Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Eléctrica

Informe Final

Proyecto 7B Detección de Supernovas

Curso:	EL4106-Inteligencia Computacional
Integrantes:	Vincko Fabres A.
	Jorge Espejo M.
Profesor:	Pablo Estévez V.
Auxiliar:	Ignacio Reyes J.
Ayudante G7B:	Bastián Andreas G. Labbé
Ayudantes:	Andrés González F.
	Daniel Baeza M.
	Francisca Cona F.
	Javier Molina F.
	Óscar Pimentel
	Pablo A. Montero S.
	Roberto Cholaky M.

Fecha de entrega: 16 de Diciembre de 2021
Santiago, Chile

Índice de Contenidos

Índice de Figuras

Índice de Tablas

1. Introducción

Motivación

La masiva cantidad de datos astronómicos que se generan día a día actualmente ha hecho necesario conseguir formas de optimizar la forma en que se procesan, analizan y clasifican estos, los que muchas veces deben verse imagen por imagen, señal por señal, objeto por objeto, etc. Entre las herramientas que han sobresalido para apoyar en la clasificación y análisis de datos, sobre todo de grandes cantidades, se encuentran los algoritmos e implementaciones del área de Machine Learning y Deep Learning, que han tenido un crecimiento y capacidad cada vez más sobresalientes para el manejo y clasificación de datos. Una tarea que se realiza en la astronomía es la "clasificación temprana", el que tiene como idea principal realizar una clasificación a las primeras detecciones de objetos, de forma de obtener en forma rápida el tipo de objeto que se detectó. Para esto se usan los datos entregados por del "Survey ZFT" ([1],[2]), mediante el escaneo de 3750 grados cuadrados cada hora del cielo en la zona norteña que tiene por objetivo explorar el cielo para encontrar día a día Supernovas jóvenes y otros objeto de interés como transientes astronómicos raros. Además la cantidad de datos producida por esta exploración permite tener muestras de otros objetos posibles para estudiar como las estrellas variables y binarias, Núcleos galácticos activos (AGN por sus siglas en inglés) y asteroides.

Objetivos

El objetivo de este proyecto es diseñar una estructura de Deep Learning capaz de utilizar las imágenes de objetos entregados por el "survey ZTF" para poder clasificar Supernovas, eventos que suceden de manera rápida y de los cuales se tienen muy pocos muestreos en comparación con otros objetos astronómicos. La estructura que se probó corresponde a una red neuronal convolucional, características del área de Deep Learning y compuestas por múltiples capas. La idea es ser capaz de ser capaz de realizar una clasificación temprana de los objetos detectados, tarea que permitiría optimizar los tiempos que se utilizan en revisar las imágenes directas de los objetos por un astrónomo.

Teniendo como objetivo principal lograr la mejor clasificación de Supernovas, también se busca lograr una clasificación de los distintos objetos del dataset en general. Se deben probar distintos enfoques para intentar mejorar el desempeño de la arquitectura desarrollada, modificando partes de la estructura que se pueden considerar útiles para mejorar el desempeño y la prueba experimental con distintos parámetros, probar metadatos que describen cualidades de los objetos detectados que se obtienen del "survey" y la aplicación de técnicas para balance de clases.

2. Marco teórico

Para comprender este proyecto se necesitan varios conceptos previos.

2.1. Arquitectura CNN

- Convolutional Layer: El principal componente de una red CNN son las capas convolucionales, las que les dan el nombre de CNN (Convolutional Neural Network). Esta capa se encarga de aplicar un filtro

o kernel a la capa anterior utilizando la operación de convolución. La operación está definifa por la siguiente ecuación:

$$y = \phi(W * x + b)$$

Donde W es un conjunto de filtros que se aplica a la entrada x mediante una operación de convolución denotada por *, y b es un vector de bias para cada filtro. ϕ es la función de activación que se explica más adelante. La principal característica de una CNN es que permite mantener cierto grado de la correlación espacial o temporal de x.

- Max pooling: Las capas de Pooling son usadas en una CNN para reducir la dimensión espacial de la entrada x. Max-pooling en particular lo hace tomando el máximo valor de una ventana de tamaño nxn donde n es el denominado stride.
- Fully Connected Layer: Las capas Fully-Connected son la base de las arquitecturas de redes neuronales. Utilizan unas unidades básicas de neuronas que mediante una entrada generan una relación lineal, seguida por una función de activacón no lineal. Utiliza la misma fórmula vista en la capa convolucional pero en vez de aplicar convolución realiza una multiplicación.
- Activation Function: Función no lineal que se aplica a la salida de una capa de la red. En esta experiencia se utilizó la función ReLU definida por:

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

- Softmax layer: Función de activación que generalmente es aplicada en la capa de salida. Entrega un un valor por clase que representa la probilidad de pertenencia a cada una. Para N neuronas, la función Softmax está definida por:

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

- Zero Padding: Técnica usada para preservar la dimensión de la entrada. Agrega 0s a los espacios que falten o sobren de la entrada según una dimensión definida
- Batch normalization: Durante el entrenamiento se encarga de normalizar los datos y computar una media móvil exponencial del promedio y varianza del set de entrenamiento. Luego del entrenamiento los valoress ajustados durante este son usados para los nuevos datos.
- Dropout: Es una técnica para reducir el sobre-ajuste de una red neuronal. Es definido mediante el parámetros "Dropout rate", que es un valor entre [0,1]. Esta técnica realiza "desconexiones" entre las neuronas en la capa aplicada según el porcentaje del valor del dropout rate. Esto genera que el modelo no depende de neuronas específicas.
- Loss Function: Función a optimizar en el entrenamiento. En esta experiencia se utiliza la función Cross-Entropy definida por la fórmula en 2.

2.2. Entrenamiento de una CNN

2.2.1. Algoritmo

El algoritmo de entrenamiento se basa en resolver la siguiente función de optimización:

$$\theta* = argmin_{\theta \in \Theta} C(\theta) = argmin_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} L(y^{(i)}, f_\theta(x^{(i)}))$$

θ es un parámetro en el conjunto de parámetros del modelo Θ, mientras que $\theta*$ es el parámetro que maximiza la distancia entre el valor deseado y el valor que entrega el modelo. $C(\cdot)$ es un error funcional definido por la función Loss $L(\cdot)$. $y^{(i)}$ es la salida deseada o real a la muestra i, y $x^{(i)}$ es la entrada asociada a la salida, mientras que $f(x^{(i)})$ es la salida del modelo que se desea acercar lo más posible a la salida deseada.

Para optimizar la función se utiliza la técnica del gradiente descendente, mediante la atualización iterativa del parámetro definida como:

$$\theta_i < -\theta_{i-1} - \mu \nabla_\theta C(\theta)$$

Donde θ_{i-1} es el parámetro antiguo, θ_i es el parámetro actualizado, y μ es el llamado learning rate.

Para esta iteración se utiliza el algoritmo back-propagation, que propaga el error desde la salida hasta la entrada, llendo "hacia atrás" en el modelo. Esto lo hace mediante el teorema de regla de la cadena para derivadas.

2.2.2. Optimizador

La iteración que se realiza para actualizar los parámetros puede ser muy costosa para grandes cantidades de datos. Es por esto que se utiliza los denominados batches, que son subconjuntos de los datos de K muestras denominado batch size. Con esto la optimización toma una fórmula no sesgada definida por:

$$\theta_j = \theta_{j-1} - \mu \frac{1}{|K|} \sum_{i=1}^{N} L(y^{(i)}, f_\theta(x^{(i)}))$$

En esta experiencia se utilizó el optimizador "Adam" [3], que utiliza un μ_k variable que va actualizando mediante los gradientes cuadráticos, e incluye el promedio móvil de los gradientes llamado momentum, lo que permite no converger mínimos locales. Utiliza dos parámetros: β_1 que cambia los promedios móviles de los gradientes, y β_2 que cambia los gradientes cuadráticos.

2.3. Literatura

Para realizar la clasificación se utilizó una red neuronal convolucional o también llamada CNN por sus siglas en inglés, que corresponde a un modelo del área de Deep Learning. Este modelo está compuesto por distintas capas o redes donde cada una contiene una cantidad de pesos relacionadas con una unidad o "neurona". Estos pesos tienen conexiones con los pesos de la capa siguiente y anterior, pero a diferencia del modelo de Perceptron Multi-Capa, no tiene todas las neuronas de una capa conectadas con todas las neuronas de la siguiente, lo que permite evitar de forma más eficiente el sobre ajuste a los datos. Mediante el uso de un filtro o kernel, con el cual se realiza una operación de convolución (de ahí su nombre), se pueden extraer características locales del input al generar los llamados feature map, correspondiente las unidades de una capa organizadas en un respectivo plano. Las CNN han demostrado ser muy útiles al trabajar con imágenes, principalmente por la forma en que extraen características locales, muy importantes

en las imágenes 2D debido a la alta correlación de los pixeles cuando están espacialmente o temporalmente cerca. Además mediante los feature map mencionados anteriormente se puede lograr extraer múltiples características de distintas zonas de la imagen. El modelo CNN utilizado se baso en las investigaciones "Enhanced Rotational Invariant Convolutional Neural Network for Supernovae Detection" de E. Reyes et. al. [5], "Alert Classification for the ALeRCE Broker System: The Real-time Stamp Classifer" de R. Carrasco [6] y en menor medida en "DEEP-HITS: ROTATION INVARIANT CONVOLUTIONAL NEURAL NETWORK FOR TRANSIENT DETECTION" [7]. Por otra parte a la hora de entrenar los modelos se hace necesaria la representatividad por clases y dada la problemática que la falta de esta puede generar se necesita utilizar una variedad de técnicas que de forma artificial enmiendan el problema, siendo utilizadas data augmentation que copian la imagen con una pequeña modificación para simular un nuevo ejemplo.

El objetivo principal es superar lo alcanzado por el modelo usado para el informe preliminar, donde se alcanzó un accuracy de test general de 0.89, y para la clase específica de Supernovas se alcanzó un valor de 0.615.

3. Solución Propuesta

Base de Datos

La base de datos utilizada corresponden a un diccionario con 52244 detecciones de objetos, con 3 llaves: las imágenes de los objetos, cada una de 63x63 con 3 canales, con la imagen centrada en el objeto astronómico; la categoría astronómica a la que pertenece cada objeto; y metadatos de cada imagen, que contienen información astronómica de las detecciones. Los canales de las imágenes corresponden a los tres tipos que obtiene el "survey". El primer canal corresponde a la imagen "Science" correspondiente a la última medición que se tuvo de la fuente; el segundo canal corresponde a la imagen "Template" que es la imagen de referencia, generalmente con una alta relación señal/ruido tomada en una época anterior; y el tercer canal corresponde a la imagen "Difference", que es la diferencia entre las imagenes anteriores. Entre el dataset existen 5 clases u objetos astronómicos, asociada cada una a una clase numérica: Active Galactic Nuclei (AGN) con la clase 0; Supernova (SN) con la clase 1; Variable Star (VS) con la clase 2; Asteroid con la clase 3; y Bogus Alerts con la clase 4. La cantidad de cada clase en el dataset se puede observar en la Figura 1, donde se puede ver que la clase SN tiene muy pocos datos en comparación con los demás.

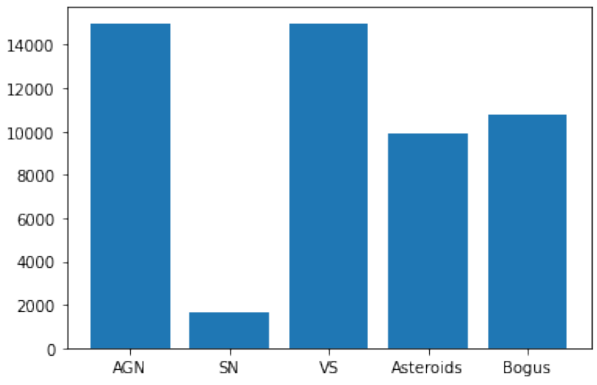

Figura 1: Cantidad de cada clase en el dataset.

Por último, existen 23 features en los metadatos, que muestran características obtenidas del "survey"

para cada detección, las que se pueden observar en la Tabla 1 con el rango de valores que toma cada una. En la Figura 2 se puede ver un ejemplo de metadata con la distribución para la característica $sgscore1$.

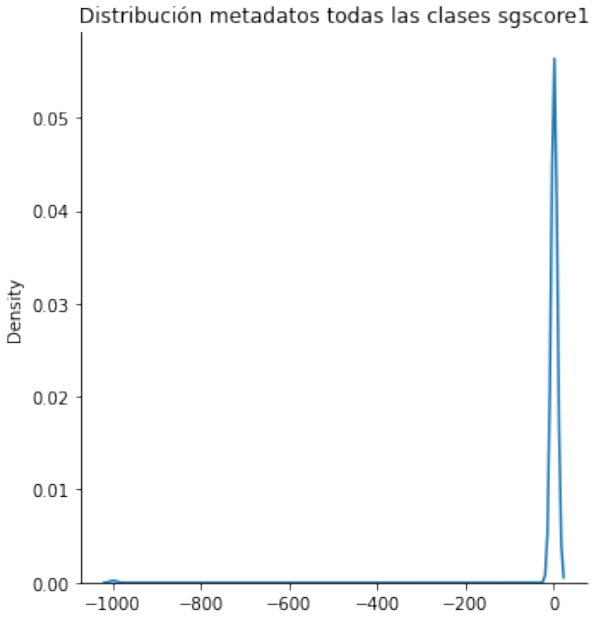

Figura 2: Distribución para la característica $sgscore1$.

Tabla 1: Lista de los metadatos. Definición de esquema ZFT avro [4].

Feature	Valores
$sgscore$ {1,2,3}	Entre [0,1]. Cuando el valor es -999 significa que no hay una fuente.
$distpsnr$ {1,2,3}	Entre [0,30]. Cuando el valor es -999 significa que no hay fuente.
$isdiffpos$	Variable discreta 0,1
$fwhm$	Entre [-0.85,106.440002]
$magpsf$	Entre [12.033670,21.455547]
$sigmapsf$	Entre [0.000685,0.217142]
ra	Entre [0.002221,359.997222]
dec	Entre [-27.985070,87.015108]
$diffmaglim$	Entre [14.138559,21.245272]
$classtar$	Entre [0,1]
$ndethist$	Entre [1, 1436]
$ncovhist$	Entre [1, 6258]
ecl_lat	Entre [-51.284163,89.771758]
ecl_long	Entre [0.000212,359.998352]
gal_lat	Entre [-89.511370,89.520706]
gal_long	Entre [0.000780,359.998670]
$non\_detections$	Entre [-190,6257]
$chinr$	Entre [0, 217.156998]. Cuando el valor es -999 significa que no hay fuente.
$sharpnr$	Entre [-1.291,]2.419000]. Cuando el valor es -999 significa que no hay fuente.

Cambios aplicados

Para las imágenes, se observó que no todas tenían el mismo tamaño, por lo que se decidió realizar un recorte al centro de la imagen. Se buscó la imagen que tenía el mínimo tamaño, encontrándose que era de largo y ancho 41. Además observando distintos ejemplos se decidió que 41 era un buena tamaño para cortar parte del objeto y no perder información que puede ser relevante.

Al momento de visualizar las distribuciones se encontraron muchas "features" que tenían objetos anómalos. Por ejemplo se observa en la Figura 2 que $sgscore1$ tiene muchos valores próximos a 0, pero tiene una cantidad mínima de valores en -999. Esto se debe a extracción de características del mismo observatorio y "survey" realizado, siendo posibles errores de la medición, por lo que no se entrará en mayores detalles. Debido a estos valores anómalos, se decidió establecer cotas superiores o inferiores, según el caso respectivo, para cambiar todos los valores que estuvieran fuera de la cota al valor de ella, es decir se asigna un valor máximo y/o mínimo dependidendo del caso. Por ejemplo para el caso del ejemplo de $sgscore1$ se cambiaron todos los valores anómalos por -1. En la Tabla 2 se aprecian los valores usados para definir los máximos y mínimo de las variables con datos anómalos.

Tabla 2: Límites utilizados para metadatos cambiados. min_value y max_value hacen referencia al valor mínimo y máximo utilizados respectivamente. Para None es que no se utilizó valores.

Feature	[min_value,max_value]
$sgscore$ {1,2,3}	[-1,None]
$distpsnr$ {1,2,3}	[-1,None]
$fwhm$	[None, 25]
$ndethist$	[None, 200]
$ncovhist$	[None, 2100]]
ecl_lat	Entre [-51.284163,89.771758]
ecl_long	Entre [0.000212,359.998352]
gal_lat	Entre [-89.511370,89.520706]
gal_long	Entre [0.000780,359.998670]
$non\_detections$	[None, 2100]
$chinr$	[-1, None]
$sharpnr$	[-1,None]

Luego se procedió a realizar una normalización estándar, definida por ecuación 1, donde se resta cada valor por su media $\bar{X}$ y se divide por su desviación estándar σ, para cada feature de los metadatos y para cada imagen en cada canal, con lo cual tanto imágenes como metadatos fueron normalizados por la ecuación:

$$\hat{X} = \frac{X - \bar{X}}{\sigma} \tag{1}$$

Cabe mencionar que la no normalización de datos resultaba en objetos de características o pixeles demasiado grandes en comparación al resto y los pesos del clasificador solo terminan distinguiendo una clase, por lo que existe sobreajuste a la hora de ignorar el proceso de estandarización.

3.1. Modelos implementados

Para realizar la clasificación se utilizó una red convolucional, también conocida como CNN. Se utilizó como base el modelo de la investigación de "Alert Classification for the ALeRCE Broker System" [6]. Este modelo se le ingresan las imágenes y los metadatos, donde lo primero pasa por una capa de zero padding y los metadatos pasan por una capa de batch normalization. Luego las imágenes (cada una de las 3 stamps) pasan por un cyclic pooling donde se genera 3 rotaciones de la imagen de 90°, 180° y 270°, que más la imagen original pasan por una red convolucional que se puede apreciar en la Figura 3 en la parte izquierda, compuesta por 5 capas convoluciones 2D, 1 de tamaño 32 con kernel 4 y otra del mismo tamaño con kernel 3, 2 de tamaño 64 con kernel 3 y otra del mismo tamaño con kernel 4; 2 capas de maxpooling, una después de las primeras capas convolucionales de tamaño 32 y la otra después de las 3 capas convolucionales de tamaño 64. Luego cada descriptor se "aplanan" mediante una capa "flatten" y se pasa por una capa fully connected de tamaño 64 lo que genera 4 vectores para cada imagen rotada y se toma el promedio en cada posición. Esto se realiza por que ha demostrado que así puede extraer características importantes del espacio de latencia de la imagen.

Luego el vector obtenido se pasa por una capa de batch normalization y se concatena con el vector obtenido del cyclic pooling. Luego se pasan por dos capas Fully-Connected de tamaño 64, para finalmente llegar a la capa de salida de 5 neuronas, también mediante una Fully-Connected. Esta red con todos los componentes genera un modelo con 215,841 parámetros en total.

Todas las capas convolucionales y la primera Fully-Connected usan como función ReLu. La última capa Fully-Connected utiliza la función Softmax para la salida. En la Figura 3 se puede ver una visualización del modelo completo, en la parte derecha con el proceso de cyclic pooling y posterior concatenación con los metadatos, y en la izquierda se ve la capa convolucional por la que pasa cada imagen rotada.

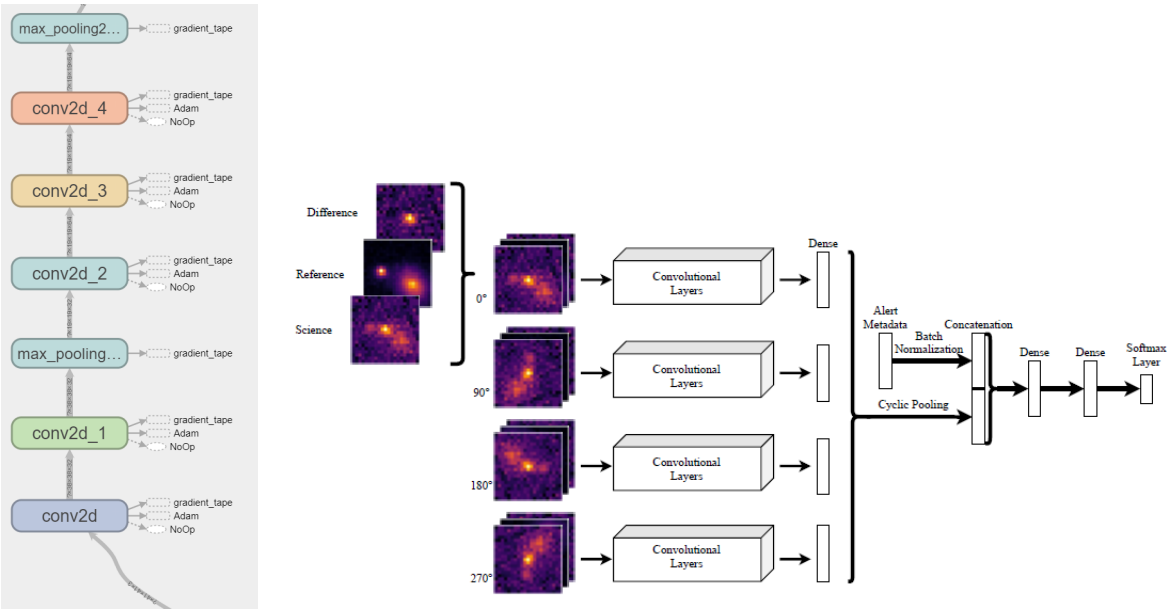

Figura 3: Diagrama de la arquitectura del modelo base. A la izquierda se ve la primera parte del modelo con las capas convolucionales y max-pooling. A la derecha se ve el proceso de cyclic pooling y la segunda parte del modelo, donde se utilizan las capas Fully-Connected.

Debido a que el problema propuesto es de clasificación, según características o imágenes, la salida deseada en ambos casos es la correcta clasificación del evento astronómico, donde el objetivo con mayor prioridad es la correcta predicción de SuperNovas, razón por la cual se debe disminuir la función de error, por lo que se usa la función de costo "categorical loss entropy", definida por la fórmula 2 con y_i el valor real de la clase y $\hat{y}_i$ el valor predicho por el modelo.

$$\mathcal{L} = -\sum_{i=1}^{N} y_i * \log \hat{y}_i \tag{2}$$

Se probó además agregando a esta arquitectura dos capas Fully-Connected a la salida del cyclic pooling, de tamaño 128 y 64. En la Figura 4 se observa como se agregan dos Fully-Connected después del cyclic pooling.

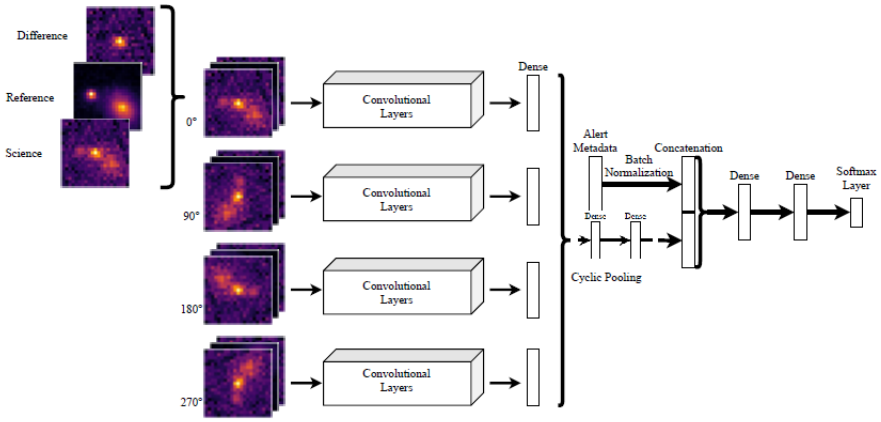

Figura 4: Diagrama de la arquitectura del modelo con dos capas Fully-Connected a la salida del cyclic pooling.

Para el problema del dataset desbalanceado se realizaron experimentos utilizando data augmentation, técnica que aumenta el volumen de muestras del dataset. Se probaron dos enfoques o formas de obtener las nuevas muestras. La primera consiste en realizar una traslación a cada imagen de la clase supernova, moviendo un pixel a la derecha, otro a la izquierda, otro abajo y otro arriba, con lo que se genera 4 imagenes nuevas desde una. Considerando que se utilizó un conjunto de train que tenía 972 clases de supernova, se obtendrían 3888 imagenes nuevas en el dataset. Para la sección que se le quita (fila o columna que queda en 0 por el desplazamiento) los pixeles se utilizó el parámetro "nearest" de la función "tfa.image.translate" de la librería tensoflow que realiza una traslación, y que rellena esos espacios con los pixeles más cercanos en su vecindad. En la Figura 5 se ve un ejemplo de las 4 traslaciones que se obtienen con la imagen original. Se puede apreciar que en las líneas correspondientes donde se realiza la traslación está más borroso.

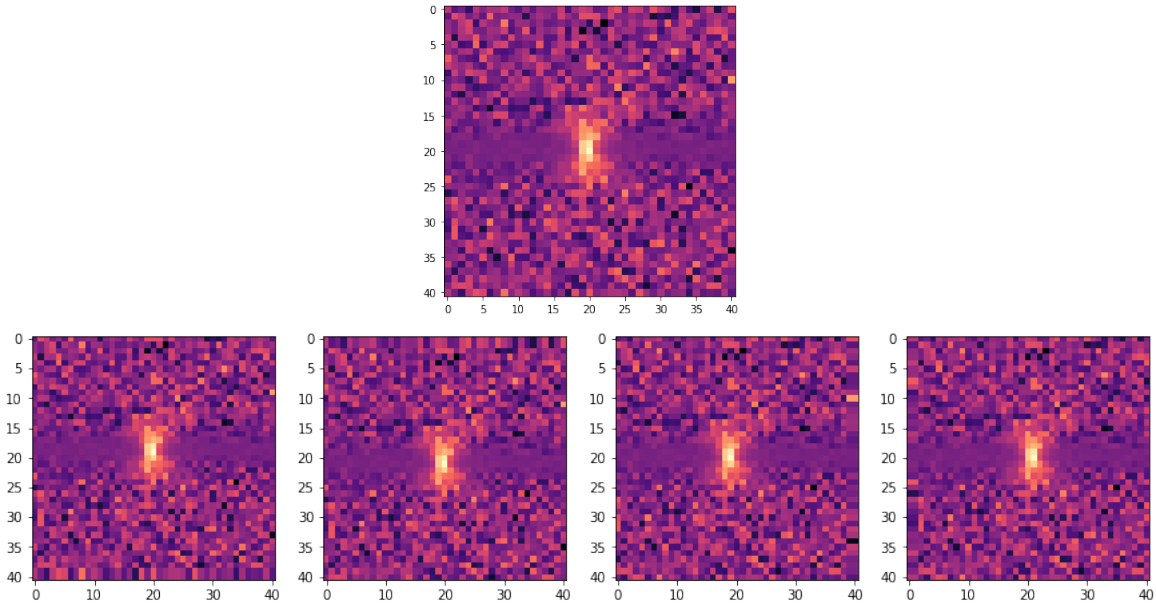

Figura 5: Ejemplo de imagenes que se obtiene de primer enfoque de Data Augmentation mediante una traslación de un pixel en 4 direcciones. Arriba imagen original. Abajo imágenes trasladadas en cada dirección.

El segundo enfoque que se probó se basó en utilizar las capas de tensorflow ”layers.RandomRotation()” y ”tf.keras.layers.RandomTranslation()”. La primera capa aplica una rotación en torno a un rango determinado que se estableció en un $[-80\,\%2*\pi, 80\,\%2*\pi]$, y la segunda capa aplica una traslación en un rango establecido que se dejó en $6\,\%$, lo que significa que para una imagen de 24x24 pixeles, se podría trasladar en un rango de 1.44 pixeles. Para la sección trasladada se aplica el mismo parámetro de ”nearest” para la sección que queda vacía. En la Figura 6 se observa un ejemplo del data augmentation aplicado a los 3 stamps de un objeto de clase supernova. Para la segunda fila se muestra un ejemplo de imágenes rotadas y trasladadas con los parámetros que se mencionaron antes y se que se usaron con los modelos. En la tercera fila se muestra un ejemplo con un rango de traslación de $60\,\%$ para mostrar un efecto más concreto de como se aplica el data augmentation.

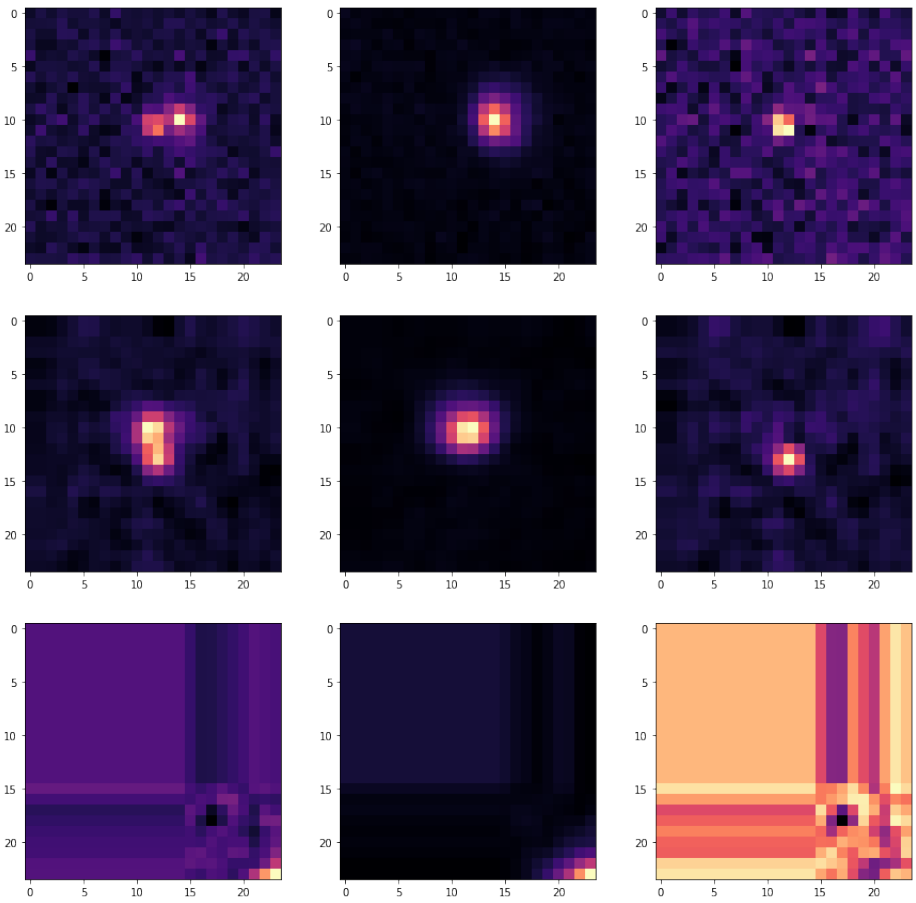

Figura 6: Ejemplo de imagenes que se obtiene de segundo enfoque de Data Augmentation mediante una rotación y traslación aleatoria. Arriba imagen original. Medio imágenes trasladadas en un rango 6 %. Abajo imagen trasladada en un rango de 60 %.

Para el entrenamiento de todos los modelos se particionó los datos en 3 conjuntos: Entrenamiento, Validación y Prueba. La proporción otorgado a cada uno corresponde a un 20 % prueba, y del 80 % se obtuvo un 25 % para el conjunto de validación, es decir un 20 % del total quedó para el conjunto de validación, y finalmente para el conjunto de entrenamiento se dejó el 60 % del total de datos. Para realizar la partición se utilizó la semilla 4016 utilizando la función de sklearn "train$_test_split()$".

Se probaron distintos hiper-parámetros para ver cual mejoraba los modelos. Se uso como base el modelo inicial para ver cual conseguía un mejor rendimiento.

- Se utilizó el principio de optimización 'Adam' [8], con los parámetros $learning\_rate = 0.0001, beta\_1 = 0.5, beta\_2 = 0.9.$, para el aprendizaje de la red. Se eligió el learning rate después de probar con los valores: 1e5, 0.0001, 0.001, 0.01, 0.1, 1. El valor 0.001 fue el que mejor obtuvo resultados para el modelo base

- Para ambas arquitecturas se utilizan 50 épocas, las cuales están sujetas a una detención temprana, determinada por la disminución durante 5 épocas seguidas de la métrica accuracy para el conjunto de

validación, evitando así un sobre-ajuste excesivo en las redes.

- El ajuste de ambas redes consta de un parámetro de *dropout* al cual se ajusta con valor 0.5 antes de utilizar las redes Fully-Connected.
- Se probaron dos tamaños del batch: 32 y 64.
- También se probaron distintos recortes de las imágenes, como se explicó en la sección anterior. Se probaron con distintos valores pero se muestran dos valores para destacar: 24 y 41.

Toda la implementación, desde el preprocesamiento hasta la construcción, entrenamiento y evaluación de desempeño de las redes es realizado en *colab*, con Python 3, utilizando las librerías de Tensorflow, Keras, Matplotlib, Time, Sklearn, Seaborn, Pandas, Pickle y Numpy. El código se encuentra en el repositorio [9]. La programación de preprocesamiento posee 2 partes; metadata, en la cual se definen valores máximos y mínimos (se "clipean" los datos) y se escalan mediante un StandardScaler, por otra parte la programación del preprocesamiento de imágenes consta del recorte de todas a un valor definido, la modificación de valores NaN por 0, y normalización en cada canal para todas las imagenes, todo mediante funciones de Numpy. Luego se programan las redes y su entrenamiento mediante Tensorflow y Keras para posteriormente ser evaluadas bajo las diferentes métricas.

Se probaron distintos experimentos que se proceden a explicar. Cada experimento se nombra con un número.

3.1.1. Modelo 1

Para el primer experimento se probó el modelo base de la Figura 3, sin utilizar ninguna técnica de Data Augmentation. Para este experimento se utilizaron los hiperparámetros comentados anteriormente, con un tamaño de batch size de 32. Este experimento fue para probar la influencia del tamaño del recorte de las imágenes y como influyen. Para este modelo denotado por modelo 1 se probó con un recorte del tamaño de imagen a 41x41 pixeles.

3.1.2. Modelo 2

Para el segundo experimento se utilizo los mismos parámetros del modelo 1, pero la diferencia es que se probó con un recorte de imágenes de 24x24.

3.1.3. Modelo 3

Para el tercer experimento, se tomó como base el modelo 2 ya que entregó mejores resultados. Pero para esta ocasión se probó con la primera técnica de Data Augmentation que fue comentada anteriormente, la cual traslada cada imagen en una dirección.

3.1.4. Modelo 4

Utilizando nuevamente como base el modelo 2, se probó con la otra técnica de Data Augmentation, que consiste en rotaciones y traslaciones aleatorias de las imágenes. Se copiaron 10 imágenes de la clase 0 (AGN), 5500 de la clase 1 (SN), 10 de la clase 2 (VS), 1000 de la clase 3 (Asteroids) y 1000 de la clase 4

(Bogus). Además, la útlima capa Fully-Connected de la red convolucional (izquierda Figura 3), se cambió por un tamaño de 128.

3.1.5. Modelo 5

Tomando como base el modelo 4, que entregó mejores resultados con data Augmentation, se utilizó como base agregando para el quinto experimento donde se agregaron dos capas después del cyclic pooling, como se observa en la Figura 4.

3.1.6. Modelo 6

Para el sexto experimento se utilizó como base el modelo 4, donde se cambió el número de imágenes nuevas. En esta ocasión se copiaron 1 imagen de la clase 0 (AGN), 6100 de la clase 1 (SN), 1 de la clase 2 (VS), 1000 de la clase 3 (Asteroids) y 600 de la clase 4 (Bogus).

4. Resultados y Análisis

Posterior al entrenamiento de las configuraciones anteriormente diseñadas se entra en la evaluación de desempeño de ambas configuraciones, bajo diversas métricas.

En la Tabla 3 se encuentran los valores de accuracy que obtuvieron los distintos experimentos para cada conjunto del dataset.

Tabla 3: Valores Accuracy de cada conjunto para los modelos probados.

	Train	Validation	Test
Modelo 1	0.934	0.932	0.934
Modelo 2	0.928	0.935	0.940
Modelo 3	0.949	0.938	0.940
Modelo 4	0.949	0.941	0.944
Modelo 5	0.955	0.938	0.940
Modelo 6	0.948	0.936	0.938

En la Figura 7 se observan las matrices de confusión en el mismo orden numérico.

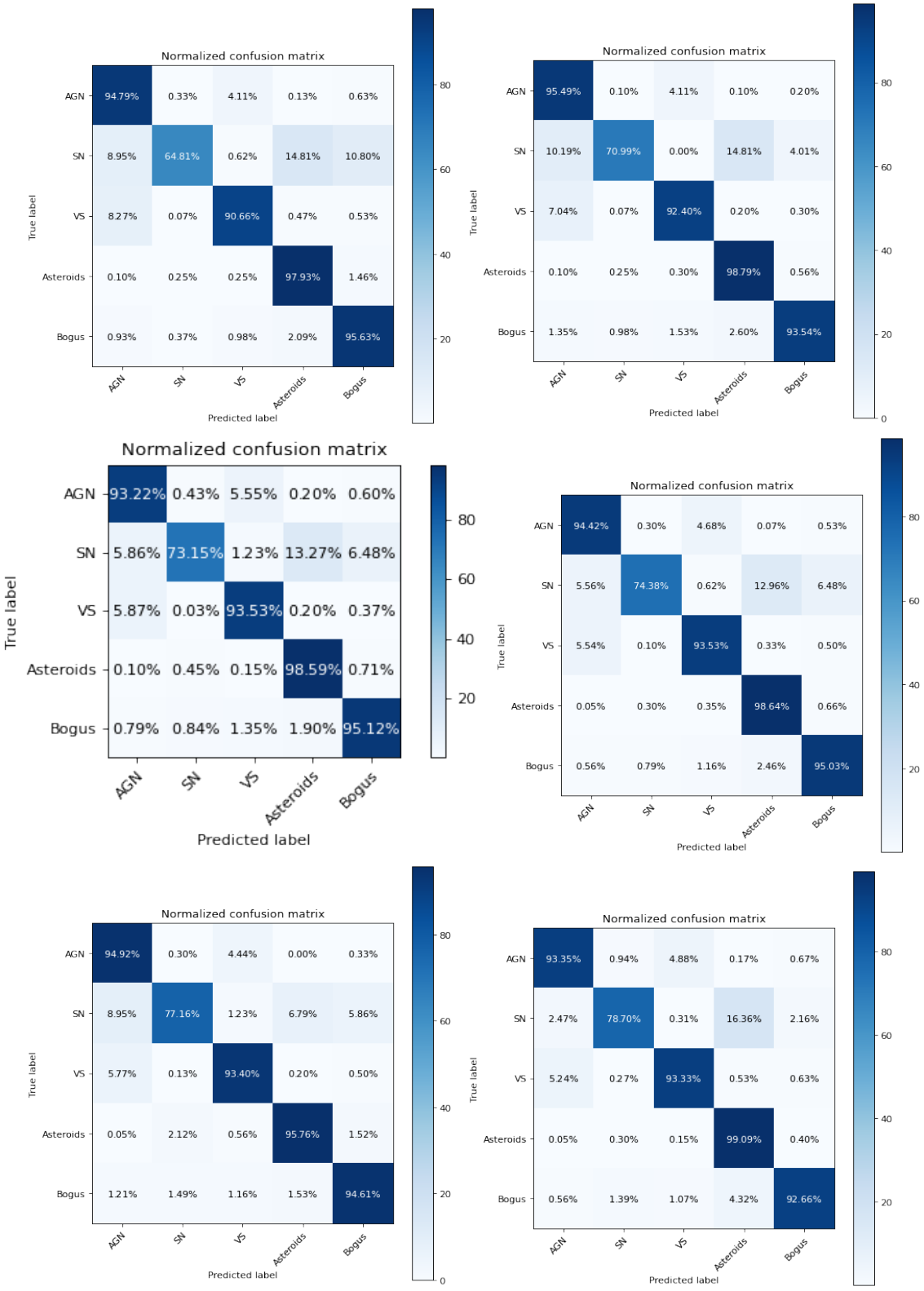

Figura 7: Matrices de confusión para cada modelo. De izquierda a derecha desde arriba hacia abajo: M1, M2, M3, M4, M5, M6

Como primera apreciación es posible discernir que el desempeño de cada modelo en comparación al primero es mejor, lo que advierte que utilizando menor cantidad de pixeles para su entrenamiento se obtiene una mejor performance, lo que sugiere que la información a estudiar es concentrada en el centro de estas.

Por otra parte la utilización de técnicas para balancear clases mejoran el desempeño de la red y precisamente clasificando de mejor manera el evento astronómico objetivo, el cual antes de las técnicas poseía una representatividad aún menor, con lo cual se verifica de forma experimental la mejora de desempeño para clasificadores usando Data Augmentation.

En última instancia puede apreciar que el mejor modelo general es el 4 bajo la métrica de accuracy. Pero el objetivo principal es clasificar lo mejor posible supernova, por lo cual se debe evaluar el desempeño bajo métricas por clase, utilizando así la matriz de confusión para este fin, las cuales al ser revisadas se observan mejores resultados considerando el trade-off entre la clase objetivo y desempeño general con modelo 5. Para comparar más específicamente, se obtuvieron distintas métricas por clase para este modelo. En la Tabla 4 se encuentran los valores de Accuracy, Precision, Recall y Fscore para cada clase obtenidas para el modelo 4. En la Tabla 5 se encuentran los mismos valores obtenidos para el modelo 5.

Tabla 4: Valores de distintas métricas para conjunto de test en modelo 4.

	AGN	SN	VS	Asteroids	Bogus
Accuracy	0.944	0.743	0.935	0.986	0.950
Precision	0.934	0.873	0.941	0.948	0.969
Recall	0.944	0.743	0.935	0.986	0.950
fscore	0.939	0.803	0.938	0.966	0.959

Tabla 5: Valores de distintas métricas para conjunto de test en modelo 5.

	AGN	SN	VS	Asteroids	Bogus
Accuracy	0.949	0.771	0.933	0.957	0.946
Precision	0.925	0.741	0.941	0.968	0.964
Recall	0.949	0.771	0.933	0.957	0.946
Fscore	0.937	0.756	0.937	0.963	0.955

Las curvas de Loss y Accuracy para el modelo 4 se encuentran en la Figura 8, mientras que las mismas curvas para el modelo 5 se encuentran en Figura 9.

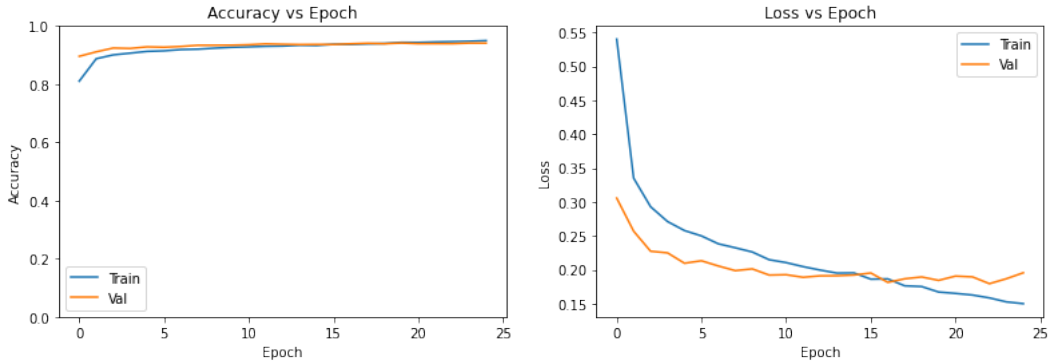

Figura 8: Accuracy y loss vs épocas para Modelo 4.

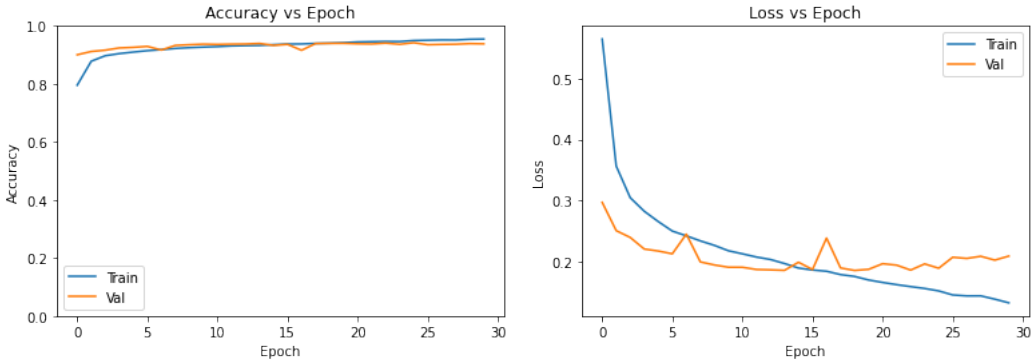

Figura 9: Accuracy y loss vs épocas para Modelo 5.

Conclusiones

El objetivo principal obtiene un progreso considerable, mejorando los resultados de clasificación para Super Novas, lo que valida las hipótesis planteadas que sugieren explorar diferentes enfoques para mejorar desempeño al igual que la examinación de distintos parámetros de forma experimental.
La correcta aplicación de preprocesamiento de datos es fundamental para lograr clasifiaciones, ya que sin la aplicación de este, las features que poseen mayores órdenes de magnitud tendrán mayor peso, lo que se traduce en una clasificación erronea y con alta probabildad de sesgo.
La utilización de metadatos como recurso complementario para entregar información acerca de las imágenes contribuye al mejor rendimiento a la hora de clasificar.
Mediante las técnicas de balanceo de clases utilizadas, siendo ambas diferentes enfoques de Data Augmentation, se aprecian mejoras en el desempeño de la red, pero al ser el problema de proporción solo con una clase de baja representatividad, la cual es la clase objetivo, el desempeño al aplicar estas técnicas ve mejorías apreciables en esta.
A la hora de trabajar con clasificación mediante redes gran parte del tiempo se otorga a los entrenamientos, los que dependen de la capacidad computacional y mejoramiento de algoritmos, al utilizar librerías especializadas el mayor contratiempo viene por limitaciones de hardware tanto en procesamiento de información como memoria a utilizar, lo que evidencia la importancia de tener recursos a la hora de entrenar.
El desempeño del clasificador puede mejorar utilizando una base de datos similar que aporte mayor data de

Super Novas o consiguiendo mayor poder de cómputo." Dado todo lo anterior es posible vislumbrar que la implementación de estos clasificadores mediante ensayo y error aportan un mejor entendimiento a la hora del desarrollo, estudio y análisis de redes.

Referencias

[1] Bellm, Eric (2013) The Zwicky Transient Facility. In: Proceedings of the Third Hot-Wiring Transient Universe Workshop. Los Alamos National Laboratory , Los Alamos, NM, pp. 27-33. Avalaible in https://authors.library.caltech.edu/95143/

[2] Smith, Roger M. and Dekany, Richard G. and Bebek, Christopher and Bellm, Eric and Bui, Khanh and Cromer, John and Gardner, Paul and Hoff, Matthew and Kaye, Stephen and Kulkarni, Shrinivas and Lambert, Andrew and Levi, Michael and Reiley, Dan (2014) The Zwicky Transient Facility Observing System. In: Ground-based and Airborne Instrumentation for Astronomy V. Proceedings of SPIE. No.9147. Society of Photo-Optical Instrumentation Engineers , Bellingham, WA, Art. No. 914779. ISBN 978-0-8194-9615-7. Avalaible in https://authors.library.caltech.edu/58413/

[3] Kingma, D. P., Ba, J. 2017, arXiv:1412.6980 [cs]. http://arxiv.org/abs/1412.6980

[4] ZFT avro scheme. Avalaible in https://zwickytransientfacility.github.io/ztf-avro-alert/

[5] Reyes, E., Estévez, P. A., Reyes, I., Cabrera-Vives, G., Huijse, P., Carrasco, R., Forster, F. (2018, July). Enhanced Rotational Invariant Convolutional Neural Network for Supernovae Detection. In 2018 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

[6] Carrasco-Davis, Rodrigo, et al. .ªlert Classification for the ALeRCE Broker System: The Real-time Stamp Classifier..ªrXiv preprint arXiv:2008.03309 (2020).

[7] Cabrera-Vives, et al. "DEEP-HITS: ROTATION INVARIANT CONVOLUTIONAL NEURAL NETWORK FOR TRANSIENT DETECTION." arXiv preprint arXiv:1701.00458 (2017) https://arxiv.org/abs/1701.00458

[8] Kingma, D. P., Ba, J. 2017, arXiv:1412.6980 [cs]. http://arxiv.org/abs/1412.6980

[9] https://github.com/jespejom/EL4106IC$_{P}royecto7b.git$