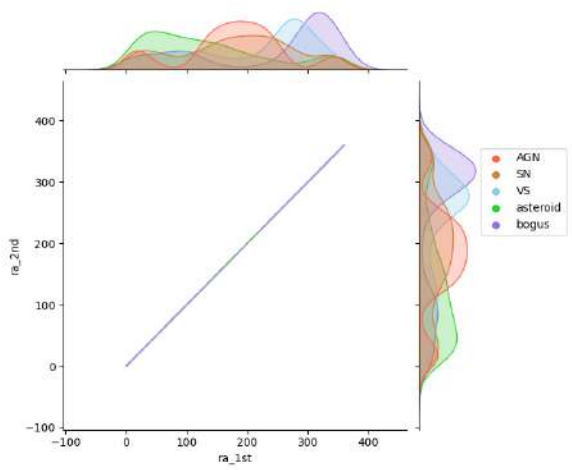


(d) *fwhm* para la primera y segunda alerta

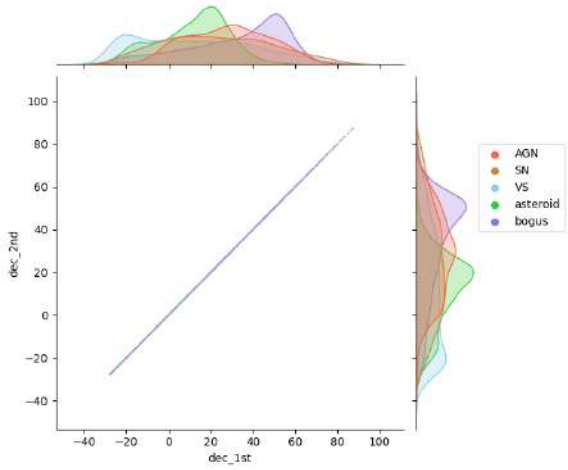


(e) *elong* para la primera y segunda alerta

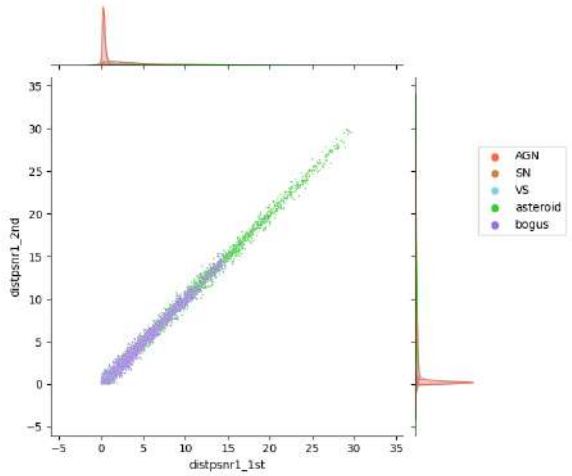
Figura C.2: Segunda porción de metadatos utilizados. Fuente: Elaboración propia, 2021



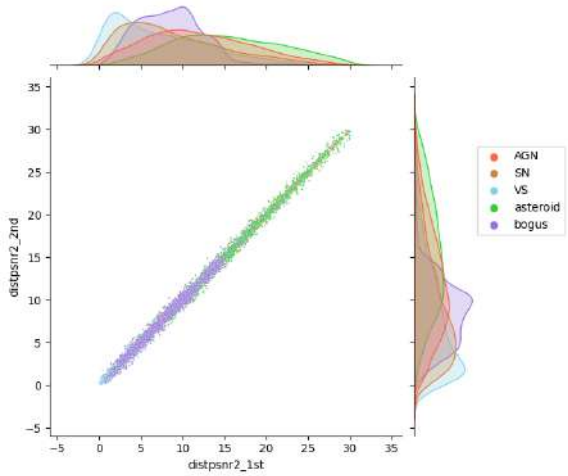
(a) *ra* para la primera y segunda alerta



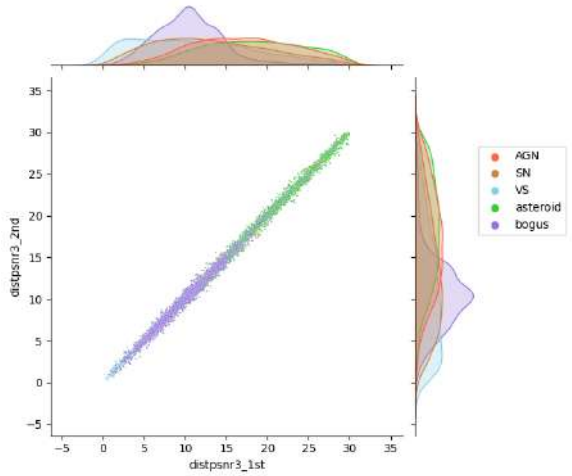
(b) *dec* para la primera y segunda alerta



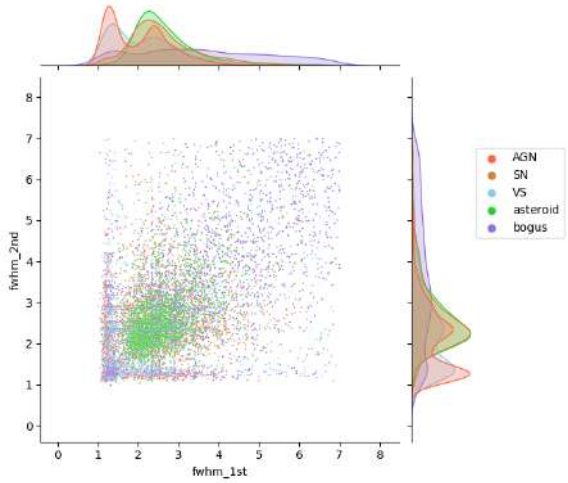
(c) *distpsnr1* para la primera y segunda alerta



(d) *distpsnr2* para la primera y segunda alerta

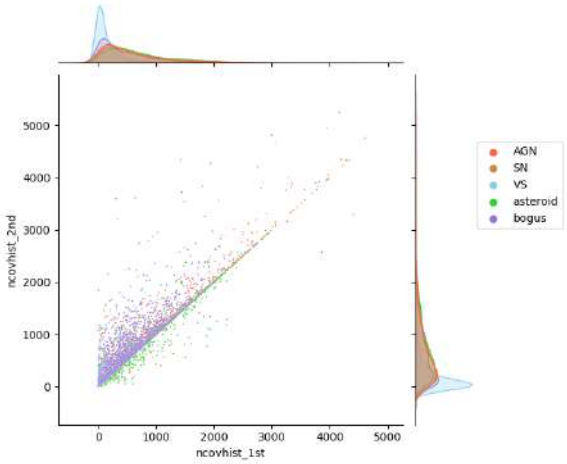


(e) *distpsnr3* para la primera y segunda alerta

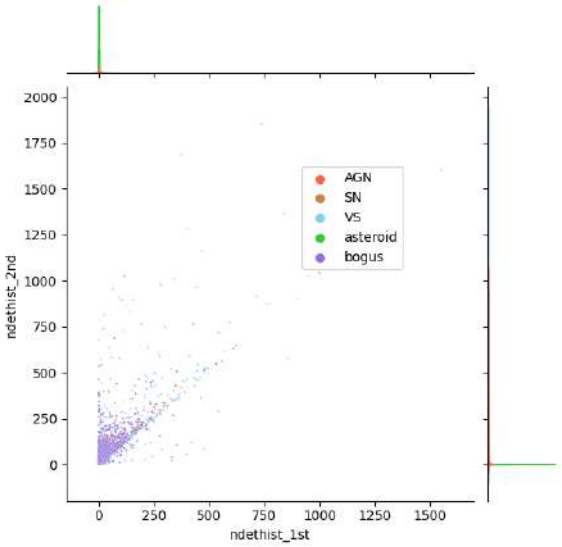


(f) *fwhm* para la primera y segunda alerta

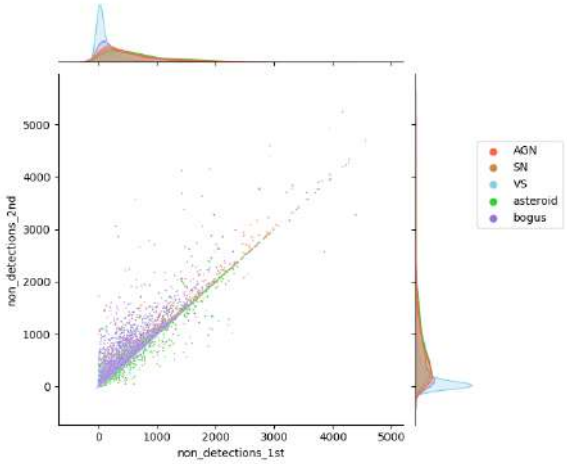
Figura C.3: Tercera porción de metadatos utilizados. Fuente: Elaboración propia, 2021



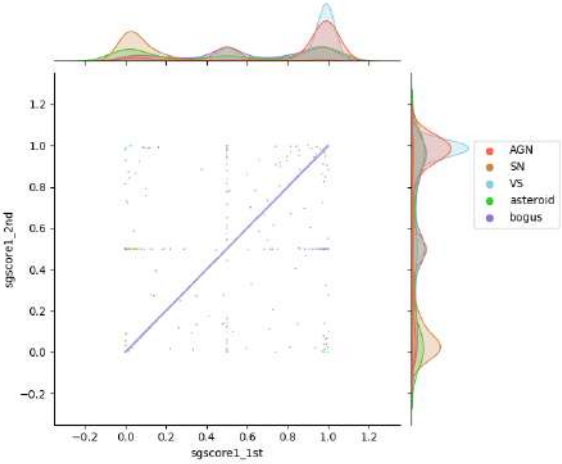
(a) *ncovhist* para la primera y segunda alerta



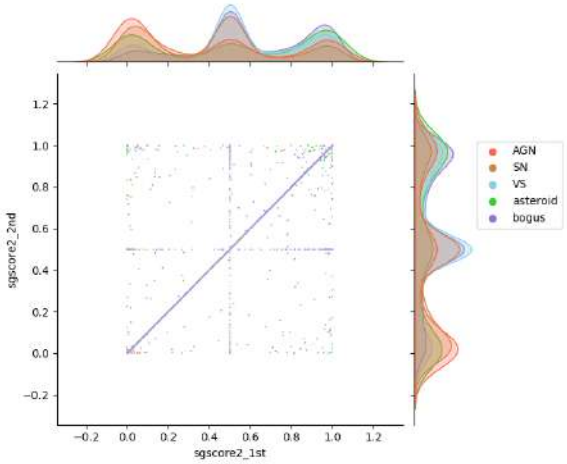
(b) *ndethist* para la primera y segunda alerta



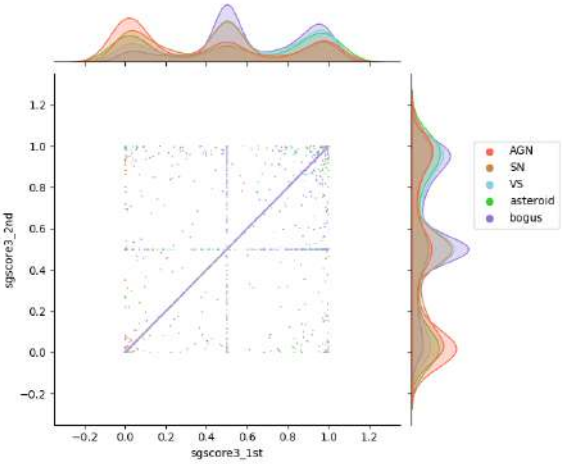
(c) *non_detections* para la primera y segunda alerta



(d) *sgscore1* para la primera y segunda alerta



(e) *sgscore2* para la primera y segunda alerta



(f) *sgscore3* para la primera y segunda alerta

Figura C.4: Cuarta porción de metadatos utilizados. Fuente: Elaboración propia, 2021