Universidad Austral de Chile

Facultad de Ciencias de la Ingeniería

Escuela de Ingeniería Civil en Informática

Capas de adaptación para la clasificación de Curvas de Luz usando Redes Neuronales Artificiales

Proyecto para optar al título de
Ingeniero Civil en Informática

PROFESOR PATROCINANTE:
PABLO HUIJSE HEISE
DOCTOR EN INGENIERÍA ELÉCTRICA

PROFESOR CO-PATROCINANTE:
RODRIGO CONTRERAS RAMOS
DOCTOR EN ASTRONOMÍA

PROFESOR INFORMANTE:
MATTHIEU VERNIER
DOCTOR EN INFORMÁTICA

ALFREDO ELÍAS MORALES HERRERA

VALDIVIA - CHILE
2020

AGRADECIMIENTOS

Quiero agradecer en primer lugar a mi familia por darme todo lo necesario y más. Por apoyarme incondicionalmente a lo largo de este camino. En segundo lugar a los profesores que tuve, en especial a aquellos que me brindaron enseñanzas más allá de lo académico, en especial a mi profesor guía por las oportunidades y la paciencia brindadas en este tiempo. Finalmente, pero no menos importante, agradecer a mis amigas y amigos por el apoyo, consejos, contención y gratos momentos vividos.

Gracias a todo aquel que esté leyendo el presente trabajo.

ÍNDICE

ÍNDICE DE TABLAS

TABLA PÁGINA

ÍNDICE DE FIGURAS

FIGURA PÁGINA

RESUMEN

El análisis de curvas de luz, series de tiempo de brillo estelar, es fundamental para el estudio de estrellas variable. Un tipo particularmente interesante de estrella variable son las RR Lyrae (RRL), cuyo brillo y periodicidad puede relacionarse con su distancia a la Tierra, lo cual permite a los astrónomos construir mapas de la Vía Láctea. Por otro lado las redes neuronales profundas representan actualmente el estado del arte en reconocimiento y clasificación de patrones en datos no estructurados. En este trabajo se propone utilizar un modelo basado en redes neuronales convolucionales para resolver un problema de clasificación de curvas de luz de estrellas de tipo RRL del proyecto *Vista Variables in the Via Lactea* (VVV). Esto presenta varios desafíos puesto que las curvas de luz son irregulares en su muestreo y variables en las características del ruido que las afecta, lo que quiebra los supuestos básicos de los modelos de redes profundas y entorpece su aplicación directa.

Para solventar estos desafíos se propone e implementa un modelo de red neuronal profunda basado en capas convolucionales parciales. Las capas convolucionales parciales fueron diseñadas para datos incompletos y en este caso ponderan de forma adaptativa la salida para ajustar la fracción de los datos faltantes e irregulares de curvas de luz. El modelo propuesto se compara contra estrategias referenciales basadas en redes convolucionales convencionales y un ensamble de tipo Random Forest entrenado sobre un conjunto de atributos ampliamente usados en astronomía. El modelo propuesto obtiene un resultado de clasificación que es 10 % superior en términos de puntaje F1 con respecto al mejor modelo referencial. Para el caso particular de las RR Lyrae se obtiene un incremento en la tasa de recuperación de un 2 % y una disminución del 77 % de la contaminación por falsos positivos. Se reconoce la existencia de complementariedad entre los modelos basados en aprendizaje profundo y los basados en atributos que se explorará en detalle en trabajos futuros.

ABSTRACT

The analysis of light curves, time series of stellar brightness, is fundamental towards the study of variable stars. A particularly interesting type of variable star are RR Lyrae (RRL). The brightness and periodicity of these stars can be related with its distance to Earth, allowing astronomers to build maps of the Milky Way. On the other hand, deep neural networks represent nowadays the state of the art in pattern recognition and classification on unstructured data. This work proposes the use of deep convolutional neural networks to solve the classification of RR Lyrae light curves from the *Vista Variables in the Via Lactea* (VVV) survey. Light curves are irregularly sampled and have heteroscedastic noise. These features break the basic assumptions of the model and hinder the direct application of convolutional neural networks.

To overcome these challenges we propose and implement a deep neural network model based on partial convolutional layers. These layers were designed for incomplete data and in this particular case they adaptively weight the output to adjust to the fraction of missing and irregular samples from the light curve. The proposed model is compared with baseline strategies based on conventional convolutional networks and a random forest ensemble that was trained on features broadly used in astronomy. The proposed model obtains a 10 % increase in the F1-score with respect to its best competitor on the VVV test set. For the particular case of RR Lyrae, the proposed models achieves a 2 % higher recovery rate with a 77 % decrease in contamination due to false positives. We recognize that there is complementarity between the deep model and attribute based model that will be explored as future work.

1. INTRODUCCIÓN

1.1. Contexto

El Norte de Chile se encuentra entre los mejores lugares del mundo para realizar observaciones astronómicas. Por esta razón se espera que durante la siguiente década más del 70 % de la capacidad de observación astronómica mundial estará instalada en Chile[1]. Los proyectos de sondaje astronómico actuales y por venir buscan mapear el cielo nocturno en busca de objetos o cuerpos celestes interesantes. Las imágenes digitales capturadas por estos instrumentos son procesadas usando técnicas de fotometría (Warner et al. 2006), resultando en extensos catálogos de "curvas de luz". Una curva de luz es una serie de tiempo del brillo aparente de una estrella en particular. El tiempo suele medirse en días Julianos mientras que el brillo aparente se mide en magnitudes, una escala logarítmica y relativa. El análisis de curvas de luz le permite al astrónomo identificar a qué clase de estrella corresponde y también estimar sus parámetros fundamentales tales como su masa, composición e incluso su distancia real hasta nosotros, entre otras tareas.

El análisis de curvas de luz es fundamental para el estudio de estrellas variables (Percy, 2007), objetos cuyo brillo percibido desde la Tierra varía considerablemente ya sea por razones intrínsecas, es decir asociadas a la física de la estrella, o por razones extrínsecas, como por ejemplo oclusiones periódicas o eclipses causados por otra estrella o planeta. Un tipo particularmente interesante de estrella variable intrínseca son las RR Lyrae (RRL) (Catelan y Smith, 2015). Estas estrellas son muy antiguas, con edades comparables a la Vía Láctea. Otra característica que las distingue de otros objetos astronómicos es que estas estrellas pulsan de forma regular, es decir que el tiempo entre mínimos y máximos sucesivos se mantiene constante. En el caso de las RRL este tiempo, denominado período de pulsación, se encuentran entre los 0,3 y 1 día. La Figura 1 muestra un ejemplo de curva de luz de una estrella RR Lyrae con un período fundamental de 0,557 días. La subfigura izquierda corresponde a la curva de luz original. La subfigura derecha es la curva de luz luego del proceso conocido como *epoch folding*, donde el eje de tiempo se transforma usando

$$\phi_i = \text{modulo}(t_i, P)/P,\ i = 1, 2, \ldots, N \tag{1}$$

donde $\{t_i\}$ son los tiempos de observación y P es el periodo de pulsación. El resultado de esta transformación se conoce como diagrama de fase, y como se muestra en la figura es muy útil para revelar el comportamiento periódico de la estrella, siempre que se conozca el valor correcto de P.

[1] https://mma.gob.cl/wp-content/uploads/2018/06/Cielos_2018_Chilean_Skies.pdf

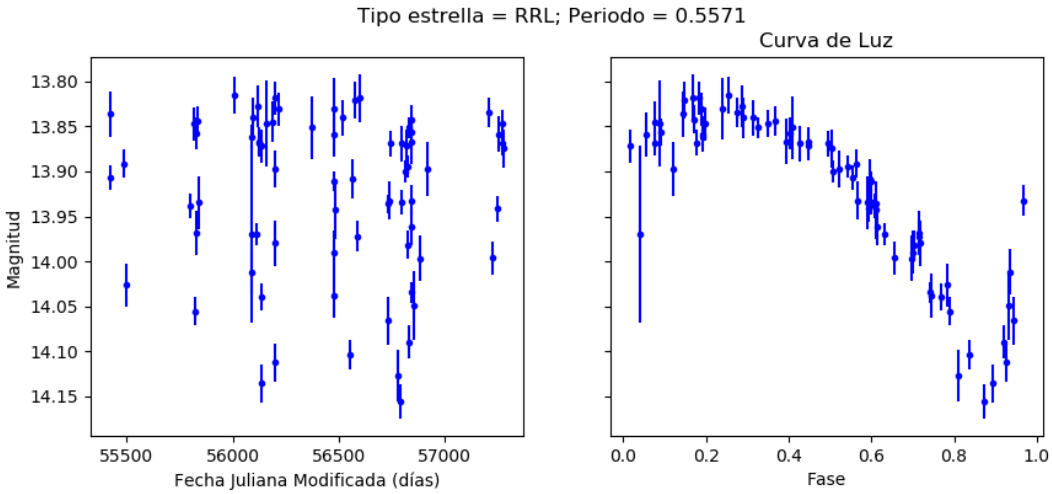


Figura 1. Ejemplo de curva de Luz de estrella RR Lyrae de VVV. La subfigura derecha muestra la curva de luz doblada con su período.

La periodicidad de las RRL está relacionada con sus mecanismos físicos internos. Previamente se han descubierto reglas que relacionan el período de una RRL con su brillo o magnitud absoluta. Esto se conoce como relación período-luminosidad y se muestra de forma esquemática para el caso óptico en la Figura 2. Las RRL siguen una relación período luminosidad estricta cuando son observadas en el rango infrarrojo cercano (Catelan, Pritzl et al. 2004). Usando la magnitud aparente que se estima a partir de las imágenes astronómicas y la magnitud absoluta que es calculada a partir del periodo, es posible deducir la distancia entre una RRL en particular y la Tierra. Esto las convierte en un instrumento fundamental para la medición de distancias en la escala de nuestro vecindario galáctico y en particular para analizar la estructura topológica de la Vía Láctea (Saito et al. 2011). Sin embargo, se requiere detectar una gran cantidad de estos objetos estelares para que los mapas sean confiables (Skowron et al. 2019)

El sondeo VISTA Variables in the Vía Láctea (VVV) (Minniti et al. 2010) es un proyecto de sondaje público financiado por la Unión Europea que observa continuamente nuestra galaxia en el espectro infrarrojo cercano (banda K) usando el instrumento VISTA, localizado en el observatorio Paranal, Región de Antofagasta, Chile. El proyecto VVV busca explicar como se formó la Vía Láctea por medio del análisis de su estructura actual e histórica. Para revelar esta estructura es fundamental la detección de un gran número de estrellas RR Lyrae que habiten en las distintas estructuras de nuestra galaxia. En el presente trabajo se busca resolver el problema de detección automática de RRL a partir de datos de VVV asociados con el bulbo de la Vía Láctea.

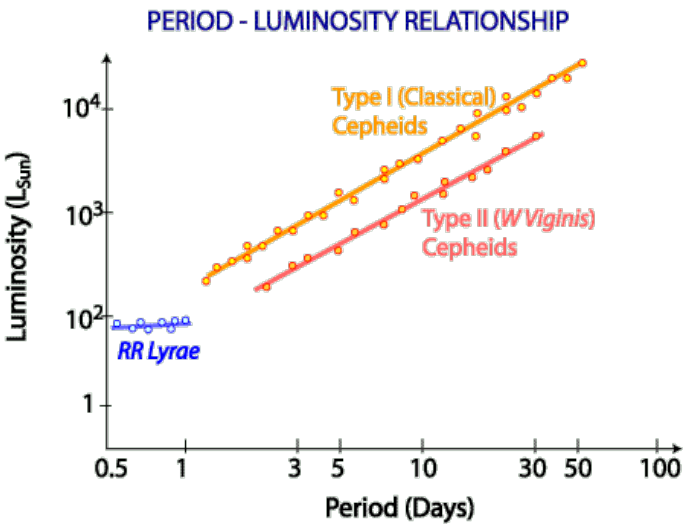


Figura 2. Diagrama esquemático de la relación Período-Luminosidad en el óptico de las RR Lyrae y otras estrellas variables pulsantes

1.2. Desafíos y estado del arte

Sondeos astronómicos modernos como VVV producen datos a una tasa que vuelve infactible el análisis manual de la información por parte de los astrónomos. Se reconoce entonces una necesidad por métodos y algoritmos automáticos que apoyen al astrónomo en las tareas de análisis y clasificación de datos (Borne, 2010; Feigelson, 2016; Huijse et al. 2014). Para enfrentar estos desafíos es necesaria la colaboración y el trabajo interdisciplinar entre astrónomos, matemáticos, científicos de datos e ingenieros. En este contexto un campo de las ciencias de la computación que ha recibido mucha atención es Machine Learning (ML) (Bishop, 2006; MacKay, 2002), el cual se concentra en el aprendizaje de modelos a partir de datos.

La metodología más extendida para la clasificación de estrellas variables se basa en el cálculo de atributos o características a partir de las curvas de luz seguido de la aplicación de un modelo de aprendizaje supervisado (Hinners et al. 2018; Martínez-Palomera et al. 2018; Narayan et al. 2018). Los modelos de tipo ensamble, como por ejemplo Random Forest (RF) (Breiman, 2001), han sido particularmente efectivos en este tarea (Richards et al. 2011). Entre los atributos típicamente usados para caracterizar las curvas de luz se encuentran los momentos estadísticos, el período o frecuencia fundamental y ajustes de modelos paramétricos de tipo autoregresivo o de serie de Fourier (Nun et al. 2015; Richards et al. 2011). Otros trabajos recientes han usado RF para clasificar estrellas variables y calcular sus distancias al centro de la galaxia obteniendo resultados coherentes con la literatura (Quezada, 2020)

Hoy en día el estado del arte en clasificación supervisada para series de tiempo en general consiste de utilizar métodos que extraigan atributos de forma automática a partir de grandes volúmenes de datos, como por ejemplo las redes neuronales artificiales profundas (Goodfellow et al. 2016). Una ventaja fundamental de este paradigma es que al no necesitar atributos diseñados *a priori* se reduce la posibilidad de añadir sesgos al modelo. Una desventaja de este paradigma es que requieren una mayor cantidad de datos para ser entrenados adecuadamente. Los métodos de aprendizaje profundo están empezando a ser empleados para clasificación de curvas de luz obteniendo resultados comparables pero en ningún caso superiores a los métodos tradicionales (Aguirre et al. 2019; Becker et al. 2020).

Los estudios muestran que los métodos de aprendizaje profundo no se han desempeñado como lo esperado debido a que han sido aplicados sin tener en consideración las características fundamentales que diferencian a las curvas de luz de otros tipos de series de tiempo:

- Las curvas de luz están muestreadas irregularmente, las observaciones se realizan en tiempos distintos cada noche. La cantidad de observaciones es distinta para cada objeto astronómico. Interpolar a una grilla regular no es posible debido a que los fenómenos de interés tienen una escala de tiempo menor que el tiempo promedio entre observaciones.
- El ruido asociado a las mediciones cambia con el tiempo. Esta condición se conoce como ruido heterocedástico (Akritas, 1997).
- Las distintas clases de estrellas variables están representadas de forma no equitativa en los datos.

Basados en estas observaciones se propone entrenar y evaluar nuevas arquitecturas de redes neuronales que estén adaptadas a estas características para resolver un problema de clasificación automática de curvas de luz del proyecto VVV. A continuación se enuncia la hipótesis y objetivos asociadas a esta propuesta.

1.3. Hipótesis del trabajo

Utilizando nuevas arquitecturas de redes neuronales que tomen en consideración las características intrínsecas de las curvas de luz se alcanzará una mayor exactitud en la clasificación de estrellas variables de VVV con respecto a arquitecturas profundas convencionales y a clasificadores de ML clásico basados en características.

1.4. Objetivos

1.4.1. General

Entrenar y evaluar una red neuronal artificiales con capas convolucionales parciales para la clasificación de series de tiempo astronómicas muestreadas irregularmente

1.4.2. Específicos

1. Estudiar el contexto astronómico y el estado del arte de las redes neuronales para extracción de características y clasificación.

2. Recopilar un conjunto de entrenamiento de curvas de luz de estrellas periódicas de VVV. Implementar estrategias de balance de clases. Dividir en conjuntos de entrenamiento, validación y prueba.

3. Entrenar dos clasificadores de referencia basados en (1) ensambles de árboles de decisión con atributos usados típicamente en astronomía y (2) redes neuronales convolucionales

4. Implementar y entrenar un modelo de red neuronal profunda usando capas convolucionales parciales para curvas de luz irregulares y con distinta cantidad de muestras.

5. Evaluar estas estrategias en el conjunto de prueba. Comparar y analizar los resultados obtenidos por cada método en términos de puntaje F1.

Lo que resta del documento se estructura como sigue. En la sección 2 se definen los sustentos teóricos de los modelos desarrollados, es decir, lo que significa el aprendizaje de máquinas, las redes neuronales y las capas convolucionales. En la sección 3 se detallan las características del conjunto de datos y su preprocesamiento, los componentes del ambiente de desarrollo, la implementación de las capas convolucionales parciales, la arquitectura del modelo propuesto, las configuraciones de entrenamiento y de las métricas de evaluación, y la descripción de los modelos referenciales. Luego, se exponen los resultados obtenidos en la sección 4, donde se compara y analiza los desempeños de cada estrategia utilizada. Finalmente, en la sección 5 se presentan las conclusiones de este trabajo resaltando los principales hallazgos, junto con proponer algunas estrategias a considerar en trabajos futuros.

2. MARCO TEÓRICO

2.1. Machine Learning y Redes Neuronales Artificiales

En comparación con el paradigma de programación tradicional, donde las secuencias de reglas codificadas se aplican a los datos, el paradigma ML se basa en algoritmos para descubrir estas reglas a partir de un conjunto de datos de entrenamiento, es decir, una colección de muestras de datos que ha sido elaborada por expertos humanos. Estos modelos se ajustan a los datos de capacitación mediante la optimización y el aprendizaje estadístico (MacKay, 2002; Bishop, 2006). Una vez que un modelo ha sido entrenado, se usa para predecir datos más generales, es decir, el conjunto de prueba. Un conjunto de entrenamiento que representa el conjunto de prueba de manera apropiada es clave para hacer una inferencia confiable.

Las redes neuronales artificiales (ANN por sus siglas en inglés *Artificial Neural Networks*) son aproximadores de función no lineal construidos mediante la conexión de unidades simples (MacKay, 2002; Bishop, 2006; Goodfellow et al. 2016). Estas unidades o neuronas artificiales se desarrollaron originalmente como modelos simplificados de neuronas biológicas (Rosenblatt, 1958). Las neuronas son básicamente una transformación lineal de la entrada seguida de una función o activación no lineal. Las neuronas están organizadas en capas y varias capas están conectadas para formar una arquitectura. Las diferentes arquitecturas neuronales se especializan en diferentes tipos de datos y tareas. La tarea más común es la clasificación supervisada, en la que la salida de la red se interpreta como un conjunto de probabilidades asociadas con las clases presentes en el problema.
La arquitectura de ANN más común es la red prealimentada y totalmente conectada (FFFC por *Feed-Forward Fully Connected*), también conocida como perceptrón multicapa (MLP por *Multi-Layer Perceptron*). En las arquitecturas prealimentadas, la información fluye desde la entrada hacia la salida sin ningún tipo de retroalimentación. Los datos ingresan a la capa de entrada de la red y cada capa oculta posterior la transforma en representaciones de más alto nivel. En problemas de clasificación, la capa de salida, es decir, la última representación, corresponde a las probabilidades de la clase. El término "Totalmente conectada" se refiere al hecho de que cada neurona oculta y de salida está conectada a todas las neuronas de la capa anterior.

El aprendizaje de la ANN se logra adaptando los pesos de las conexiones neuronales para minimizar una función de costo en el conjunto de datos de entrenamiento. Por lo general, se utilizan algoritmos de optimización de primer orden basados en el descenso de gradiente. El entrenamiento en redes de varias capas se logra utilizando el algoritmo de retropropagación de errores. Cuanto mayor sea el número de capas, más expresivo será el modelo, pero también el entrenamiento se vuelve cada vez más difícil debido a problemas numéricos como el desvanecimiento o la explosión de gradientes (Bengio et al. 1994).

2.2. Aprendizaje profundo

Los recientes avances en potencia computacional y disponibilidad de datos han permitido la exitosa aplicación de redes neuronales con gran cantidad de capas, es decir redes neuronales profundas (Goodfellow et al. 2016). El paradigma de aprendizaje profundo (DL por sus siglas en inglés *Deep Learning*) se basa en el entrenamiento de una ANN muy compleja y flexible directamente en grandes volúmenes de datos en bruto. El modelo profundo realiza la extracción y clasificación de características, es decir, no hay necesidad de ingeniería de características manual.

Los modelos de DL han sido particularmente exitosos en el manejo de datos no estructurados como imágenes, texto y audio. Un ejemplo destacado son las redes neuronales convolucionales o CNN (LeCun et al. 1989), modelos diseñados para datos similares a una cuadrícula, como por ejemplo imágenes (matriz bidimensional de píxeles) y series de tiempo regularmente muestreadas (vector unidimensional).

La CNN utiliza capas especializadas donde las neuronas comparten parámetros y están conectadas solo a un subconjunto de la entrada. Esto es similar a un banco de filtros discretos que están involucrados con la entrada. Cuando se entrena en imágenes naturales, las capas convolucionales aprenden a extraer filtros progresivamente más complejos, comenzando desde bordes y degradados de color hasta figuras complejas como rostros humanos (Zeiler y Fergus, 2014). La arquitectura CNN también tiene capas de *pooling* que se utilizan para disminuir la muestra de las salidas de las operaciones de convolución. En las tareas de clasificación, las capas completamente conectadas se emplean típicamente como capas finales de una CNN. Al cambiar la dimensionalidad de los filtros, la aplicación de CNN a series temporales (1D), imágenes (2D) y video (3D) es sencilla.

Dentro de las aplicaciones de las CNN con capas convolucionales 1D se encuentran aquellas enfocadas a datos temporales como por ejemplo clasificación de eventos sonoros en audio, clasificación de datos biomédicos, monitoreo de calidad estructural, detección de fallas de motores, entre otras (Abdoli et al. 2019; Kiranyaz et al. 2019). Sin embargo, dichas redes esperan que los datos de entrada se encuentren en un cierto orden, en particular que estén equiespaciados temporalmente, lo cual es el caso contrario a lo que sucede con las curvas de luz, donde los tiempos de observación no ocurren con la misma frecuencia y por lo tanto, no presentan la estructura de grilla deseada para la aplicación directa de una CNN.

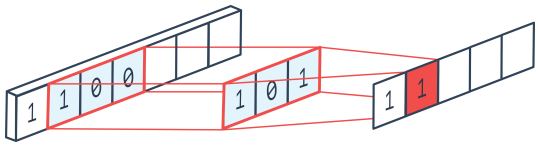


Figura 3. Ejemplo de Convolución 1D.

Una capa convolucional para datos unidimensionales tiene sus neuronas ordenadas como una lista de arreglos unidimensionales todos del mismo largo. El largo y cantidad de estos arreglos son parámetros que el usuario define durante el diseño de la arquitectura del modelo. Estos arreglos o filtros se convolucionan con el arreglo de entrada. La operación de convolución consiste en una secuencia donde en cada paso se realiza una multiplicación punto a punto de los componentes del filtro y la entrada seguido de una reducción suma. El filtro se desplaza en una cierta cantidad de posiciones con respecto a la entrada para repetir el procedimiento anterior hasta que se haya recorrido completamente la entrada. La cantidad de posiciones que el filtro se desplaza en cada iteración se denomina paso o *stride* y es otro hiperparámetro que el usuario debe seleccionar durante el diseño del modelo. Otra consideración de diseño es el uso de relleno o *padding*. El tipo de padding más usual es el relleno con ceros, donde el usuario define una cierta cantidad de ceros que se concatenan en los extremos del arreglo de entrada previo a la convolución. El padding sirve para manejar condiciones de borde y/o para forzar que la salida de la capa convolucional mantega un tamaño equivalente al del vector de entrada. En la Figura 3 se puede apreciar una convolución de una dimensión con un filtro de tamaño 3, con stride de 1 donde no se utiliza padding.

2.3. Capas convolucionales parciales

La convolución parcial es un tipo de convolución donde la salida está condicionada solo a las entradas válidas. Es decir se pondera de forma adaptativa la salida para ajustar la fracción de los datos faltantes. Esta fue propuesta en Liu et al. 2018, donde se utilizó una capa convolucional 2D para restauración de imágenes con partes faltantes o corruptas.

Se refiere tanto a la operación de convolución parcial como a la función de actualización de la máscara conjuntamente como la capa convolucional parcial. Sean $\mathbf{W}$ los pesos del filtro de convolución, $\mathbf{X}$ los valores de la entrada o de la capa anterior y $\mathbf{M}$ la máscara binaria correspondiente. La convolución parcial en cada ubicación se expresa como:

$$x' = \begin{cases} \mathbf{W}^T(\mathbf{X} \otimes \mathbf{M}) \frac{sum(\mathbf{1})}{sum(\mathbf{M})} & \text{si} \quad sum(\mathbf{M}) > 0 \\ 0 & \text{sino} \end{cases} \tag{2}$$

donde $\otimes$ denota la multiplicación elemento a elemento, y $\mathbf{1}$ tiene el mismo tamaño que $\mathbf{M}$ pero con todos los elementos siendo igual a 1. El factor escalar $sum(1)/sum(M)$ aplica una escala apropiada para ajustar la cantidad variable de entradas válidas (desenmascaradas).

Después de cada operación de convolución parcial, se actualiza la máscara de la siguiente manera: si la convolución pudo condicionar su salida en al menos un valor de entrada válido, entonces es marcada esa ubicación como válida. Esto se expresa como:

$$m' = \begin{cases} 1 & \text{si} \quad sum(\mathbf{M}) > 0 \\ 0 & \text{sino} \end{cases} \tag{3}$$

Estas capas convolucionales parciales son útiles para las curvas de luz dado que los datos de observaciones difieren en la cantidad de puntos que forman la curva de luz, es decir, existe diferencia cuantitativa en los registros de magnitud aparente entre distintas estrellas observadas, por lo que es necesario aplicar un método para hacer uniforme este *largo* de las curvas de luz para que formen la grilla uniforme que espera una CNN de 1D.

3. METODOLOGÍA

3.1. Datos, preprocesamiento y ambiente de desarrollo

Un subconjunto con $88,454$ curvas de luz de estrellas variables VVV divididas en tres categorías se utilizan para entrenar y evaluar los modelos de clasificación. Estos datos fueron proporcionados por investigadores del Instituto Milenio de Astrofísica, quienes a su vez obtuvieron las etiquetas mediante inspección manual de las curvas de luz y cruzamiento (crossmatch) con la base de datos del proyecto OGLE (Optical Gravitational Lensing Experiment) (Soszynski et al. 2011), que tiene un traslape importante con VVV en términos de campos de observación.

La Tabla 1 muestra la distribución de los distintos objetos astronómicos presentes en la base de datos en función de su clase. A partir de la tabla se observa que existe un desbalance importante en la base de datos. La Tabla 2 muestra el número promedio de muestras (puntos en la curva de luz) y el tiempo promedio del total de observaciones, ambos con sus respectivas desviación estándar. Se observa que los valores para todas las clases son similares. Cabe mencionar también que la cantidad menor de observaciones en todo el conjunto es 49 y la mayor es 335.

Clase	Número de objetos
Eclipses binarias de contacto	67873
RR Lyrae tipo ab	19087
Cefeidas	1494

Tabla 1. Distribución de clases de estrellas variables presentes en la base de datos

Clase	Número promedio de muestras	Tiempo total entre obs.
Eclipses binarias de contacto	82.52 ± 28.76	1938.82 ± 182.61
RR Lyrae tipo ab	80.38 ± 5.63	1929.49 ± 193.32
Cefeidas	80.85 ± 8.51	1928.34 ± 144.01

Tabla 2. Número de muestras y tiempo de observaciones presentes en la base de datos

La figura 4 muestra un histograma de la densidad de frecuencia en los intervalos de tiempo en días entre observaciones para todas las curvas de luz presentes en el dataset.

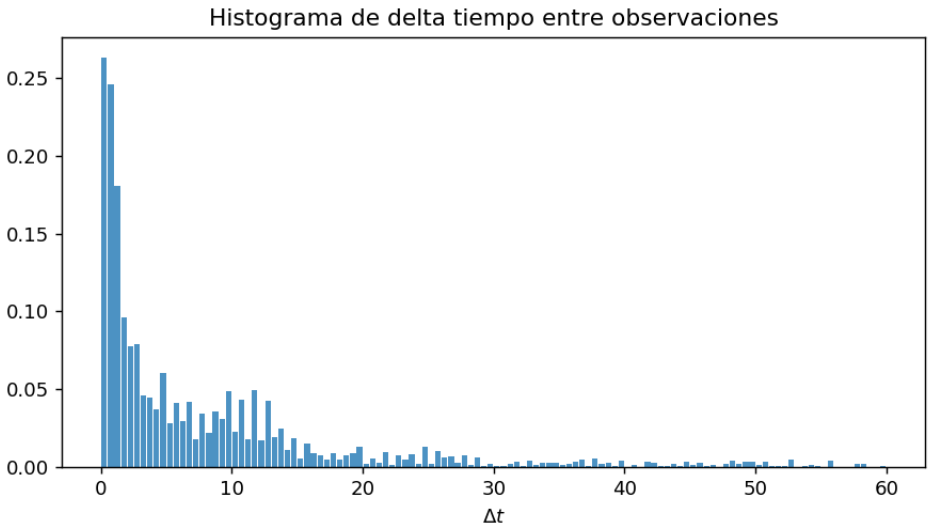


Figura 4. Diferencia entre tiempo de observaciones sucesivas para todas las curvas de luz

El conjunto de datos se preprocesa efectuando las siguientes operaciones:

- Se obtiene la fase de la curva de luz por medio de la ecuación 1. El período se asume conocido y este caso proviene de estudios previos
- Dado que las curvas de luz tienen distinto largo, se aplica zero-padding, es decir que las curvas de luz se completan con ceros a la derecha hasta alcanza el largo máximo (335)
- Se agrega a la curva de luz una máscara binaria cuyos valores son 1 para las posiciones con datos existentes o 0 si corresponde a un espacio rellenado en el proceso de zero-padding

De esta forma se tiene un conjunto de datos uniformado que puede ser manipulado y segmentado en lotes (minibatches) durante el proceso de entrenamiento de forma eficiente.

Para implementar los modelos se utiliza el lenguaje de programación `Python 3.7` y la librería de redes neuronales `PyTorch` 1.7.0 (Paszke et al. 2019). El desarrollo se lleva a cabo usando el interprete `IPython` y el ambiente de cómputo interactivo `Jupyter`. Otras librerías de cómputo científico utilizadas son: `matplotlib` 3.1.3 (Hunter, 2007), `NumPy` 1.18.1 (Walt et al. 2011), `SciKitLearn` 0.23.2 (Pedregosa et al. 2011) y `pandas` 1.0.3 (McKinney et al. 2011).

3.2. Implementación de la capa convolucional parcial unidimensional en PyTorch

A continuación se describe la estructura y funcionamiento de la clase *PartialConv*, la cual representa el núcleo de la innovación y propuesta de este trabajo. Primero se detalla la inicialización y componentes. Luego se explica el funcionamiento de la operación en sí. Esta clase es una adaptación de la versión 2D usada en Liu et al. 2018 para restauración de imágenes.

En primer lugar se muestra la implementación del constructor. La capa convolucional parcial hereda de `torch.nn.Module`, lo cual permite registrar automáticamente los parámetros de los submódulos declarados del modelo y también su transferencia e intercambio entre memorías de CPU y GPU.

```
1  class PartialConv(nn.Module):
2      def __init__(self, in_channels_C,in_channels_M, out_channels,
       kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True):
3          super().__init__()
4          self.input_conv = nn.Conv1d(in_channels_C, out_channels,
       kernel_size, stride, padding, dilation, groups, bias)
5          self.mask_conv = nn.Conv1d(in_channels_M, out_channels,
       kernel_size, stride, padding, dilation, groups, False)
6
7          torch.nn.init.constant_(self.mask_conv.weight, 1.0)
8          # mask is not updated
9          for param in self.mask_conv.parameters():
10             param.requires_grad = False
```

El constructor recibe como entrada los canales con datos (**X**) junto a una máscara binaria (**M**). Recibe también los parámetros usuales de una capa convolucional como: tamaño del filtro, stride, padding, etc.

En las lineas 4 y 5 se declaran dos submódulos, las cuales corresponden a dos convoluciones usuales de 1D, una para los datos y otra para actualizar la máscara, respectivamente. Finalmente se inicializan los pesos de la capa de convolución de la máscara con un valor constante. También se desactivan sus gradientes dado que el modelo no debe *aprender* nada de ella, sino que es sólo una guía para ponderar operaciones.

A continuación se muestra la función `forward` del módulo, encargada de realizar la operación de convolución parcial sobre un *minibatch* de datos

```
1      def forward(self,input, mask):
2          output = self.input_conv(input * mask)
3          if self.input_conv.bias is not None:
4              output_bias = self.input_conv.bias.view(1, -1, 1).expand_as
       (output)
5          else:
```

```
            output_bias = torch.zeros_like(output)

        with torch.no_grad():
            output_mask = self.mask_conv(mask)

        no_update_holes = output_mask == 0
        mask_sum = output_mask.masked_fill_(no_update_holes, 1.0)
        output_pre = (output - output_bias) / mask_sum + output_bias
        output = output_pre.masked_fill_(no_update_holes, 0.0)
        new_mask = torch.ones_like(output)
        new_mask = new_mask.masked_fill_(no_update_holes, 0.0)

        return output, new_mask
```

La operación principal ocurre siguiendo el proceso descrito junto a la ecuación 2, es decir, se multiplica la entrada por la máscara actual, luego se pondera por los respectivos pesos. Finalmente se actualiza la máscara siguiendo la Ecuación 3.

3.3. Arquitectura de red neuronal para curvas de luz

El modelo de red convolucional final se obtuvo de un proceso iterativo empírico, es decir, se realizaron variadas pruebas para modelos distintos con el fin de obtener el mejor resultado de clasificación.

El modelo de red neuronal cuenta con tres capas convolucionales parciales, una capa de max pooling adaptativo y tres capas ocultas totalmente conectadas. Se utiliza *ReLU* (Nair y Hinton, 2010) como función de activación para cada capa excepto en la de máx pooling y en la de salida donde se usa *Softmax* (Bridle, 1990). Está demostrado que ReLU mejora el entrenamiento de las redes neuronales profundas. La función de activación ReLU se define como

$$\text{ReLU}(x) = \text{máx}(0, x),$$

es decir que satura en 0 cuando $x < 0$ y se comporta como una función lineal cuando $x \geq 0$. La función Softmax se define como

$$\sigma(z)_i = \frac{\exp(z_i)}{\sum_{j=1}^{n} \exp(z_j)}, 1 \leq i \leq n$$

La Figura 5 muestra una representación visual de la arquitectura descrita, donde se detallan las dimensiones intermedias de entrada y salida para cada operación. Los filtros para cada capa tienen tamaño 3, con stride de 2 y sin padding. El tamaño del feature map final que produce el max pooling adaptativo es de $64x1$. Se reciben dos canales como entrada la magnitud aparente y el error asociado de la curva de luz, ambos normalizados. Además se recibe como canal adicional la máscara binaria que opera la convolución parcial. La

cantidad total de parámetros entrenables del modelo es de 6691. La salida final, es decir, de la última capa, tiene tamaño 3 dado que representa la clasificación entregada por el modelo como probabilidad de pertenecer a cada una de las clases de estrella variable consideradas.

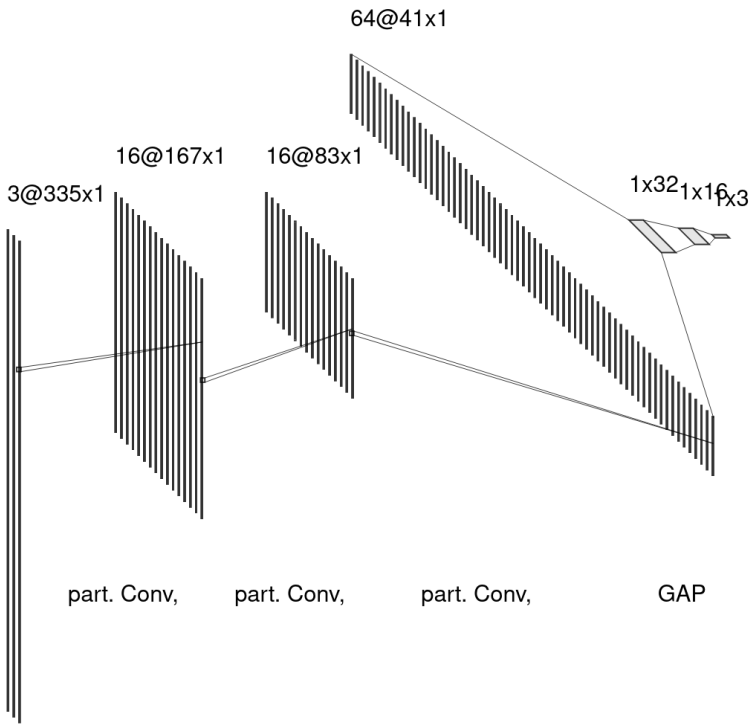


Figura 5. Representación visual de la arquitectura de red neuronal usada.

3.4. Detalles del entrenamiento

Para entrenar el modelo se utiliza gradiente descendente estocástico con tasa de aprendizaje adaptativa (ADAM) (Kingma y Ba, 2014) y con lotes (minibatches) de tamaño 32. La tasa de aprendizaje inicial se fija en 0,0002. La función de costo a optimizar es la entropía cruzada, que es la más apropiada para un problema de clasificación. Los parámetros de la red neuronal se inicializan aleatoriamente siguiendo una distribución uniforme cuyo rango depende del tamaño de la entrada.

El conjunto de datos se particiona de manera estratificada en sets de entrenamiento, evaluación y validación dejando $70, 20$ y $10\,\%$ de los datos respectivamente. Debido al gran desbalance en el muestreo de las clases, se utiliza un selector especializado para

la selección de batches, llamado *Imbalanced Dataset Sampler*[2] cuyo funcionamiento a grandes rasgos consiste en eliminar muestras de las clases mayoritarias (submuestreo) y/o agregar más ejemplos de la clase minoritaria (sobremuestreo) de forma dinámica durante el entrenamiento.

Para la evaluación del modelo se implementa *early-stopping* al entrenamiento cuando pasan 300 épocas sin que aumente el macropromedio F1-Score. Se utilizan lotes de datos de tamaño 64 para el conjunto de validación.

3.5. Evaluación de resultados

Para evaluar el desempeño de los modelos se utiliza como métrica el macropromedio de F1-Score por el hecho de que promedia sobre el desempeño de clases individuales siendo más robusto al desbalance. Primero se calculan la precisión y la exactitud usando:

$$P_{macro} = \frac{1}{|G|}\sum_{i=1}^{|G|}\frac{TP_i}{TP_i + FP_i} = \frac{\sum_{i=1}^{|G|} P_i}{|G|}$$

$$R_{macro} = \frac{1}{|G|}\sum_{i=1}^{|G|}\frac{TP_i}{TP_i + FN_i} = \frac{\sum_{i=1}^{|G|} R_i}{|G|}$$

Donde |G| es el número de clases. La precisión y la exactitud macro-promediadas dan lugar a la puntuación macro F1:

$$F1_{macro} = 2\frac{P_{macro}R_{macro}}{P_{macro} + R_{macro}}$$

Si el puntaje $F1_{macro}$ tiene un valor alto, esto indica que un clasificador funciona bien para cada clase individual. Por lo tanto, el macropromedio es más adecuado para datos con una distribución de clases desequilibrada.

[2]https://github.com/ufoym/imbalanced-dataset-sampler

3.6. Modelos de referencia

3.6.1. Random Forest

Los resultados del modelo propuesto se contrastan con un modelo de referencia basado en un ensamble de árboles de decisión de tipo Random Forest (RF). Se considera una metodología clásica y ampliamente usada para la clasificación de curvas de luz. En primer lugar se computa una tabla de atributos para el conjunto de curvas de luz usando la librería de procesamiento de series de tiempo astronómicas FATS (Nun et al. 2015). Se consideran todas los atributos de la librería que pueden aplicarse a series unidimensionales, siendo los más relevantes

1. Periodo y modelo armónico: El periodo de la curva de luz estimado mediante ajuste de mínimos cuadrados de un modelo trigonométrico. Una vez que se tiene el período se ajusta un modelo de mayor orden y los coeficientes armónicos se usan como atributo

2. Autocorrelación: Se calcula la autocorrelación discreta y la autocorrelación para series irregulares. La distancia hasta el primer máximo de cada una se usa como atributo

3. Modelo CAR: La media y desviación estándar de un modelo autoregresivo continuo ajustado a la curva de luz

4. Momentos estadísticos: Se usan como atributos la media, varianza, simetría y kurtosis de la curva de luz. También se usan como atributos versiones robustas a los outliers como la mediana y rango intercuartil

Una vez calculados los atributos se procede a ajustar el clasificador RF. Debido a que el conjunto de datos es sumamente desbalanceado se considera una versión de RF donde las muestras utilizadas para entrenar cada árbol de decisión se balancean por submuestreo (undersampling) de las clases mayoritarias. Este modelo de RF balanceado está implementado en la librería de Python `imblearn`[3] (Lemaître et al. 2017).

Para calibrar el modelo se utiliza una estrategia de validación cruzada con cinco particiones estratificadas, es decir manteniendo la proporción de clase. Para comparar con el modelo propuesto se selecciona la combinación de hiperparámetros que maximiza el promedio F1 macro en el conjunto de validación. Los hiperparámetros del modelo que son calibrados en esta etapa y los valores que se prueban son:

[3] https://imbalanced-learn.readthedocs.io

1. Cantidad de árboles: 50, 500, 5000
2. Profundidad de árbol: 10, 20 y máxima, es decir hasta que todos los nodos sean puros
3. Criterio para separación de rama: Entropía de Shannon o índice de Gini
4. Estrategia de remuestreo: Submuestrear todos las clases excepto la minoritaria versus remuestrear todas las clases
5. Muestreo con reemplazo: Permitir o denegar la reselección de elementos al muestrear

3.6.2. Modelos convolucionales secundarios

Para establecer una base comparativa y a su vez dentro del proceso iterativo de búsqueda de un modelo acertado, se desarrollaron ciertos modelos secundarios que ponían a prueba determinadas características, ya sea en la arquitectura como en las configuraciones de entrenamiento. Estos modelos se caracterizan por:

- Modelo convolucional parcial que no usa balanceo por lotes, donde se puso a prueba una estrategia de sobremuestreo y submuestreo de clases para cada minibatch. La arquitectura es la misma utilizada para el modelo final.
- Modelo convolucional que no utiliza zero-padding para rellenar las curvas de luz, por lo cual se utilizan lotes de entrenamiento de tamaño 1. Además, tampoco se utiliza máscara binaria por lo que la arquitectura implementa convoluciones convencionales.
- Modelo convolucional con zero-padding, el cual tampoco utiliza máscara binaria y es idéntico al anterior en arquitectura, sin embargo, el hecho de implementar zero-padding en las curvas de luz permite el uso de lotes de mayor tamaño.

4. RESULTADOS y ANÁLISIS

En esta sección se comparan y analizan los resultados obtenidos para cada modelo, a través de matrices de confusión con valores porcentuales entre 0 y 100, además de gráficos para diagnosticar el aprendizaje de la red.

4.1. Diagnóstico de entrenamiento del modelo propuesto

En primer lugar, se muestran comportamientos y resultados del entrenamiento para el modelo de red convolucional parcial propuesto, lo que involucra el análisis de curvas de aprendizaje para diagnosticar la convergencia y/o potencial sobreajuste durante el entrenamiento. La Figura 6 muestra un ejemplo de curvas de aprendizaje, donde a la izquierda se aprecian las funciones de costo minimizadas (*Loss)* y a la derecha el puntaje F1 macro alcanzado en el conjunto de validación.

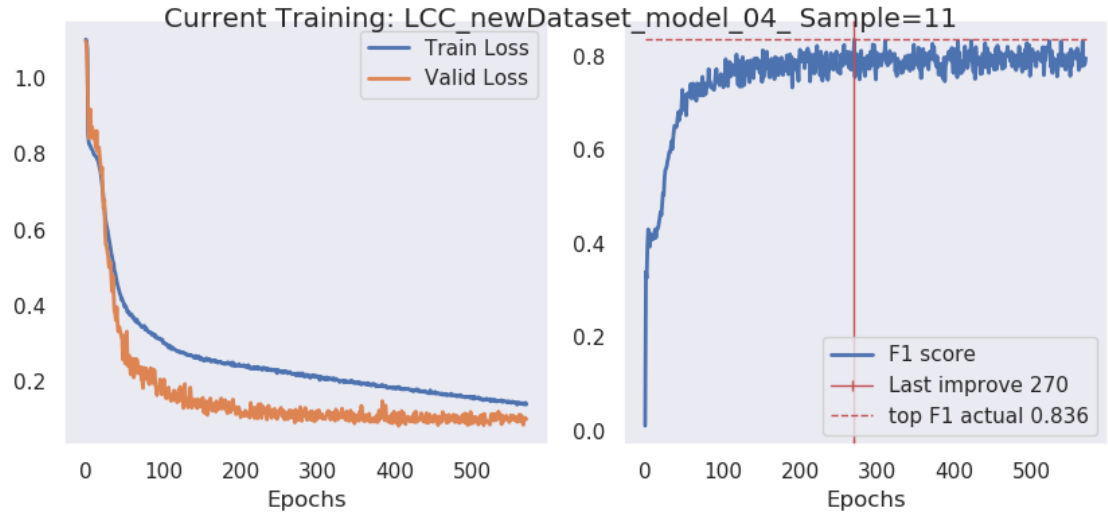
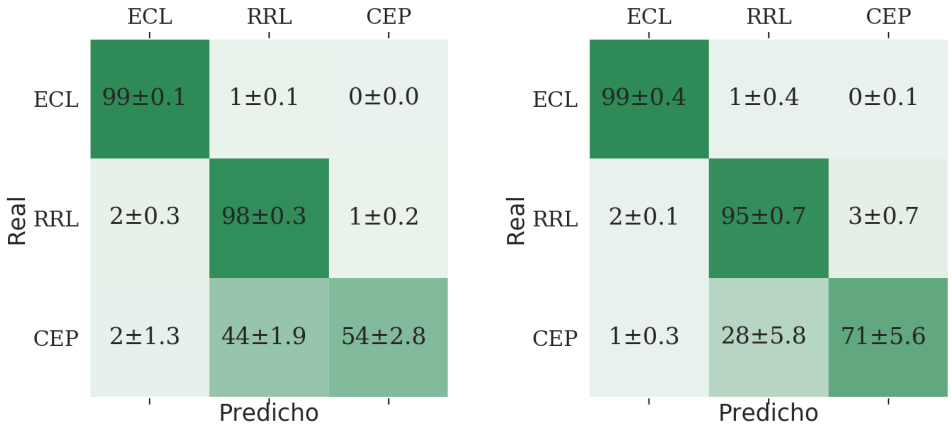


Figura 6. Curva de aprendizaje

Se observa que último incremento en el puntaje F1 de validación ocurre en la época 270 y el entrenamiento se detiene luego de esperar 300 épocas desde el último incremento (paciencia). Esto coincide en gran medida con el estancamiento de la función de costo para la validación (en naranjo). El comportamiento presentado en la Figura 6 es representativo de lo observado para los entrenamientos en general. La época de convergencia promedio usando el dataset VVV y la arquitectura propuesta es de 321,4 ± 102,73. Los entrenamientos nunca superaron las 503 épocas.

4.2. Evaluación de estrategia de aumentación de datos a nivel de minibatch

En segundo lugar se evalúa la importancia de contar con datos balanceados al momento de entrenar, es decir, que las representaciones de las muestras por clase sean similares en tamaño. En ciertas ocasiones esto es difícil de alcanzar dada la naturaleza de los datos, sin embargo existen estrategias para lidiar con estos inconvenientes, como es el caso de *Imbalanced Dataset Sampler* (IDS) utilizado para aplicar submuestreo y sobremuestreo a las clases con más y menos elementos, respectivamente. La Figura 7 muestra matrices de confusión para dos entrenamientos de la arquitectura convolucional parcial, sin y con IDS, respectivamente.



(a) No usando IDS (b) Usando IDS

Figura 7. Comparación uso de Imbalanced Dataset Sampler

Se observa que cuando no se usa IDS, el resultado de clasificación se ve muy afectado por las proporciones de representatividad en las clases (véase Tabla 1), llegando a presentar gran confusión entre RR Lyrae y Cefeida. Esta confusión se disminuye considerablemente cuando se usa IDS para entrenar, observándose una mejora en más de 20 % para la clase minoritaria (Cefeidas). Como se puede ver en la Tabla 3 el promedio F1 macro de ambos modelos no difiere considerablemente, esto debido a que la mejora en la clasificación de Cefeidas va de la mano con una pequeña disminución en los porcentajes de recuperación de las clases mayoritarias y a que el F1 macro es robusto al desbalance de clases.

Modelo	P_{macro}	R_{macro}	$F1_{macro}$
pConv con IDS	0.843 ± 0.01	0.884 ± 0.017	0.861 ± 0.008
pConv sin IDS	0.935 ± 0.012	0.837 ± 0.009	0.873 ± 0.003

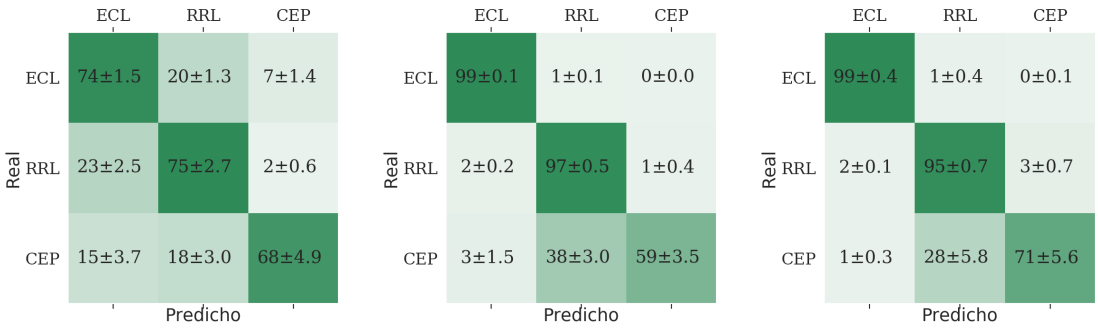
Tabla 3. Tabla comparativa de desempeño entre estrategias de muestreo

4.3. Comparación con modelo convolucional convencional

Como se describió en la Sección 1.2, las curvas de luz representan registros de observaciones que no se realizan con una frecuencia de muestreo constante resultando en series de tiempo con distinta cantidad de puntos y distinto espaciado temporal entre sus puntos. Esto supone un desafío para la aplicación de la operación de convolución discreta convencional.

En esta sección se compara la propuesta de convolución parcial basada en máscaras con dos estrategias referenciales basadas en la convolución discreta. La primera estrategia de referencia consiste en utilizar capas convolucionales convencionales sobre minibatches de tamaño 1 y curvas de luz sin zero-padding. En esta estrategia es clave la capa de max pooling adaptativo pues esta se encarga de retornar una salida que es de un tamaño constante sin importar el tamaño de la entrada que recibe. En la segunda estrategia se utilizan las curvas de luz extendidas con zero-padding pero se ignora la máscara. Todas las demás condiciones y parámetros se mantienen igual.

La Figura 8 muestra tres resultados: (a) Usando convolución usual sin máscara ni zero-padding, en cuyo caso el entrenamiento se realizó con lotes de tamaño 1. (b) Usando convolución discreta usando zero-padding para lotes de tamaño idéntico; y (c) Usando la capa de convolución parcial propuesta.



(a) Conv. no-padding (b) Conv. zero-padding (c) partial Conv

Figura 8. Comparación uso de convolución parcial y zero-padding.

En primer lugar se observa que el caso (a) es notoriamente inferior que los demás en todas las clases a excepción de las Cefeidas donde supera al caso (b). El caso (b) en cambio obtiene resultados similares al modelo propuesto (c), siendo un 2 % superior, equivalente a 38 curvas de luz, en la clasificación de RR Lyrae y un 12 % inferior en el caso de clase Cefeida, lo que equivale a 18 curvas de luz.

Sin embargo, estudiando en más detalle el modelo (b), es notable que difícilmente podría generalizar en un conjunto de datos distinto pues está afectado por un importante sesgo. La clase de las Eclipsantes Binarias (ECL) tiene en promedio un número de muestras mayor (véase Tabla 2). Esto significa que en general las curvas de luz más rellenadas con ceros son las RR Lyrae y las Cefeidas. Esto provoca que el clasificador que utiliza padding sin máscara (b) aprenda rápidamente a separar entre ECL y las otras clases sólo en base a la cantidad de ceros que tiene. Por lo tanto la estrategia ingenua de entrenar directamente con curvas de luz extendidas con zero-padding no se considerará en lo que sigue, enfoncándose sólo en la alternativa con capas convolucionales parciales.

Otro detalle importante a considerar es el tiempo de entrenamiento por época el cual se muestra en la Tabla 4. En el caso (a) donde no se ocupan minibatches, el tiempo es más de un orden de magnitud superior que los casos (b) y (c). El caso (c) toma un tiempo 43.3 % superior con respecto al caso (b), lo cual es esperable pues cada capa de convolución parcial realiza dos convolucionales convencionales en su implementación.

Estrategia	Δt por época [s]
No-padding	372.78 ± 4.57
Zero-padding	12.3 ± 0.19
partial Conv	17.62 ± 0.12

Tabla 4. Tiempos promedio de entrenamiento por épocas para cada estrategia.

En lo que sigue se profundiza el análisis considerando el caso particular de las RR Lyrae en el conjunto de prueba. Para esto tipo de estrella variable el modelo RF recupera exitosamente un 92 % lo cual equivale a 1755 curvas de luz. Sin embargo se tiene también 478 falsos positivos correspondientes a un 7 % del total de Eclipses Binarias y a un 2 % del total de Cefeidas. Con respecto a este res

4.4. Comparación con clasificador basado en features

Es importante tener en cuenta que una red neuronal no es la única solución a un problema de clasificación como este. Existen otros modelos como por ejemplo el Random Forest (RF), que también pueden llevar a cabo la tarea en cuestión y que como fue explicando en la Sección 1.2 ha sido ampliamente utilizado. Sin embargo este modelo requiere de

Modelo	P_{macro}	R_{macro}	$F1_{macro}$
pConv con IDS	0.843 ± 0.01	0.884 ± 0.017	0.861 ± 0.008
Balanced RF	0.709 ± 0.001	0.918 ± 0.001	0.769 ± 0.001

Tabla 5. Tabla comparativa de desempeño entre modelos

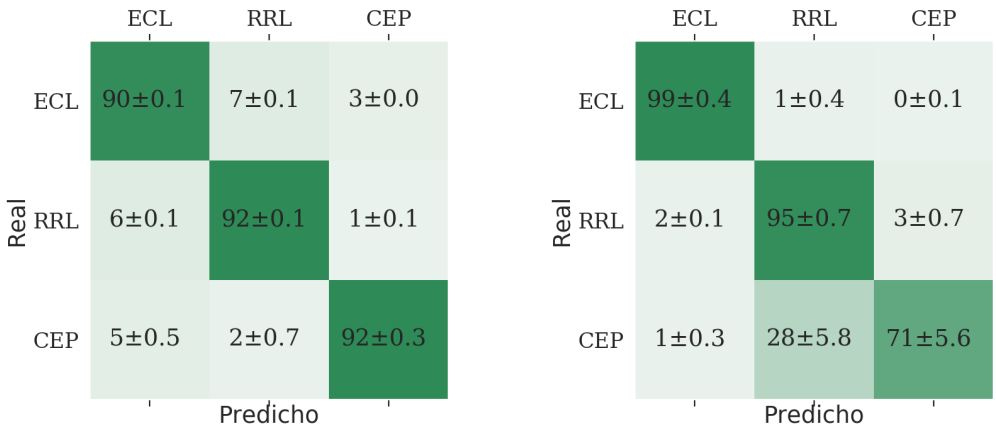
representar la curvas de luz usando características y no es claro ni sencillo definir el mejor conjunto de atributos.

En esta sección se compara el modelo clasificador propuesto, el cual no considera el cálculo de features y predice la clase directamente desde la curva de luz, contra un clasificador referencial basado en un modelo RF balanceado sobre los atributos calculados por la librería FATS (ver Sección 3.6). Del proceso de validación cruzada con el modelo RF balanceado se obtiene que la mejor combinación de parámetros es 5000 clasificadores débiles, profundidad máxima de árbol de 20, criterio de índice de Gini para separación de ramas, submuestreo de todas las clases y selección con reemplazo.

La Tabla 5 muestra el promedios macro del recall, precisión y el puntaje F1 score de ambos modelos. La Figura 9 muestra la matriz de confusión para los modelos comparados en el conjunto de prueba. Podemos notar que el modelo RF balanceado obtuvo un rendimiento similar para todas las clases. El modelo propuesto es un 9 % superior identificando correctamente estrellas eclipsantes, 3 % superior identificando correctamente RR Lyrae y un 20 % inferior en el caso de las Cefeidas, presentados estas confusión principalmente con las RR Lyrae.

En lo que sigue se profundiza el análisis considerando el caso particular de las RR Lyrae en el conjunto de prueba. Para esto tipo de estrella variable el modelo RF recupera exitosamente un 92 % lo cual equivale a 1755 curvas de luz. Sin embargo se tiene también 478 falsos positivos correspondientes a un 7 % del total de Eclipses Binarias y a un 2 % del total de Cefeidas. Con respecto a este resultado, la red neuronal con capas convolucionales parciales recupera un 2 % adicional de RR Lyrae (38 curvas de luz) pero con una contaminación notablemente menor de sólo 109 curvas de luz correspondientes a un 1 % del total de las Eclipses Binarias y a un 28 % del total de las Cefeidas. Es claro que para el objetivo de detectar RR Lyrae el modelo propuesto es superior al Random Forest.

Aproximadamente la mitad del total de las falsas RR Lyrae predichas por el modelo de red corresponden a la clase Cefeida. Estas falsas detecciones podrían en teoría separarse usando el período ya que esta característica intrínseca de la variabilidad de estas estrellas presenta una diferencia reconocida como muestra la Figura 2. La Figura 10 muestra la distribución del período de las RR Lyrae del conjunto de test estimado por expertos. De la figura es claro que usar el período como entrada a la red neuronal podría ayudar a



(a) Random Forest

(b) pConv

Figura 9. Comparación desempeño vs modelo referencia basado en Random Forest

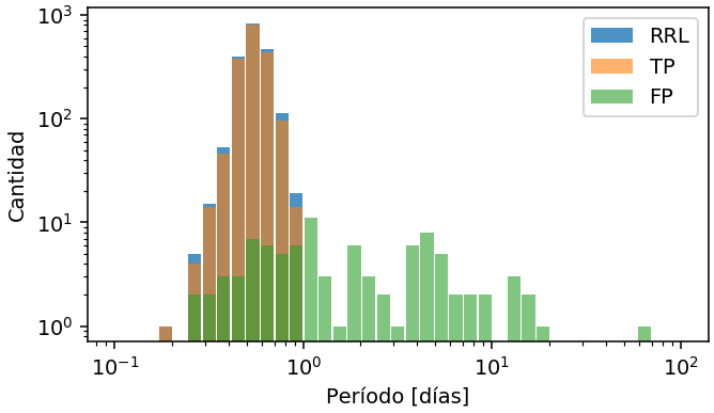


Figura 10. Histograma del período para el subconjunto de las RR Lyrae en el conjunto de test (color azul). En color naranjo se muestran los periodos de las curvas de luz predichas correctamente como RR Lyrae por el modelo y en verde los falsos positivos (Cefeidas y Eclipsantes Binarias).

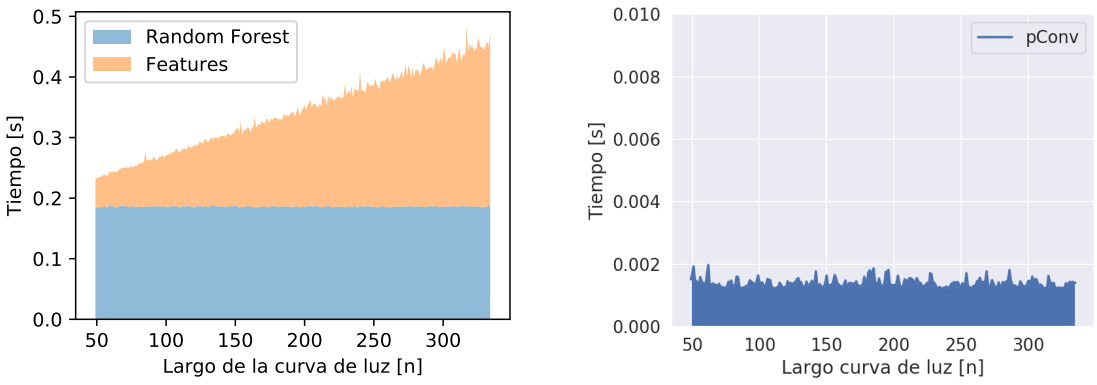
disminuir la cantidad de falsos positivos (histograma verde). A priori podríamos suponer que agregar el período y otros features de relevancia física como entradas adicionales a las capas fully connected del modelo propuesto podría mejorar el resultado obtenido. Esto se explorará como trabajo futuro.

Analizando cuantitativamente las predicciones de cada modelo, se obtiene que el modelo RF recupera correctamente 1770 (92,72 %) RR Lyrae, mientras que el modelo

convolucional parcial recupera 1805 (94,558 %) RR Lyrae de las 1909 del conjunto de test. Ahora bien, si se analizan en detalle las predicciones, tenemos que la intersección de ambas predicciones consiste en 1688 (88, 42 %) de efectividad mutua. La diferencia entre lo predicho por el modelo propuesto y RF es de 117 (6, 13 %), mientras que la diferencia de lo predicho por el Random Forest y el modelo propuesto es de 82 (4, 3 %)

Por otro lado, no sólo es importante el resultado de clasificación, sino que también lo es el tiempo de inferencia para estos modelos, es decir el tiempo que toma clasificar una curva de luz. En la Figura 11 se muestran el tiempo de inferencia para (a) Random forest como referencia y (b) Modelo de red neuronal convolucional parcial. Se observa que para el caso del modelo Random Forest existe un tiempo constante en la inferencia de 188 milisegundos, a lo cual se agrega el tiempo para calcular los atributos o features para cada curva de luz, las cuales cabe recordar que son de largo variable dependiendo de la cantidad de observaciones registradas en cada una. Se puede observar que se superan los 0,4 segundos en las curvas más largas.

Para el caso de la red con capas convolucionales parciales se observa que el tiempo de inferencia es independiente del largo original de la curva de luz, con una media de 1,388 ± 0,138 milisegundos. Cabe destacar que si bien el largo inicial de las curvas de luz es distinto, todas se someten al zero-padding para la utilización de minibatches, quedando de largo 335 con mayor o menor cantidad de relleno. Aún así, el tiempo de inferencia de la red convolucional parcial es de dos órdenes de magnitud menor que el del modelo referencial, por lo que resultaría más apropiado en escenarios de clasificación en tiempo casi-real como el que presentan telescopios modernos como el Vera Rubin Observatory.



(a) Random Forest (b) pConv

Figura 11. Comparación de tiempo inferencia entre el modelo propuesto y el modelo referencia basado en Random Forest

5. CONCLUSIONES

En este trabajo se propone un modelo de red neuronal convolucional para un problema de clasificación de curvas de luz de estrellas variables en base al uso de capas convolucionales parciales, que a diferencia de otras implementaciones basadas en redes neuronales toman en consideración el muestreo irregular de las curvas de luz.

Este modelo se entrena en un subconjunto de la base de datos VVV contrastándose con una estrategia referencial ampliamente utilizada basada en Random Forest, obteniendo un resultado de clasificación que es 10 % superior en términos de puntaje F1. Para el caso particular de la clase objetivo, es decir, para las RR Lyrae, el modelo propuesto obtiene una mayor tasa en recuperación y una menor contaminación con curvas de luz de otro tipo de estrellas, por lo que se cumple la hipótesis principal de este trabajo.

Como trabajo futuro, es importante considerar que los modelos comparados son complementarios y que según el análisis realizado podría ser de interés añadir algunas features esenciales directamente en algunas neuronas de las capas fully connected, tales como el período o la amplitud. Notar que esta información se pierde en la metodología actual pues las curvas se doblan y normalizan antes de ser clasificadas. Además, es menester identificar qué partes o características de las curvas de luz son distintivas para cada modelo, en otras palabras, identificar qué es lo que les permite a los algoritmos discernir entre una clase u otra. Se propone entonces realizar un análisis más acabado y profundo de los filtros aprendidos y como estos afectan a la clasificación.

Otra avenida que se explorará durante la extensión de este trabajo como tesis de magister son las capas convolucionales paramétricas. Estas capas podrían adaptarse automáticamente y de mejor manera a la distribución no estructurada de las series de tiempo de las curvas de luz, donde la idea clave es usar como filtro o kernel de convolución una función paramétrica que abarque todo el espacio vectorial continuo, lo cual reemplaza el filtro discreto usual de las capas convoluciones por una función determinista, ya sea gaussiana, laplaciana u otra.

Referencias

Abdoli, S., Cardinal, P. & Koerich, A. L. (2019). End-to-end environmental sound classification using a 1D convolutional neural network. *Expert Systems with Applications*, *136*, 252-263.

Aguirre, C., Pichara, K. & Becker, I. (2019). Deep multi-survey classification of variable stars. *Monthly Notices of the Royal Astronomical Society*, *482*(4), 5078-5092.

Akritas, M. G. (1997). Astronomical (Heteroscedastic) Measurement Errors: Statistical Issues and Problems, En *Statistical Challenges in Modern Astronomy II*. Springer.

Becker, I., Pichara, K., Catelan, M., Protopapas, P., Aguirre, C. & Nikzat, F. (2020). Scalable end-to-end recurrent neural network for variable star classification. *Monthly Notices of the Royal Astronomical Society*, *493*(2), 2981-2995.

Bengio, Y., Simard, P. & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157-166.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg, Springer-Verlag.

Borne, K. D. (2010). Astroinformatics: data-oriented astronomy research and education. *Earth Science Informatics*, *3*(1-2), 5-17. https://doi.org/10.1007/s12145-010-0055-2

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

Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition, En *Neurocomputing*. Springer.

Catelan, M., Pritzl, B. J. & Smith, H. A. (2004). The RR Lyrae Period-Luminosity Relation. I. Theoretical Calibration. *The Astrophysical Journal Supplement Series*, *154*(2), 633-649. https://doi.org/10.1086/422916

Catelan, M. & Smith, H. A. (2015). *Pulsating Stars*.

Feigelson, E. D. (2016). The changing landscape of astrostatistics and astroinformatics. *Proceedings of the International Astronomical Union*, *12*(S325), 3-9. https://doi.org/10.1017/S1743921317003453

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning* [http://www.deeplearningbook.org]. MIT Press.

Hinners, T. A., Tat, K. & Thorp, R. (2018). Machine Learning Techniques for Stellar Light Curve Classification. *The Astronomical Journal*, *156*(1), 7. https://doi.org/10.3847/1538-3881/aac16d

Huijse, P., Estevez, P. A., Protopapas, P., Principe, J. C. & Zegers, P. (2014). Computational Intelligence Challenges and Applications on Large-Scale Astronomical Time Series Databases. *IEEE Computational Intelligence Magazine*, *9*(3), 27-39.

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science Engineering*, *9*(3), 90-95. https://doi.org/10.1109/MCSE.2007.55

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

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M. & Inman, D. J. (2019). 1D convolutional neural networks and applications: A survey. *arXiv preprint arXiv:1905.03554*.

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), https://doi.org/10.1162/neco.1989.1.4.541, 541-551. https://doi.org/10.1162/neco.1989.1.4.541

Lemaître, G., Nogueira, F. & Aridas, C. K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research*, *18*(17), 1-5. http://jmlr.org/papers/v18/16-365

Liu, G., Shih, K. J., Wang, T., Reda, F. A., Sapra, K., Yu, Z., Tao, A. & Catanzaro, B. (2018). Partial Convolution based Padding. *CoRR*, *abs/1811.11718*arXiv 1811.11718. http://arxiv.org/abs/1811.11718

MacKay, D. J. C. (2002). *Information Theory, Inference & Learning Algorithms*. USA, Cambridge University Press.

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., Medina, G., Pignata, G., Martín, J. S., Hamuy, M. & Muñoz, R. R. (2018). The High Cadence Transit Survey (HiTS): Compilation and Characterization of Light-curve Catalogs. *The Astronomical Journal*, *156*(5), 186. https://doi.org/10.3847/1538-3881/aadfd8

McKinney, W. Et al. (2011). pandas: a foundational Python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, *14*(9).

Minniti, D., Lucas, P., Emerson, J., Saito, R., Hempel, M., Pietrukowicz, P., Ahumada, A., Alonso, M., Alonso-Garcia, J. & Arias, J. (2010). VISTA Variables in the Via Lactea (VVV): The public ESO near-IR variability survey of the Milky Way. *New Astronomy*, *15*(5), 433-443. https://doi.org/10.1016/j.newast.2009.12.002

Nair, V. & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines, En *ICML*.

Narayan, G., Zaidi, T., Soraisam, M. D., Wang, Z., Lochner, M., Matheson, T., Saha, A., Yang, S., Zhao, Z., Kececioglu, J., Scheidegger, C., Snodgrass, R. T., Axelrod, T., Jenness, T., Maier, R. S., Ridgway, S. T., Seaman, R. L., Evans, E. M., Singh, N., ... and, S. Z. (2018). Machine-learning-based Brokers for Real-time Classification of the LSST Alert Stream. *The Astrophysical Journal Supplement Series*, *236*(1), 9. https://doi.org/10.3847/1538-4365/aab781

Nun, I., Protopapas, P., Sim, B., Zhu, M., Dave, R., Castro, N. & Pichara, K. (2015). FATS: Feature Analysis for Time Series.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. Et al. (2019). Pytorch: An imperative style, high-

performance deep learning library, En *Advances in neural information processing systems*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. Et al. (2011). Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, 2825-2830.

Percy, J. R. (2007). *Understanding variable stars*. Cambridge University Press.

Quezada, C. (2020). *Implementación de un algoritmo de Random Forest para la clasificación automática de estrellas variables en el survey VVV*. Pontificia Universidad Católica de Chile.

Richards, J. W., Starr, D. L., Butler, N. R., Bloom, J. S., Brewer, J. M., Crellin-Quick, A., Higgins, J., Kennedy, R. & Rischard, M. (2011). ON MACHINE-LEARNED CLASSIFICATION OF VARIABLE STARS WITH SPARSE AND NOISY TIME-SERIES DATA. *The Astrophysical Journal*, *733*(1), 10. https://doi.org/10.1088/0004-637x/733/1/10

Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. *Psychological Review*, 65-386.

Saito, R. K., Zoccali, M., McWilliam, A., Minniti, D., Gonzalez, O. A. & Hill, V. (2011). Mapping the X-shaped Milky Way bulge. *The Astronomical Journal*, *142*(3), 76.

Skowron, D. M., Skowron, J., Mróz, P., Udalski, A., Pietrukowicz, P., Soszyński, I., Szymański, M. K., Poleski, R., Kozłowski, S., Ulaczyk, K., Rybicki, K. & Iwanek, P. (2019). A three-dimensional map of the Milky Way using classical Cepheid variable stars. *Science*, *365*(6452), https://science.sciencemag.org/content/365/6452/478.full.pdf, 478-482. https://doi.org/10.1126/science.aau3181

Soszynski, I., Dziembowski, W., Udalski, A., Poleski, R., Szymanski, M., Kubiak, M., Pietrzynski, G., Wyrzykowski, L., Ulaczyk, K., Kozlowski, S. Et al. (2011). The optical gravitational lensing experiment. the OGLE-iii catalog of variable stars. XI. RR Lyrae stars in the Galactic Bulge. *arXiv preprint arXiv:1105.6126*.

Walt, S. v. d., Colbert, S. C. & Varoquaux, G. (2011). The NumPy array: a structure for efficient numerical computation. *Computing in science & engineering*, *13*(2), 22-30.

Warner, B. D. Et al. (2006). *A practical guide to lightcurve photometry and analysis* (Vol. 300). Springer.

Zeiler, M. D. & Fergus, R. (2014). Visualizing and understanding convolutional networks, En *In Computer Vision–ECCV 2014*, Springer.