
Universidad Internacional de La Rioja

Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Inteligencia Artificial

# Rethinking Stella: comparativa de redes neuronales convolucionales para la clasificación de fulguraciones estelares en curvas de luz de la misión TESS.



| | |
|---|---|
| Trabajo fin de estudio presentado por: | Ansony Rolando Medina Baca |
| Tipo de trabajo: | Comparativa de soluciones |
| Director: | Roberto Baena Galle |
| Codirector: | Octavi Fors Aldrich |
| Fecha: | 1/03/23 |

# Resumen

Este estudio tiene como objetivo mejorar el desempeño del algoritmo Stella en la clasificación de fulguraciones estelares en curvas de luz obtenidas a través de la misión TESS. Para ello, se evaluaron 11 modelos con diferentes arquitecturas de redes neuronales convolucionales (CNN), incluyendo 6 arquitecturas conocidas y 5 nuevas implementaciones. Los resultados demuestran que los 5 modelos propuestos en el segundo experimento mejoran significativamente las métricas de *accuracy*, *precision* y *recall*, logrando una mejora de 2.1%, 3.02% y 7.1% respectivamente en el conjunto de validación. Además, se obtuvo una mejora de 5.4% en la clasificación de verdaderos positivos en el conjunto de validación y 4.42% en el conjunto de test según sus matrices de confusión. Además, las arquitecturas optimizadas permiten reducir en un 67.2% el número de parámetros entrenables con respecto al algoritmo Stella.

**Palabras clave:** (De 3 a 5 palabras)

CNN, TESS, Fulguraciones Estelares, curvas de luz, Astrofísica

# Abstract

This study aims to improve the performance of the Stella algorithm in the classification of stellar flares in light curves obtained through the TESS mission. To do this, 11 models with different convolutional neural network (CNN) architectures were evaluated, including six known architectures and five new implementations. The results show that the five models proposed in the second experiment significantly improve the accuracy, precision, and recall metrics, achieving an improvement of 2.1%, 3.02%, and 7.1% respectively in the validation set. Additionally, a 5.4% improvement was obtained in the classification of true positives in the validation set and 4.42% in the test set according to their confusion matrix. Furthermore, the optimized architectures allow reducing the number of trainable parameters by 67.2% compared to the Stella algorithm.

**Keywords**: (De 3 a 5 palabras en inglés)

CNN, TESS, Stellar Flares, Light Curves, Astrophysics

# Índice de contenidos

# Índice de figuras

# Índice de tablas

# 1. Introducción

## 1.1. Motivación

Este Trabajo de Fin de Máster aborda el desafío de clasificar fulguraciones estelares mediante el uso de redes neuronales convolucionales (CNN) entrenadas con curvas de luz de la misión TESS. La clasificación de estas curvas de luz es esencial para conocer cuán activa es una estrella a partir de sus fulguraciones. Tales fenómenos, que se acontecen en la fotosfera de la estrella, pueden tener importantes implicaciones sobre la composición de la atmósfera de los planetas que orbitan esas estrellas activas.

En la actualidad, existen diversas herramientas para detectar fulguraciones estelares basadas en métodos estadísticos que detectan valores atípicos, pero estos requieren ajustes manuales por parte del usuario. Por lo tanto, es crucial automatizar el proceso de clasificación para mejorar la eficiencia y la precisión de los resultados.

El algoritmo Stella (Feinstein, Montet & Ansdell, 2020) es uno de los primeros intentos para solucionar la problemática de la clasificación automática de fulguraciones estelares. Basado en redes neuronales convolucionales, reduce la interacción entre el usuario y el ajuste manual de parámetros, lo que permite una clasificación más eficiente en curvas de luz.

## 1.2. Planteamiento del problema

En la actualidad las redes neuronales y algoritmos de aprendizaje automático han invadido el campo de la química, física y astronomía para dar solución a diversos problemas, que puede incluir desde predecir la estructura de una proteína, como lo hace AlphaFold (Jumper et al., 2021), hasta la detección de radiogalaxias (Teimourian & Dimililer, 2019).

De la misma forma la inteligencia artificial está siendo utilizada para la búsqueda y clasificación de exoplanetas. En este Trabajo de Fin de Máster, se aborda el uso de la inteligencia artificial para la clasificación de fulguraciones estelares a partir de curvas de luz proporcionadas por la misión TESS de la NASA. Se compararán diferentes arquitecturas basadas en redes neuronales para alcanzar una solución eficaz y automatizada a la problemática. La información para el

entrenamiento será obtenida a partir del catálogo de fulguraciones TESS creado por Günther et al. (2020).

## 1.3. Estructura de la memoria

En el siguiente Capítulo se presenta el contexto que dio lugar a la necesidad de clasificar fulguraciones estelares de manera automática con el fin de mejorar la caracterización de las estrellas activas y la habitabilidad de sus exoplanetas orbitantes. Se brinda una revisión histórica de las misiones que han sido llevadas a cabo para recopilar información sobre la existencia de exoplanetas. Las curvas de luz forman parte de la información recabada por estas misiones.

También se hace una revisión sobre las herramientas desarrolladas para poder clasificar fulguraciones estelares en curvas de luz. Ya sea con mucha interacción humana o de manera automática. En este mismo capítulo se definen algunos conceptos técnicos que se utilizaran en los capítulos posteriores.

En el Capítulo 3, se establece el objetivo global y general de la investigación, así como sus objetivos específicos. En este mismo capítulo se plantea la metodología utilizada para alcanzar estos objetivos, incluyendo los pasos a seguir y los indicadores para evaluar el éxito.

En el Capítulo 4, se presenta el diseño de la comparativa que se llevará a cabo. Se incluye una descripción detallada de cómo se llevará a cabo la comparativa, así como una revisión exhaustiva de las arquitecturas de redes neuronales que se utilizarán en el capítulo posterior.

En el Capítulo 5, se desarrollan las comparativas en dos partes. Cada comparativa incluye métricas como el *accuracy*, *precision*, *recall*, *F1-Score* y curva *ROC-AUC*, así como un seguimiento visual mediante gráficas del rendimiento de los modelos. En el Capítulo 6 discutimos y analizamos los resultados obtenidos.

En el Capítulo 7, se presentan las conclusiones clave de la investigación y se evalúa si se han cumplido con éxito los objetivos establecidos en el Capítulo 3. También se identifican las posibles direcciones futuras de investigación que pueden derivar de nuestro trabajo actual.

# 2. Contexto y estado del arte

## 2.1. EXOPLANETAS

### 2.1.1. Definición de exoplanetas

Nuestro sistema solar alberga 8 planetas que enlistados en base a su proximidad al Sol son: Mercurio, Venus, Tierra, Marte, Júpiter, Saturno, Urano y Neptuno. De los cuales los primeros cuatro son denominados rocosos y los últimos gaseosos. Los planetas que están fuera de nuestro sistema solar reciben el nombre de exoplanetas. Los exoplanetas normalmente orbitan otras estrellas, minoritariamente existen exoplanetas que orbitan libremente, estos son llamados planetas rebeldes, que orbitan el centro galáctico y no están atados a ninguna estrella (Administración Nacional de Aeronáutica y el Espacio [NASA], 2021).

### 2.1.2. Detección de exoplanetas

Fischer et al. (2014) describe diversos métodos de cómo detectar exoplanetas, como microlentes, imagen directa, astrometría, variaciones temporales de tránsito y duración de tránsito y variaciones de pulsos. Entre los métodos más destacables podemos encontrar el método de la velocidad radial, el método de los tránsitos y el método de las curvas de fase o variaciones de brillo orbitales. Sin embargo, existen métodos indirectos para poderlos detectar.

La tasa de fulguraciones en estrellas jóvenes son objetivo atractivo en el estudio de exoplanetas. Dichas fulguraciones provocan daños químicos irreparables en la atmósfera de los exoplanetas, pueden acelerar la erosión atmosférica, principalmente en los primeros cientos de millones de años cuando aún se están contrayendo (Feinstein, Montet, Ansdell, et al., 2020). Cabe destacar que la existencia de fulguraciones no garantiza la existencia de exoplanetas orbitando la estrella.

También, el estudio de la tasa de (super)fulguraciones en estrellas frías en la secuencia principal es de capital importancia para determinar cuán comunes son procesos fotoquímicos claves en las atmósferas planetarias como, por ejemplo, la creación/destrucción de gases de efecto invernadero.

La combinación del análisis de las fulguraciones estelares en combinación con manchas estelares o de tránsito funcionan muy bien para acotar y refinar la búsqueda de exoplanetas.

En la actualidad, se han confirmado más de 5000 exoplanetas, que incluyen tanto mundos pequeños y rocosos similares a la Tierra, como gigantes gaseosos que son mucho más grandes que Júpiter.

## 2.2. FULGURACIONES ESTELARES

### 2.2.1. Definición de fulguración estelar

> "Las fulguraciones son explosiones magnéticas que ocurren en las atmósferas de todas las estrellas que poseen una zona de convección exterior. Son fenómenos de alta energía, durante los cuales la reconexión magnética desencadena una liberación repentina de energía en forma de radiación electromagnética desde bandas de radio, pasando por óptica y ultravioleta (UV) hasta emisión de rayos X blandos y duros, cada uno con una evolución temporal característica." (Ilin, Poppenhäger, et al., 2022)

Las fulguraciones estelares son erupciones de energía de alta intensidad que se producen en la superficie de una estrella. Estas erupciones son causadas por la liberación repentina y masiva de energía en forma de luz, radiación y partículas cargadas. Las fulguraciones estelares están relacionadas con las tormentas solares, que son erupciones de energía de baja intensidad que se producen en el Sol. Ambas erupciones pueden tener efectos en la Tierra, causando perturbaciones en los campos magnéticos terrestres y en los sistemas eléctricos. Las fulguraciones estelares son eventos temporales, generalmente duran unos minutos, aunque en algunos casos pueden durar horas.

## 2.3. CURVAS DE LUZ

Las curvas de luz son gráficos que muestran el brillo de un objeto durante un período de tiempo. En nuestro caso los objetos son las estrellas. (NASA, 2013)

Las curvas de luz de las estrellas son gráficos que muestran cómo cambia la magnitud aparente de una estrella a medida que avanza el tiempo. Estas curvas proporcionan información sobre el periodo de la estrella, su brillo máximo y su variación en el tiempo. Las curvas de luz también

se usan para medir la variación en el brillo de la estrella en un tiempo dado, así como para determinar si la estrella está girando. Esto ayuda a los astrónomos a comprender mejor la física de las estrellas y sus variaciones en el tiempo.

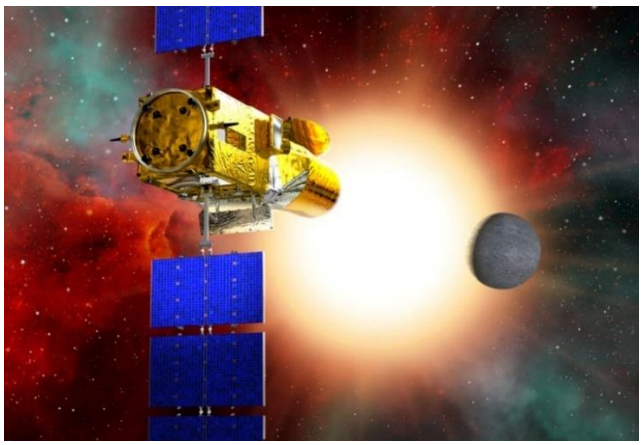
Las curvas de luz de las estrellas pueden ser calculadas con distintos instrumentos desde un smartphone hasta el actual satélite TESS, es claro notar que la calidad de los datos será mayor en dispositivos especializados para esta tarea. Debido a la gran variedad de equipos que pueden generar curvas de luz, en el siguiente apartado nos centraremos en los principales satélites cuya misión es brindarnos esas curvas de luz.

### 2.3.1. CoRoT Satellite

**Co**nvection, **Ro**tation and planetary **T**ransits (COROT) es un satélite lanzado por French Space Agency (CNES) en conjunto con European Space Agency (ESA) en búsqueda de exoplanetas rocosos. (Baglin et al., 2006).

CoRoT es un satélite de observación de la Agencia Espacial Europea (ESA). Fue lanzado el 27 de diciembre de 2006 y fue diseñado para estudiar el interior de los planetas, estrellas y galaxias. Está equipado con una cámara de luz visible, una cámara infrarroja y una cámara de rayos X. Está diseñado para observar el universo desde un punto de vista único. Está equipado con una cámara de luz visible de alta precisión, lo que le permite observar los cambios de brillo de una estrella mientras se observa. El satélite CoRoT fue desactivado el 17 de junio del 2014.

***Figura 1:*** *Ilustración del satélite espacial CoRoT.*



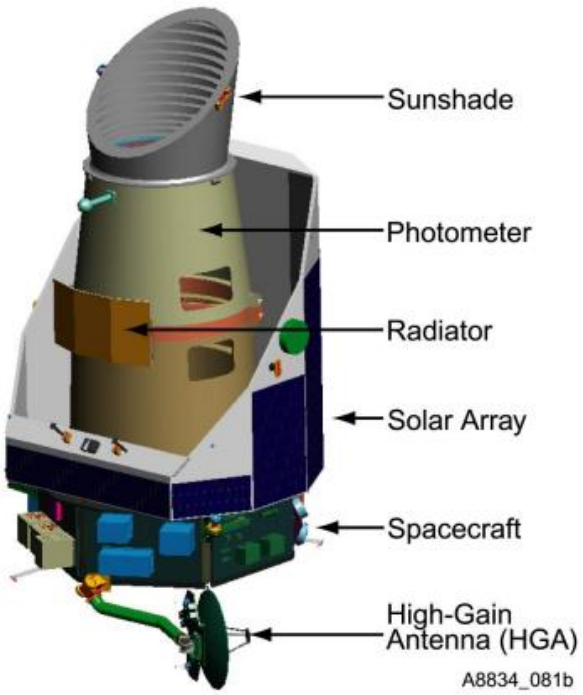
Fuente: (*CoRoT*, s. f.)

### 2.3.2. Misión Kepler.

Kepler es una iniciativa del Programa Discovery de la NASA, llevada a cabo a través de una misión espacial con un enfoque específico: identificar planetas terrestres similares en tamaño y composición al nuestro, en órbita alrededor de estrellas distintas a nuestro Sol (Borucki et al., 2003).

La misión Kepler fue lanzada el 6 de marzo de 2009 desde Cabo Cañaveral, Florida, con el objetivo de identificar y caracterizar exoplanetas. Equipada con una cámara digital de alta precisión que detecta el parpadeo de luz de estrellas distantes, la misión permitió identificar los planetas que orbitan alrededor de ellas y proporcionar información importante sobre la formación de los sistemas de exoplanetas. La misión se convirtió en la primera de la NASA en descubrir una gran cantidad de exoplanetas, permitiendo a los astrónomos estudiar el contenido de nuestra galaxia y descubrir nuevos mundos. La misión finalizó el 5 de noviembre de 2018.

***Figura 2:*** *Ilustración del Telescopio Espacial Kepler donde se muestran algunos de sus componentes.*

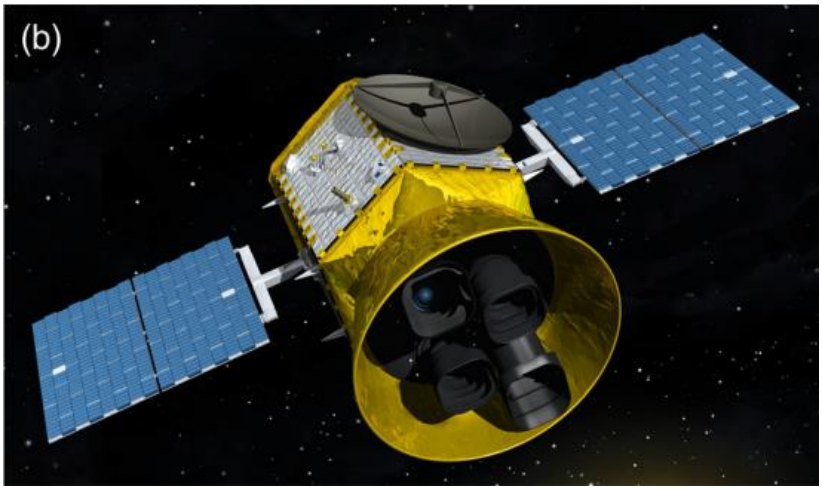



Fuente: (Borucki et al., 2003)

### 2.3.3. Misión TESS

TESS (Transiting Exoplanet Survey Satellite) es una misión de la NASA diseñada para buscar planetas fuera de nuestro sistema solar. Se lanzó el 18 de abril de 2018 y usa el método de tránsitos para detectar los planetas. Esto significa que mide la disminución de la luz emitida por una estrella mientras un planeta pasa delante de ella. Esta misión se concentra principalmente en estrellas cercanas a la Tierra, buscando planetas de tamaño terrestre y superiores que orbitan en la zona habitable de estrellas. El objetivo de TESS es descubrir exoplanetas para futuros estudios de detalle de su atmósfera y características. La misión también está buscando estrellas variables, recursos de asteroides (Ricker et al., 2015).

***Figura 3:*** *Ilustración del Satélite de la misión TESS*




Fuente: (Ricker et al., 2015)

## 2.4. ESTADO DEL ARTE EN LA DETECCIÓN DE FULGURACIONES ESTELARES EN CURVAS DE LUZ

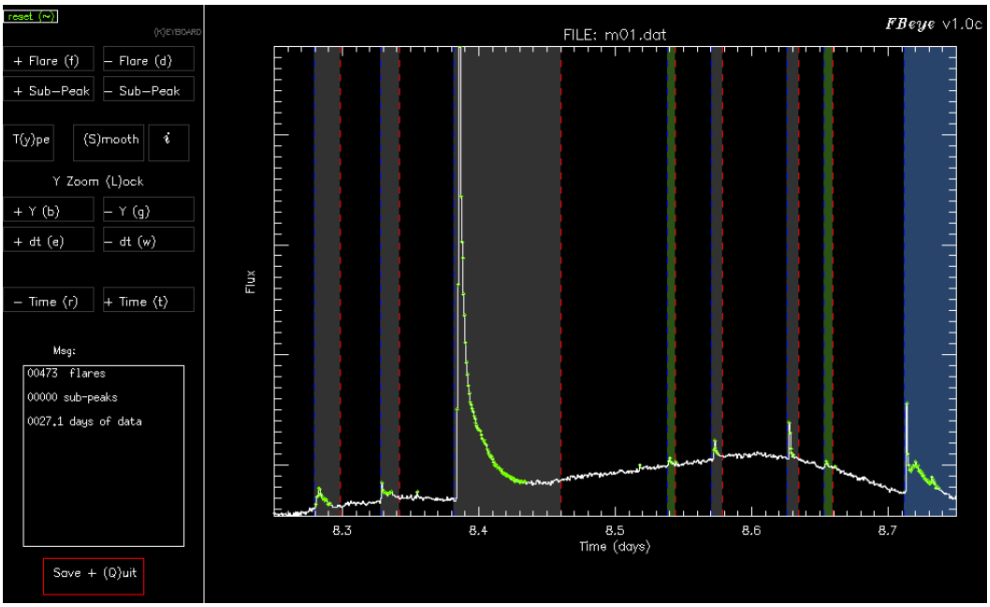
### 2.4.1. Flares By EYE

FBEYE es la suite de detección "Flares By-Eye", desarrollada por el grupo de investigación de astronomía estelar de baja masa de la Universidad de Washington. Incorpora algoritmos de muchos autores y el trabajo de múltiples proyectos de tesis doctorales bajo la supervisión de

la Prof. Suzanne L. Hawley. Este algoritmo está escrito en IDL (Interactive Data Language) y sirve para ver curvas de luz e identificar fulguraciones (Davenport et al., 2014).

El algoritmo FBEYE es una técnica de análisis de curvas de luz de estrellas que busca detectar fulguraciones estelares mediante análisis visual. El algoritmo se basa en la idea de que las fulguraciones estelares son eventos transitorios que se caracterizan por un aumento repentino y breve en el brillo de una estrella. El algoritmo utiliza una interfaz gráfica de usuario (GUI) que permite al usuario examinar visualmente las curvas de luz y señalar eventos transitorios que se asemejan a las fulguraciones estelares.

***Figura 4:*** *Ejemplo de vista del algoritmo FBEYE visualizando una curva de luz.*



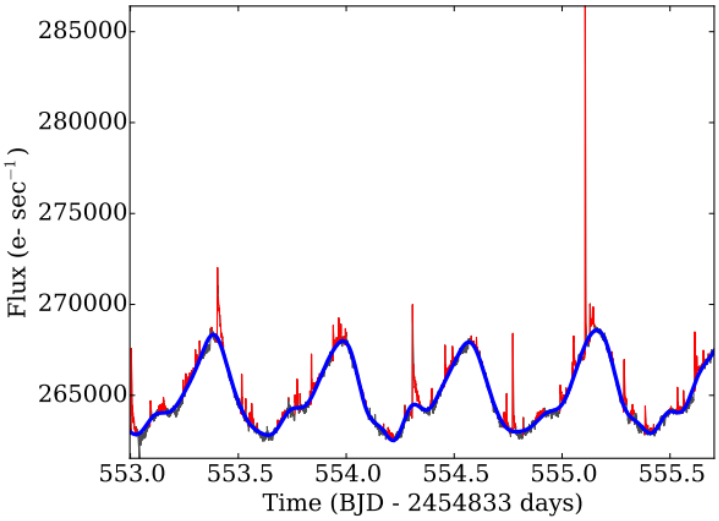

Fuente: (Davenport et al., 2014)

El algoritmo FBEYE es una técnica de detección manual, lo que significa que requiere la intervención humana para seleccionar eventos transitorios y calcular los parámetros de las fulguraciones estelares detectadas. Aunque es una técnica poco automatizada, FBEYE es útil para detectar fulguraciones estelares en curvas de luz con un gran nivel de ruido y eventos transitorios que no son detectados por algoritmos automatizados.

### 2.4.2. Appaloosa

Appaloosa (Davenport, 2016) es otro algoritmo escrito en Python para la clasificación de fulguraciones estelares en curvas de luz proporcionadas por la misión Kepler.

***Figura 5:*** *Se muestra Kepler SAP FLUX (línea negra) con el modelo de curva de luz inactiva final superpuesto (línea azul). Las fulguraciones recuperadas en este análisis están resaltadas (líneas rojas).*




Fuente: (Davenport, 2016)

### 2.4.3. AltaiPony

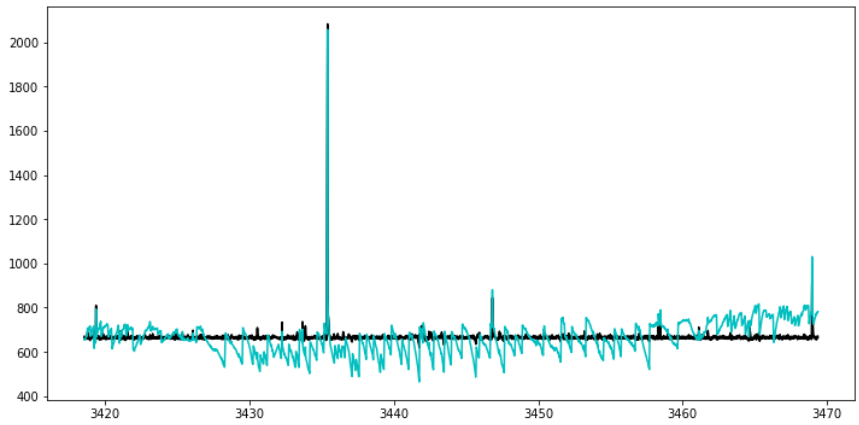
AltaiPony desarrollado en Python, es una caja de herramientas para estudios estadísticos de fulguraciones adaptadas, pero no limitados a observaciones de series temporales obtenidas por las misiones Kepler, K2 y TESS (Ilin et al., 2019).

La funcionalidad de AltaiPony incluye:

- Eliminación de tendencias de curvas de luz: es un paso previo necesario para detectar las fulguraciones estelares con precisión, ya que las tendencias pueden interferir con la detección.
- Búsqueda y caracterización de fulguraciones estelares: el algoritmo busca eventos transitorios en las curvas de luz y los caracteriza mediante diferentes parámetros como tiempo de inicio, tiempo de finalización, brillo máximo y ancho temporal.

- Inyección y recuperación de fulguraciones sintéticas: esta función permite al usuario inyectar eventos transitorios sintéticos en las curvas de luz y evaluar la capacidad del algoritmo para recuperarlos.
- Comandos de visualización: AltaiPony proporciona una interfaz gráfica para visualizar las curvas de luz y los resultados del análisis.
- Métodos para el análisis estadístico de muestras de fulguraciones: el algoritmo proporciona herramientas para analizar las estadísticas de las muestras de fulguraciones detectadas.

***Figura 6:*** *La línea azul es una curva de luz de la misión K2 y la línea negra es el resultado después del procesado por el algoritmo AltaiPony, en donde observamos que las fulguraciones se mantienen y la variabilidad de la intensidad de la luz se suaviza.*




Fuente: (Ilin, Schmidt, et al., 2022)

### 2.4.4. FLATW'RM

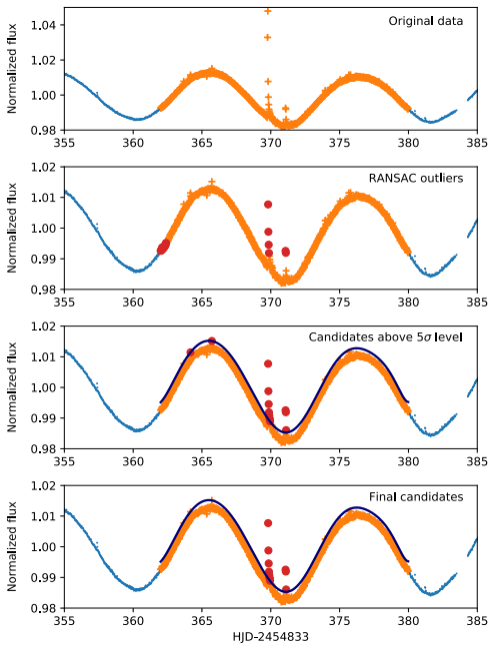
FLATW'RM es un modelo de inteligencia artificial entrenado con el algoritmo RANdom SAmple Consensus (RANSAC) con curvas de luz proporcionadas por la misión Kepler (Vida & Roettenbacher, 2018). RANSAC es un algoritmo de aprendizaje automático que se utiliza para estimar modelos en conjuntos de datos con *outliers*. Este algoritmo es capaz de interpretar/suavizar los datos experimentales que contiene una gran cantidad de errores (Fischler & Bolles, 1981).

Este algoritmo consta de separar los datos en dos grupos, inliers y outliers (en nuestro caso fulguraciones estelares). Este algoritmo se puede resumir en los siguientes pasos:

- Se selecciona un pequeño subconjunto de datos (muestra aleatoria) que se supone que son inliers (Non Flare).
- Se estima un modelo a partir de esa muestra.
- Se determinan todos los datos que están dentro de un margen determinado del modelo estimado (inliers).
- Se vuelve a estimar un modelo a partir de los inliers.
- Se repiten los pasos 3 y 4 hasta que se encuentre un número suficiente de inliers o se alcance un número máximo de iteraciones (criterios de detención).

Los datos que están fuera del margen del modelo estimado son considerados como posibles fulguraciones.

***Figura 7:*** *Demostración del algoritmo FLATW'RM en donde la primera figura es la curva de luz original normalizada. La segunda figura muestra los outliers identificados por el algoritmo RANSAC, en la tercera figura se establece un umbral de 5σ y la última figura se seleccionan los outliers por sobre el umbral 5σ que representan los candidatos finales.*



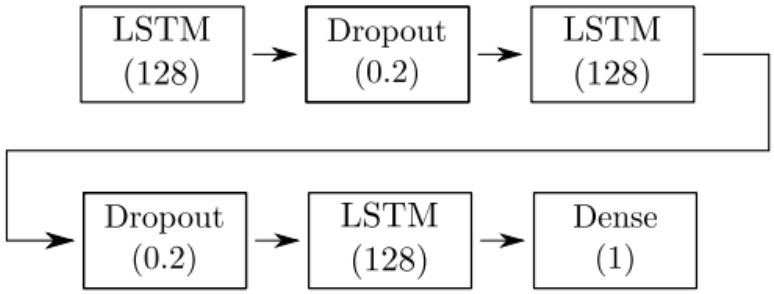

Fuente: (Vida & Roettenbacher, 2018)

### 2.4.5. FLATW'RM2

FLATW'RM2 es un modelo de inteligencia artificial entrenado con una arquitectura RNN (Red Neuronal Recurrente) para la clasificación de fulguraciones estelares en curvas de luz proporcionadas por la misión Kepler y TESS (Vida et al., 2021).

FLATW'RM2 consta de 3 capas LSTM (Long Short-Term Memory) con un tamaño de celda de 128 unidades cada una. La red neuronal posee 2 capas Dropout con una tasa de 0.2, lo cual implica que, en cada iteración del proceso de entrenamiento, se desactivan de manera aleatoria cerca del 20% de las unidades neuronales. Y una capa densa de una neurona con la función de activación sigmoidal, lo que indica que es una red de clasificación binaria y la función sigmoid determina un rango entre 0 y 1 que puede ser considerado como una probabilidad.

***Figura 8:*** *En esta figura se muestra la arquitectura de la RNN FLATW'RM2.*




Fuente: (Vida et al., 2021)

### 2.4.6. Stella

Stella es un paquete open-source de Python para identificar fulguraciones en los datos de dos minutos de TESS con redes neuronales convolucionales (CNN) (Feinstein, Montet, & Ansdell, 2020).

Los métodos de detección de fulguraciones estelares previos a Stella eliminan las tendencias en las curvas de luz mediante la aplicación de una heurística para detectar valores atípicos, los cuales son considerados como candidatos. Sin embargo, estos métodos requieren de la

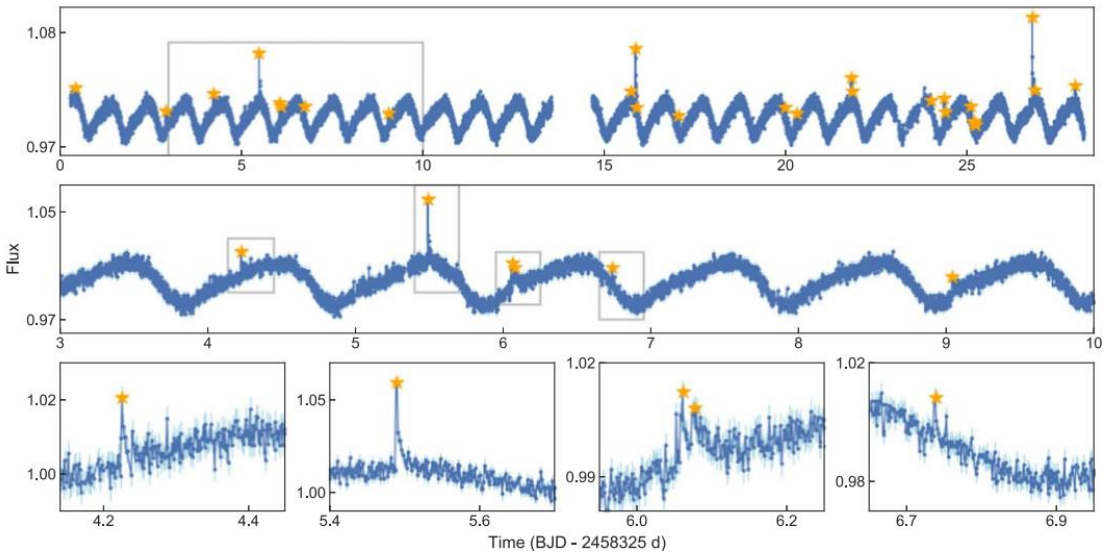
definición de un umbral de valores atípicos, lo cual puede resultar en la exclusión de fulguraciones de baja energía, sesgando los resultados hacia fulguraciones de mayor energía.

El paquete Stella posee diversos módulos que detallamos a continuación:

El módulo **download_nn_set** contiene la clase **DownloadSets** que se encarga de descargar el catálogo de fulguraciones estelares y las curvas de luz necesarias para entrenar, testear y validar la red neuronal convolucional.

El catálogo que usa Stella en este módulo es el "TESS Catalog of Stellar Flares" (Günther et al., 2020). Es una base de datos completa de fulguraciones estelares observadas por el satélite TESS en el sector 1 y 2 durante los primeros dos meses de la misión. El catálogo incluye información detallada sobre la ubicación de las estrellas, su brillo y otras características, así como datos específicos sobre cada fulguración estelar observada.

***Figura 9:*** *Demostración de la detección de picos candidatos a fulguraciones. En el eje X se muestra el tiempo en Barycentric Julian Days (BJD) y en el eje Y se muestra el TESS PDC-SAP flux normalizado. Los símbolos de estrellas naranjas resaltan los picos candidatos a fulguraciones detectados y los rectángulos inferiores muestran una vista ampliada de los rectángulos marcados en gris.*

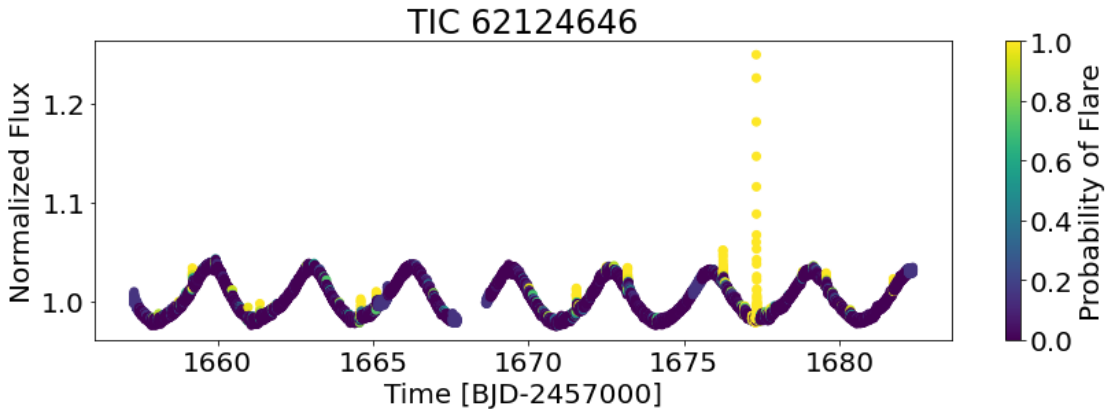


Fuente: (Günther et al., 2020)

Este módulo también incluye el método para poder descargar los modelos pre entrenados listos para hacer predicciones.

El módulo **mark_flares** contiene la clase **FitFlares** que recibe como parámetros la curva de luz y la predicción proporcionada por el modelo, pasándolo por un umbral de probabilidad para aceptar el evento como una fulguración, por defecto está definida en 0.5.

***Figura 10:*** *Demostración de la predicción fulguraciones del algoritmo Stella.*

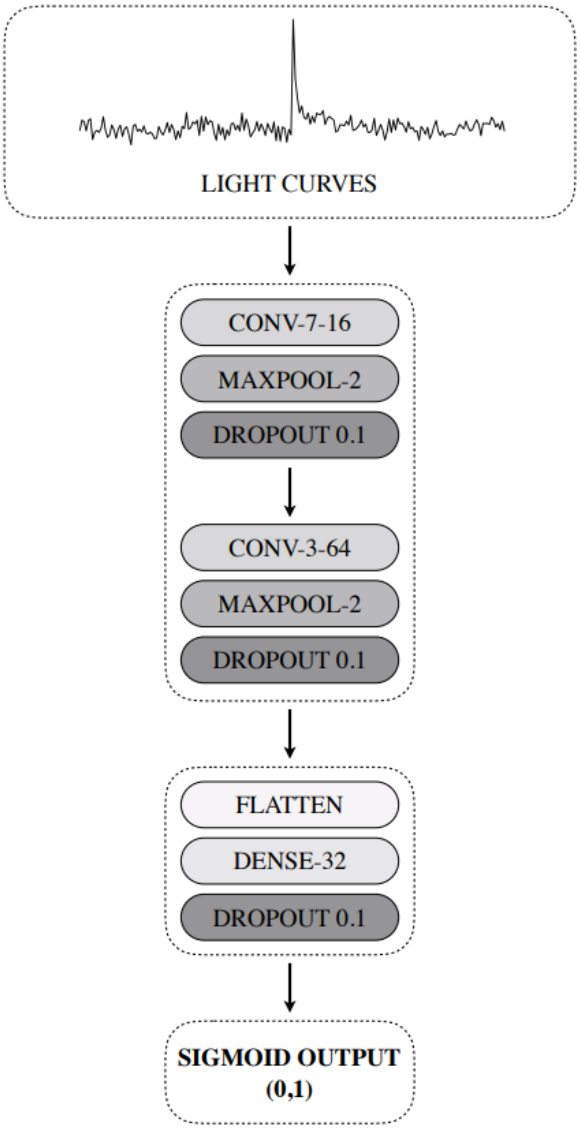



Fuente: (Feinstein, Montet, Ansdell, et al., 2020)

El módulo **metrics** contiene la clase **ModelMetrics** que proporciona las métricas utilizadas para evaluar el modelo entrenado. Entre las cuales resaltan accuracy, precision y recall. Detallaremos más al respecto en el Capítulo 2.5.3.

El módulo **neural_network** contiene la clase **ConvNN** en la cual se define la arquitectura de la red neuronal convolucional empleada para entrenar el modelo, así como la definición de los métodos **create_model** que se utiliza para crear un nuevo modelo basado en la arquitectura de la CNN, **load_model** que carga los modelos ya pre entrenados, **train_models** método utilizado para iniciar el entrenamiento del nuevo modelo, **cross_validation** método utilizado para entrenar el nuevo modelo utilizando la técnica de validación cruzada y el método **predict** que es utilizado para detectar las fulguraciones en las curvas de luz.

***Figura 11:*** *La arquitectura del algoritmo Stella CNN. El conjunto de entrenamiento consta de casos "flare" y "non-flare", donde los destellos están en el centro de una sección de 200 cadencias de la curva de luz.*




Fuente: (Feinstein, Montet, Ansdell, et al., 2020)

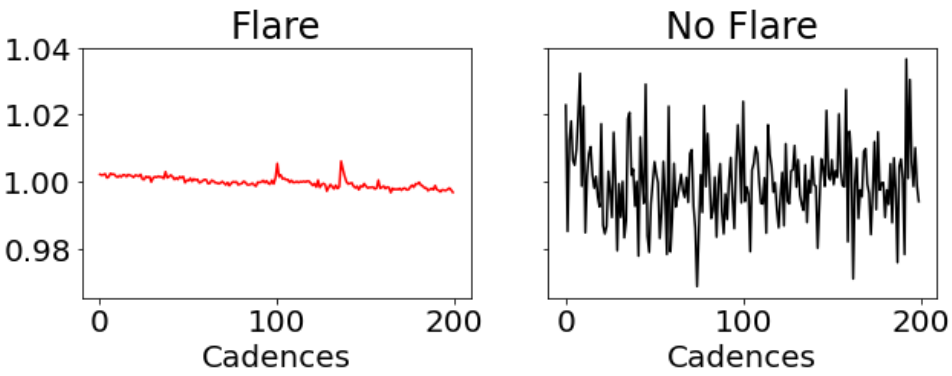
***Figura 12:*** *Arquitectura detallada del algoritmo Stella CNN.*

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv1d (Conv1D) | (None, 200, 16) | 128 |
| max_pooling1d (MaxPooling1D) | (None, 100, 16) | 0 |
| dropout (Dropout) | (None, 100, 16) | 0 |
| conv1d_1 (Conv1D) | (None, 100, 64) | 3136 |
| max_pooling1d_1 (MaxPooling1D) | (None, 50, 64) | 0 |
| dropout_1 (Dropout) | (None, 50, 64) | 0 |
| flatten (Flatten) | (None, 3200) | 0 |
| dense (Dense) | (None, 32) | 102432 |
| dropout_2 (Dropout) | (None, 32) | 0 |
| dense_1 (Dense) | (None, 1) | 33 |

Total params: 105,729
Trainable params: 105,729
Non-trainable params: 0

Fuente: (Feinstein, Montet, Ansdell, et al., 2020)

***Figura 13:*** *En la figura de la izquierda podemos identificar picos abruptos de energía, por lo que esa curva de luz está etiquetada como "Flare" y en la figura de la derecha no vemos ningún pico abrupto, por lo que está etiquetado como "No Flare".*




Fuente: (Feinstein, Montet, Ansdell, et al., 2020)

El módulo **preprocessing_flares** contiene la clase **FlareDataSet** que se encarga de leer y reformatear las curvas de luz.

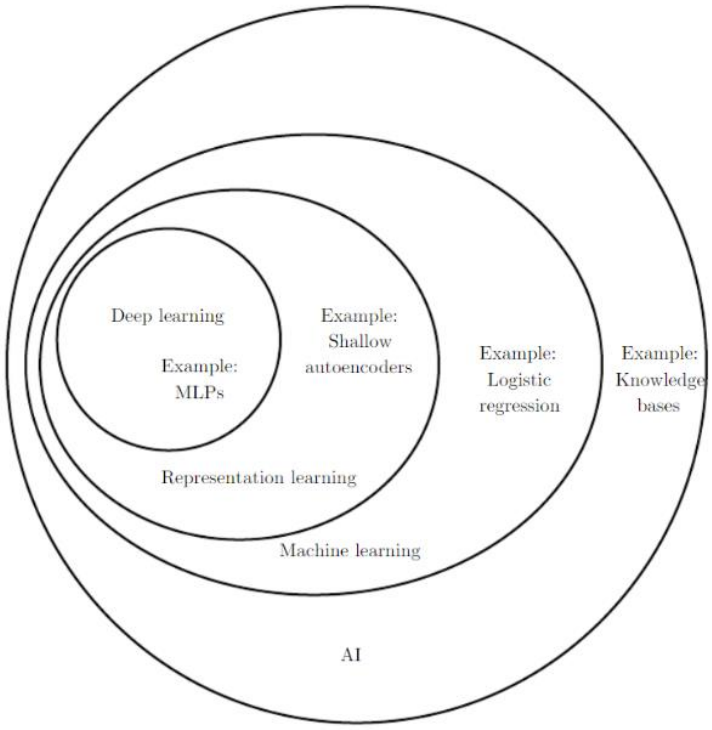
El módulo **visualize** contiene la clase **Visualize** que muestra las métricas definidas en el módulo **metrics.**

## 2.5. CONCEPTOS TÉCNICOS

### 2.5.1. Deep Learning

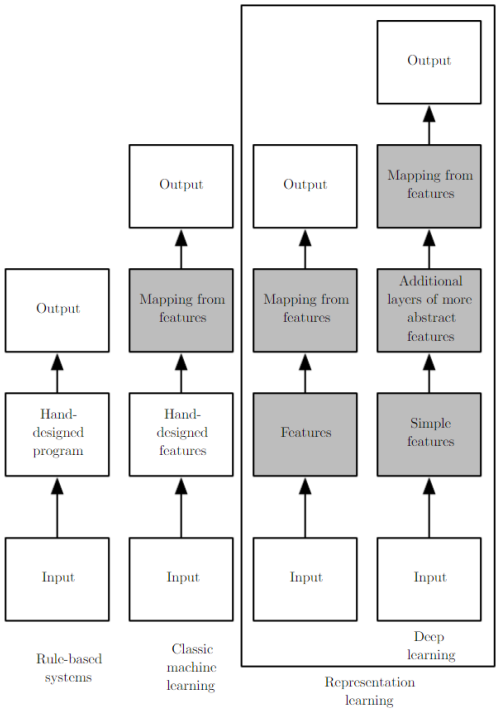
El Deep Learning, en el contexto de la inteligencia artificial, se encuentra dentro de la categoría de Machine Learning, que a su vez es un conjunto de técnicas y métodos más amplio en este campo de estudio.

***Figura 14:*** *En esta figura podemos ver que Deep Learning es incluso un subconjunto de Representation Learning que forma parte de Machine Learning.*




Fuente: (Goodfellow et al., 2016)

***Figura 15:*** *Comparativa entre la estructura de algoritmos basados en reglas, machine learning y los tipos de representation learning.*

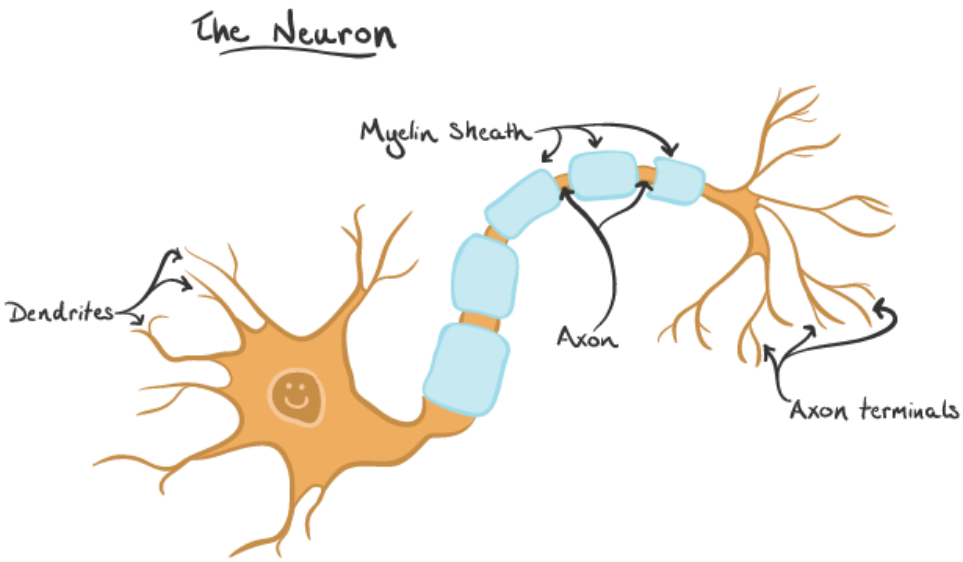



Fuente: (Goodfellow et al., 2016)

En el Deep Learning el perceptrón es uno de los primeros modelos de red neuronal artificial que se inspiró en el cerebro biológico. Un perceptrón es un algoritmo de aprendizaje automático que se inspira en la estructura de las neuronas.

Una neurona biológica a grandes rasgos es una célula del sistema nervioso que se encarga de recibir, procesar y transmitir información. Una neurona biológica consta de diferentes partes, como el cuerpo celular, el axón y las dendritas.

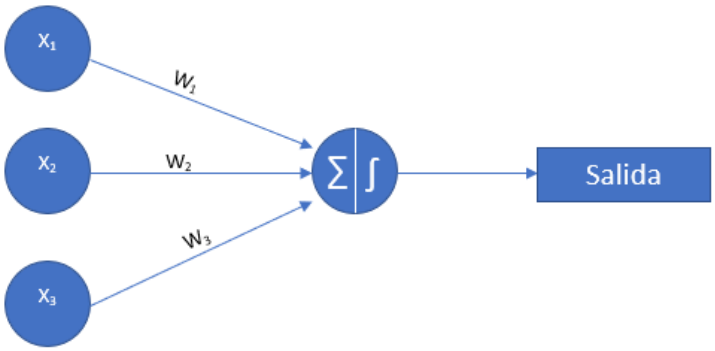
***Figura 16:*** *En esta figura podemos observar la estructura de una neurona biológica, que a grandes rasgos posee una estructura compuesta por 3 principales componentes: dendritas (Son las que reciben la señal de neuronas vecinas), axón (transmite la señal), terminal axónica (transmite la señal a otras neuronas).*




Fuente: (Khan Academy, s. f.)

De la misma forma el perceptrón intenta simular este comportamiento. A grandes rasgos podemos decir que un perceptrón es un modelo lineal que toma una serie de características de entrada y produce una salida.

***Figura 17:*** *En esta figura podemos observar la estructura de un perceptrón simple, que recibe como input* $\boldsymbol{x_i}$ *los conectados por los pesos* $\boldsymbol{w_i}$*, posteriormente en el nodo central llamado cuerpo celular se encarga del procesamiento y produce una salida.*



La siguiente ecuación muestra el funcionamiento del perceptrón simple:

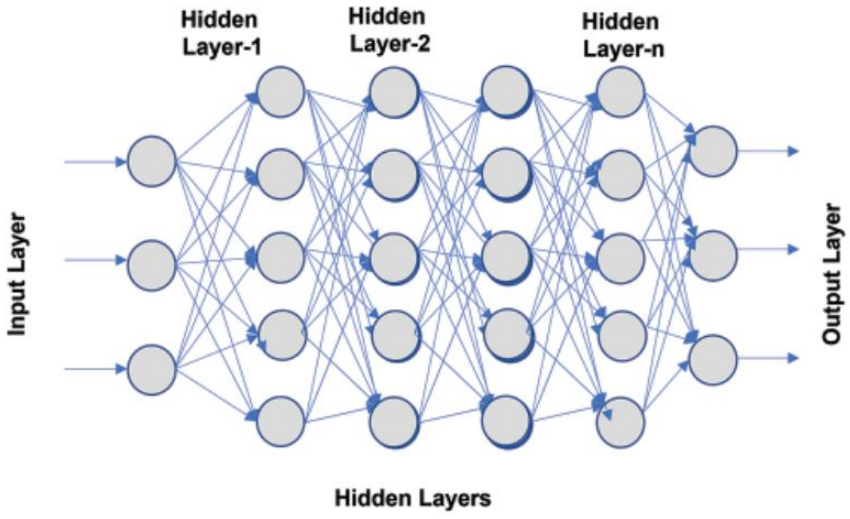
$$salida = \sigma\left(\sum_{i=1}^{n} x_i w_i + b\right)$$

Donde $b$ representa el sesgo, y $\sigma$ una función de activación.

Como es de imaginar un perceptrón simple no podrá resolver todos los problemas que nos encontremos en el día a día. Por lo que debemos buscar una forma de poder utilizar esta estructura y sea capaz de resolver problemas más complejos. Una forma de hacerlo, y no es la única, es usar varias neuronas y concatenarlas secuencialmente en capas.

Un perceptrón multicapa es un tipo de red neuronal artificial que se compone de múltiples capas de perceptrones conectadas entre sí. Un perceptrón multicapa se utiliza para resolver problemas de aprendizaje automático que no pueden ser resueltos por un perceptrón simple. Si bien no hay una definición clara de cuál es el máximo de capas que debe tener un perceptrón multicapa antes de convertirse en una red neuronal profunda. En este trabajo consideraremos que una red neuronal profunda tiene al menos 3 capas.

***Figura 18:*** *En esta figura podemos observar la estructura de una red neuronal profunda.*




Fuente: (Tarek et al., 2022)

Anteriormente mencionamos la función de activación, pero aún no hemos definido qué es ni para qué sirve. La función de activación en una red neuronal sirve para romper la linealidad.

Porque de lo contrario, la concatenación de varias funciones lineales colapsa en una sola función lineal. Por ejemplo:

Si tenemos dos funciones lineales $f(x) = 2x + 3$ y $g(x) = 3x + 1$. Si concatenamos estas dos funciones, obtendremos una nueva función $h(x) = f(g(x)) = f(3x + 1) = 2(3x + 1) + 3 = 6x + 5$. Como se puede ver, la función $h(x)$ es una función lineal. Por lo que la capacidad de abstracción de la red no aumenta según el número de capas.

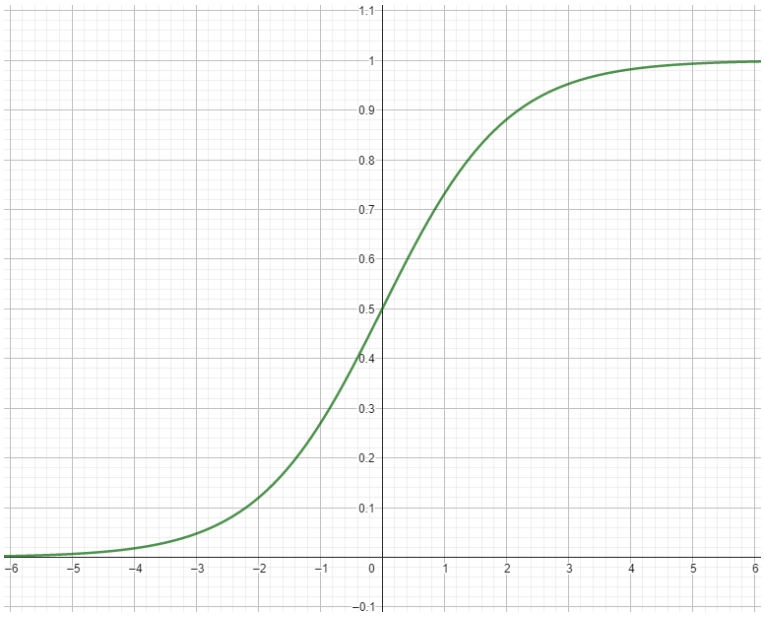
La forma en la que podemos romper la linealidad es con las funciones de activación. A continuación, describiremos las más utilizadas:

- Función de activación sigmoide.

La función sigmoide está definida por la siguiente expresión:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

***Figura 19:*** *En la figura podemos observar que el rango de la función de activación sigmoide es el intervalo entre 0 y 1.*

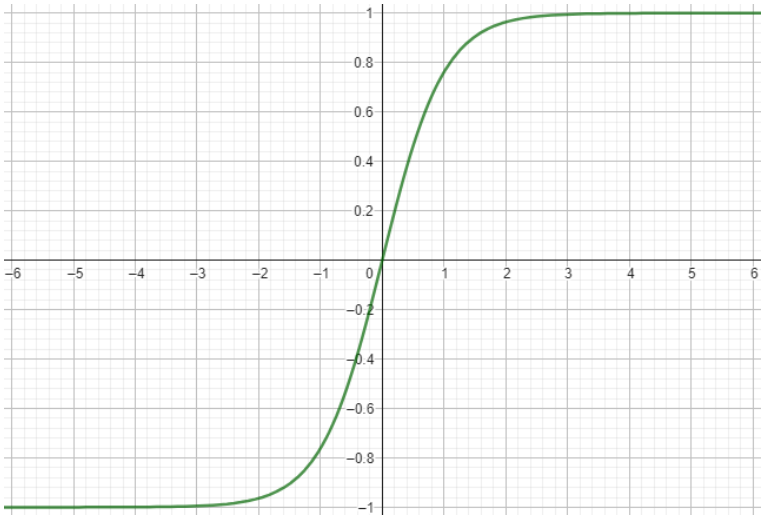



- Función de activación tangente hiperbólica (tanh).

La función tanh está definida por la siguiente expresión:

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

***Figura 20:*** *En esta podemos observar que el rango de la función de activación tangente hiperbólica es el intervalo entre -1 y 1.*

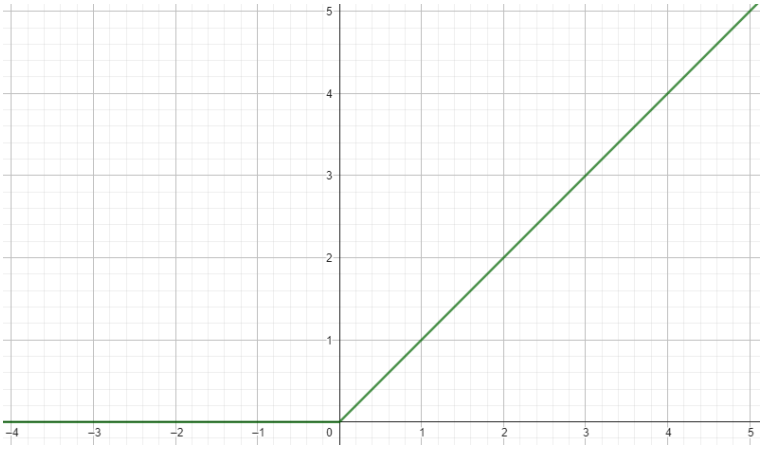



- Función de activación ReLU (rectified linear unit).

La función de activación ReLU está definida por la siguiente expresión:

$$f(x) = x^{+} = \max(0, x)$$

***Figura 21:*** *Esta función de activación devuelve 0 para valores menores o iguales a cero y la identidad para valores positivos.*




Antes del entrenamiento de nuestro modelo debemos definir una función de pérdida, que dependerá del tipo de problema que se desea resolver.

Para problemas de regresión se suele utilizar:

- Error medio absoluto (MAE).

$$MAE = \frac{1}{N}\sum_{i=1}^{n}|\widehat{y_i} - y_i|$$

- Error cuadrático medio (MSE).

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(\widehat{y_i} - y_i)^2$$

Para problemas de clasificación:

- Categorical Cross-Entropy (Si la clasificación es multiclase).

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{n} y_{ij} log(p_{ij})$$

- Binary Cross-Entropy (Si la clasificación es binaria).

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{n} y_i log(\widehat{y_i}) + (1 - y_i) log(1 - \widehat{y_i})$$

### 2.5.2. Tipos de redes neuronales artificiales (RNA)

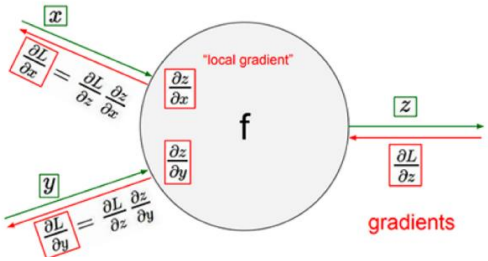
Existen varias formas de clasificar los tipos de redes neuronales artificiales como, por ejemplo: por el número de capas, tipo de conexiones, grado de conexiones, etc. En este trabajo las clasificaremos según el tipo de entrada que recibe y nos centraremos en dos tipos:

- Red neuronal feedforward
- Red neuronal Convolucional

Las redes neuronales artificiales de topología feedforward se definen por su capacidad de realizar el procesamiento de datos en una única dirección. Este tipo de redes se componen de tres capas principales de unidades independientes de cálculo, conocidas como neuronas: la capa de entrada, donde se reciben los datos a procesar; la capa intermedia, donde se lleva a cabo el procesamiento en sí mismo; y la capa de salida, donde se emiten los resultados finales (Vásquez López, 2014). Es una de las topologías de red más usadas y dependiendo de la bibliografía también puede ser llamada como red neuronal clásica.

En este tipo de redes utilizamos función de pérdida para ajustar los pesos de la red mediante el algoritmo de backpropagation, el cual calcula los gradientes locales propagándose desde el output (Rumelhart et al., 1986).

***Figura 22:*** *Los gradientes de las capas posteriores se multiplican por los gradientes locales y son propagados a las capas anteriores.*



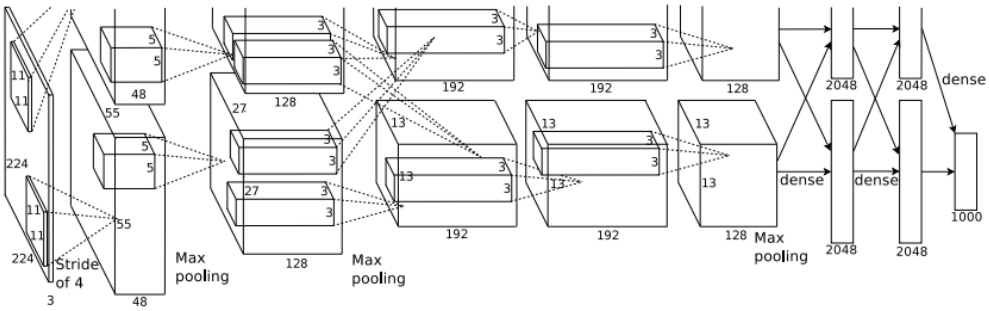

Fuente: (Lorente et al., 2021)

> "*Cuando hablamos de redes neuronales convolucionales, nos referimos a un tipo de red neuronal artificial donde las neuronas corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria de un cerebro biológico*" (KeepCoding, 2020).

Las redes neuronales de este tipo han supuesto una verdadera revolución en el ámbito de la visión por computadora. Su irrupción en el mundo de la investigación y el desarrollo tecnológico se produjo en la conocida competición de ImageNet, una base de datos que se ha convertido en un punto de referencia para evaluar los algoritmos de clasificación y reconocimiento de imágenes. Desde 2010, ImageNet ha organizado cada año el ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2015).

En 2012, Alex Krizhevsky y su tutor de tesis, Geoffrey Hinton, presentaron una nueva arquitectura de red neuronal convolucional (Alexnet), la cual logró reducir significativamente el porcentaje de error en comparación con los modelos previos (Krizhevsky et al., 2012).

***Figura 23:*** *Arquitectura de Alexnet.*



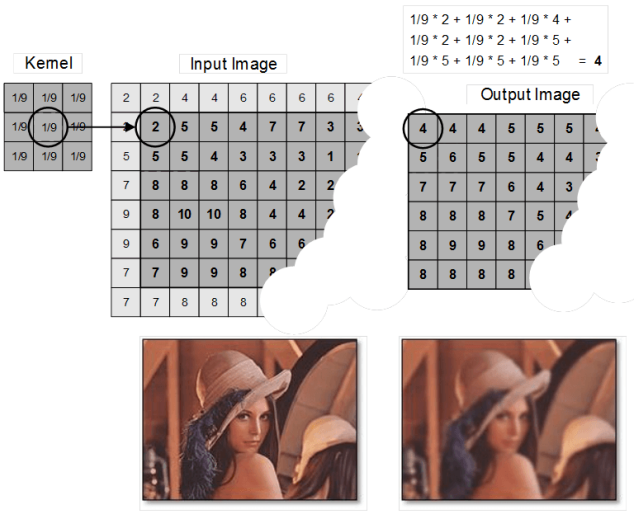

Fuente: (Krizhevsky et al., 2012)

La arquitectura propuesta en este caso es diferente a las redes neuronales feedforward tradicionales.

Una convolución es una técnica matemática utilizada en procesamiento de señales y redes neuronales para combinar dos funciones para producir una tercera función. Esta técnica consiste en un kernel o filtro que se utiliza para deslizar sobre la imagen de entrada y calcular un producto punto entre los valores del kernel en cada posición.

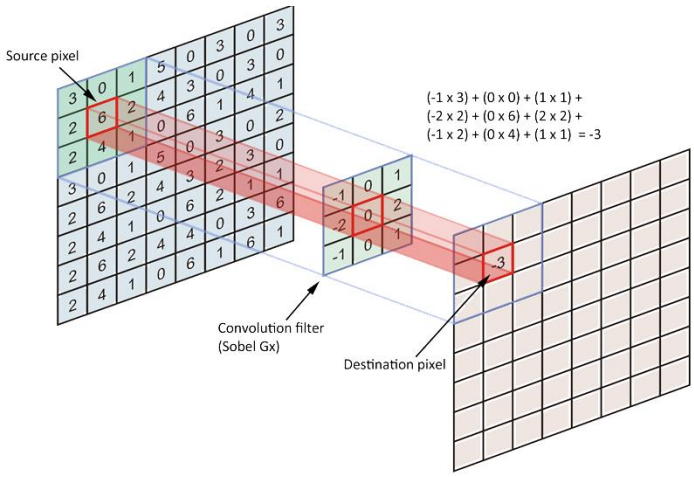
Por ejemplo:

***Figura 24:*** *En esta figura podemos observar una imagen de entrada (inferior izquierda) produce una imagen de salida (inferior derecha) después de aplicar la operación convolución con el kernel de dimensión 3x3 (superior izquierda). Esta convolución produce un efecto de desenfoque en la imagen de salida.*




Fuente: (KeepCoding, 2022)

***Figura 25:*** *En esta imagen podemos observar de mejor forma cómo funciona la convolución.*

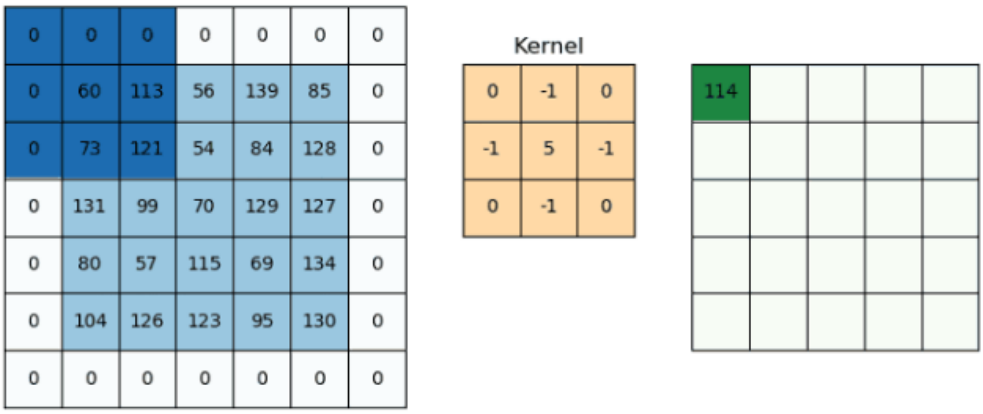



Fuente: (KeepCoding, 2022)

Algunos kernels producen un efecto de desenfoque (como en la Figura 24), otros detectan bordes, frecuencias y formas específicas. En las redes neuronales clásicas, se entrenan los pesos que ponderan las entradas de las neuronas, pero en las Redes Neuronales Convolucionales, se entrenan los valores del kernel.

Un problema habitual en las convoluciones es el tratamiento de los bordes. Como observamos en la figura anterior, se traza una ventana del tamaño del kernel centrado en el pixel en donde aplicaremos la convolución. Lo cual no es posible realizar en los pixeles que están en el borde, lo que involucra pérdida de información y reducción de la imagen resultante. Para solucionar este problema, podemos utilizar la técnica de Padding, una de las técnicas de padding más utilizadas es el zero padding que consta de rellenar de ceros alrededor de nuestra imagen de entrada para poder controlar el tamaño de la imagen de salida (Stanford, 2022).

***Figura 26:*** *En esta figura podemos observar que a la imagen original (5x5) se le ha añadido zero padding de 1 píxel, esto para poder mantener el tamaño de 5x5 píxeles en la imagen de salida.*



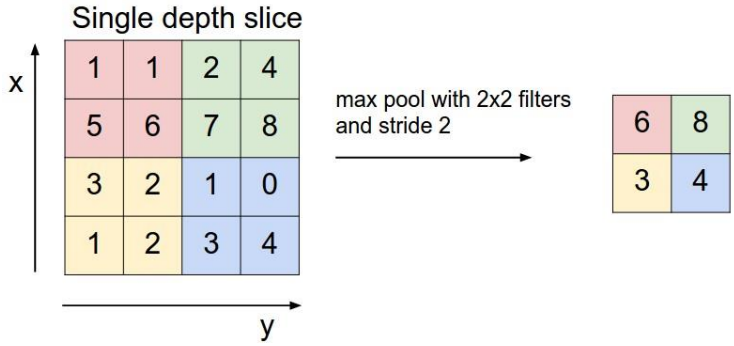

Fuente: (Dharmaraj, 2021)

El stride es un hiper parámetro importante en las operaciones de convolución de una Red Neuronal Convolucional (CNN). Indica la cantidad de píxeles que salta la ventana de la convolución en cada paso. Por defecto, el stride es 1, pero puede ser modificado según las necesidades del problema que se está tratando y si se desea reducir el tamaño de la imagen de salida después de aplicar la convolución.

En una CNN, la profundidad de los mapas de activación puede incrementar exponencialmente en cada capa, lo que resulta en un coste computacional elevado. Aplicar stride ayuda a reducir el tamaño de la imagen, pero a costa de perder información. Para solucionar este problema, se utiliza la técnica de pooling, que es una técnica de reducción de dimensionalidad que permite disminuir la resolución espacial de los mapas de características, manteniendo solo las características más importantes y reduciendo la complejidad computacional de la red.

Existen varios tipos de pooling, como max pooling y average pooling, pero en general, la técnica consiste en dividir el mapa de características en regiones más pequeñas y, a continuación, seleccionar un solo valor de cada región. Por ejemplo, en max pooling se selecciona el valor máximo en cada región. Esto permite mantener las características más fuertes y descartar las menos relevantes.

***Figura 27:*** *En esta figura se está aplicando un max pooling de dimensión 2x2.*




Fuente: (Stanford, 2022)

El uso de pooling en una red neuronal convolucional ayuda a mejorar la capacidad de la red para generalizar, ser resistente a cambios pequeños en los datos de entrada y es considerada un mecanismo de regularización.

### 2.5.3. Evaluación de un modelo entrenado con redes neuronales

Las métricas de evaluación en una red neuronal se establecen dependiendo el problema que se intente resolver. Para mantenernos en el alcance de este trabajo nos centraremos en el marco del aprendizaje supervisado y describiendo las métricas más comunes aplicables a los problemas de regresión y clasificación binaria.

En problemas de regresión normalmente se utilizan las métricas descritas en el Capítulo 2.5.1 sobre las funciones de pérdida en el apartado regresión.

En problemas de clasificación binaria se suele utilizar:

**Confusion Matrix:** Una matriz de confusión es una herramienta de visualización que se utiliza para evaluar el rendimiento de un modelo de clasificación. Muestra la cantidad de veces en que un modelo clasificó correctamente o incorrectamente cada una de las clases. Esto puede ser útil para evaluar el desempeño del modelo, identificar tendencias en los resultados y mejorar el modelo.

**Tabla 1:** *Matriz de confusión binaria.*

| | | *Real Condition* | |
|---|---|---|---|
| | | Positive | Negative |
| *Predict Condition* | Positive | True Positive (TP) | False Positive (FP) |
| | Negative | False Negative (FN) | True Negative (TN) |

**Accuracy:** es una medida de evaluación comúnmente usada en Machine Learning para medir la exactitud de un modelo. Se calcula como la fracción de predicciones correctas que el modelo realiza sobre un conjunto de datos de prueba.

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

**Precision:** se refiere a la capacidad de un modelo para identificar correctamente a los elementos positivos. Se utiliza para medir qué tan bien clasifica el modelo los verdaderos positivos en relación con los falsos positivos. Esto significa que un alto grado de precisión en el modelo significa que hay una proporción más alta de verdaderos positivos que falsos positivos.

$$precision = \frac{TP}{TP + FP}$$

**Recall:** es una métrica de evaluación de desempeño en Machine Learning que mide la exactitud de la clasificación, también conocida como sensibilidad. Mide la proporción de instancias positivas verdaderas que se clasifican correctamente como positivas. Es una métrica importante para evaluar el rendimiento de un modelo de aprendizaje automático, ya que mide la capacidad del modelo para detectar verdaderos positivos.

$$recall = \frac{TP}{TP + FN}$$

**F1 Score:** La métrica F1 es una medida de la precisión y el recall en un problema de clasificación binaria. La métrica F1 combina ambas medidas en una sola y se puede interpretar como una medida de la calidad del equilibrio entre la precisión y el recall.

En un problema de clasificación binaria, una métrica F1 alta indica que la precisión y el recall son ambos altos, lo que significa que el modelo es efectivo en términos de identificar tanto los verdaderos positivos como los verdaderos negativos. Por lo tanto, una métrica F1 alta es un indicador de un buen modelo de clasificación.

$$F1 = \frac{2 \cdot precision \cdot recall}{recall + precision} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

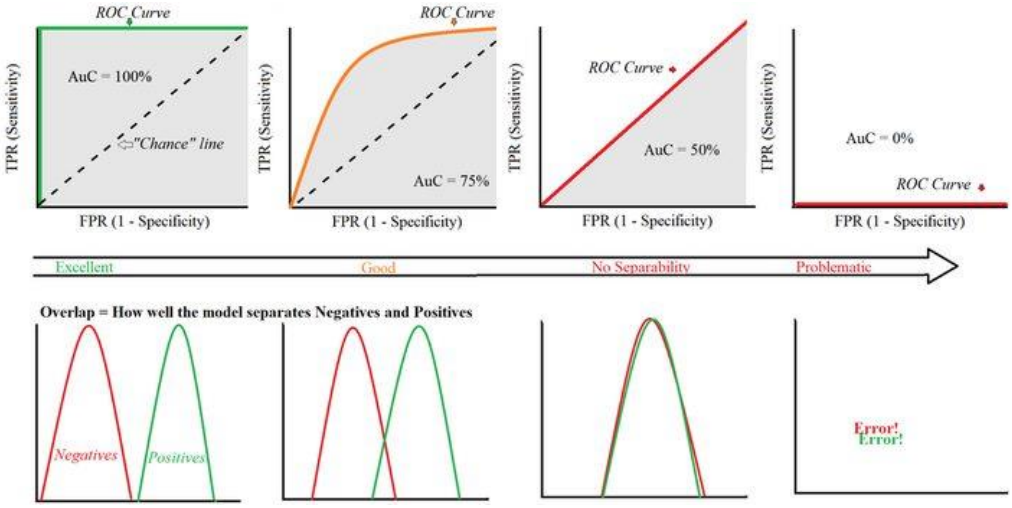
**Curvas ROC, AUC:** ROC significa curvas Receiver Operating Characteristic también conocida como Area Under the Curve (AUC) es una representación gráfica que se utiliza para evaluar el rendimiento de un modelo de clasificación binaria. La curva ROC se traza tomando como eje x la tasa de falsos positivos (1 - especificidad) y como eje y la tasa de verdaderos positivos (sensibilidad).

$$sensibilidad = recall = \frac{TP}{TP + FN}$$

$$especificidad = \frac{TN}{TN + FP}$$

La tasa de verdaderos positivos (también conocida como sensibilidad) se refiere a la proporción de ejemplos positivos que se clasificaron correctamente como positivos. Por otro lado, la tasa de falsos positivos se refiere a la proporción de ejemplos negativos que se clasificaron incorrectamente como positivos.

***Figura 28:*** *En esta figura podemos observar cómo se clasifica el modelo dependiendo del área bajo la curva ROC.*




Fuente: (Glen, 2019)

La interpretación de la curva ROC depende del contexto, pero en general, se interpreta de la siguiente manera:

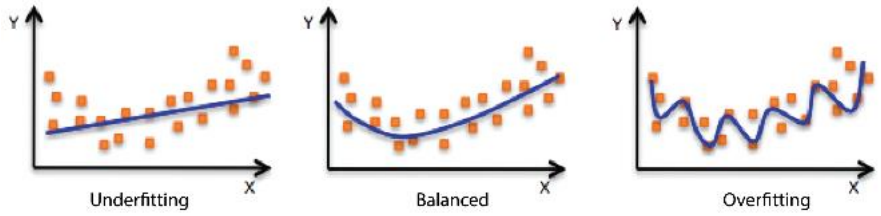
- Un modelo que no discrimina en absoluto entre las dos clases producirá una curva ROC que sea una línea recta con un ángulo de 45° y un área bajo la curva (AUC) de 0,5.
- Un modelo perfecto producirá una curva ROC que sea una línea recta que toca el extremo superior izquierdo del gráfico, con un AUC de 1.
- Un modelo peor que el azar producirá una curva ROC que esté por debajo de la línea recta de 45°, con un AUC menor a 0,5.
- Un modelo con una curva ROC que está más cerca del extremo superior izquierdo es mejor que un modelo con una curva más cercana a la línea recta de 45°.

Otras métricas generales para evaluar el rendimiento del modelo son la visualización del rendimiento del entrenamiento, en el conjunto de validación y en el conjunto de test.

El rendimiento del entrenamiento y el conjunto de validación nos permite identificar si el modelo presenta sobreajuste (Overfit). “El sobreajuste es un comportamiento de aprendizaje automático no deseado que se produce cuando el modelo de aprendizaje automático proporciona predicciones precisas para los datos de entrenamiento, pero no para los datos nuevos” (Amazon Web Services [AWS], s. f.-b)

Como resultado, el modelo se sobre optimiza para los datos de entrenamiento y pierde su capacidad para generalizar y hacer predicciones precisas en datos nuevos o desconocidos.

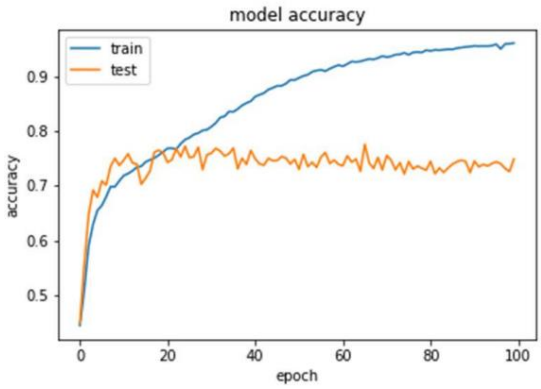
***Figura 29:*** *En esta figura podemos observar 3 modelos distintos para el mismo conjunto de datos.*




Fuente: (AWS, s. f.-a)

El overfitting se puede identificar por una mejora en las métricas de entrenamiento, pero una peor performance en el conjunto de validación/prueba.

***Figura 30:*** *Esta figura muestra que el accuracy del modelo aumenta en el conjunto de entrenamiento, pero disminuye en el conjunto de prueba, lo que indica un sobreajuste y falta de capacidad para generalizar a nuevos datos.*




Fuente: (Julin, 2019)

Por lo tanto, para evaluar el rendimiento de un modelo, debemos monitorear métricas de entrenamiento y validación, y luego evaluar en un conjunto de prueba distinto.

# 3. Objetivos y metodología de trabajo

El objetivo global de este proyecto es realizar un trabajo de investigación, análisis y desarrollo de modelos de inteligencia artificial basados en redes neuronales para la clasificación de estrellas fulgurantes en curvas de luz proporcionadas por la misión TESS. Estos modelos sentarán las bases, junto con otras técnicas como análisis de tránsito, para la caracterización de las atmósferas de los exoplanetas orbitantes.

## 3.1. Objetivo general

Optimizar el algoritmo Stella para clasificar fulguraciones estelares en curvas de luz proporcionadas por la misión TESS utilizando nuevas arquitecturas de redes neuronales obteniendo una mejora en la clasificación de verdaderos positivos con un menor coste computacional.

## 3.2. Objetivos específicos

- Identificar y explorar las principales herramientas utilizadas para clasificar fulguraciones estelares en curvas de luz y el tipo de técnicas que emplean.
- Analizar y profundizar en el entendimiento del algoritmo Stella.
- Modificar y Entrenar el algoritmo Stella con otras arquitecturas de redes neuronales: Inception, ResNet, basados en convoluciones dilatadas, basadas en redes convolucionales temporales (TCN) y Redes Neuronales convolucionales con mecanismos de atención.
- Evaluar el modelo utilizando las métricas: Accuracy, precisión, recall, F1-score, ROC-AUC y coste computacional.

## 3.3. Metodología del trabajo

EL proceso del presente trabajo se llevó a cabo en 4 fases:

Comprensión del algoritmo Stella y su funcionamiento: Esta fase se llevó a cabo mediante la instalación del módulo Stella para entender su funcionamiento a nivel usuario,

posteriormente se clonó el repositorio de github para ahondar su funcionamiento a nivel código y la solución de algunos bugs ocasionados por la actualización de algunas dependencias.

Revisión del estado del arte en la clasificación automática de fulguraciones estelares en curvas de luz: la revisión del estado del arte en la clasificación automática de fulguraciones estelares en curvas de luz nos brinda una comprensión del problema que Stella busca resolver, enfatizando la importancia de contar con un sistema automático para clasificar estas fulguraciones estelares.

Implementación y evaluación de arquitecturas conocidas: la implementación y evaluación de las arquitecturas de redes neuronales conocidas como Inception en sus cuatro versiones y ResNet en sus dos versiones permitió ampliar nuestro conocimiento sobre la evolución histórica de las redes neuronales convolucionales. La evaluación de estas arquitecturas permitió identificar sus fortalezas y debilidades, y especialmente la aplicabilidad de redes tan complejas en la tarea de clasificación de fulguraciones estelares en curvas de luz.

Desarrollo de Nuevas Arquitecturas: Durante esta fase, nos enfocamos en la investigación de soluciones innovadoras para abordar el problema de la clasificación de fulguraciones estelares en curvas de luz. La implementación de capas especializadas para ampliar el campo receptivo de las convoluciones, aprovechar el componente temporal presente en las curvas de luz y mecanismos de atención que permiten a la red generalizar de manera más efectiva, son algunos de los aspectos que se abordaron en esta fase. Estos esfuerzos permitieron obtener mejoras significativas en las métricas de evaluación de la red con un menor coste computacional.

# 4. Planteamiento de la comparativa

La clasificación de fulguraciones estelares es objeto de estudio debido a su contribución en la optimización de la búsqueda y caracterización de exoplanetas. Esta tarea se lleva a cabo mediante diferentes enfoques y herramientas, incluyendo técnicas estadísticas y transformaciones matemáticas que permiten suavizar las curvas de luz y detectar valores atípicos.

Dentro del dominio de la inteligencia artificial hay 4 principales trabajos publicados para la detección de fulguraciones estelares. El modelo de Vida & Roettenbacher (2018) entrenado con el algoritmo RANSAC para la detección de outliers en curvas de luz proporcionadas por la misión Kepler; el modelo de Vida et al. (2021) que está entrenado con una arquitectura de redes neuronales recurrentes aplicado en curvas de luz proporcionadas por las misiones Kepler y TESS; el modelo entrenado por Tu et al. (2022) entrenado con una arquitectura basada redes neuronales convolucionales 2D aplicado a los píxeles de luz proporcionados por la misión TESS; y el algoritmo Stella por Feinstein, Montet, & Ansdel (2020) entrenado con una arquitectura basada en redes neuronales convolucionales 1D aplicados a curvas de luz proporcionados por la misión TESS.

En esta comparativa se plantea clasificar fulguraciones estelares en curvas de luz mediante redes neuronales convolucionales 1D por lo que nuestro punto de referencia será el algoritmo Stella.

El algoritmo Stella, no solamente consta de la arquitectura de la CNN. Este algoritmo posee un ecosistema de descarga, preprocesamiento, visualización, etc. Por lo que será nuestra base para el entrenamiento de nuevos modelos con arquitecturas distintas a las propuestas por Feinstein, Montet, & Ansdel (2020).

Se plantea evaluar 11 arquitecturas distintas, enumeradas a continuación:

- Inceptionv1 1D
- Inceptionv2 1D
- Inceptionv3 1D

- Inceptionv4 1D
- ResNetv1 1D
- ResNetv2 1D
- Arquitectura con convoluciones dilatadas
- Arquitectura con convoluciones dilatadas + capa de atención
- Arquitectura con convoluciones dilatadas + capa multihead de atención
- TCN
- TCN + capa de atención

Para evaluar cada modelo se utilizarán distintas métricas como:

- Matriz de confusión
- Accuracy
- Precision
- Recall
- F1-score
- Curva ROC
- Cantidad de parámetros

Se evaluarán 3 estados del modelo:

- Entrenamiento
- Validación
- Test

Para la implementación del primer experimento recurrimos al código de las 4 versiones de Inception y las dos versiones de ResNet adaptados para el tratamiento de señales unidimensionales desarrollado por Mahmud[1]

[1] https://github.com/Sakib1263/Inception-InceptionResNet-SEInception-SEInceptionResNet-1D-2D-Tensorflow-Keras

## 4.1. ARQUITECTURAS EXPERIMENTO 1

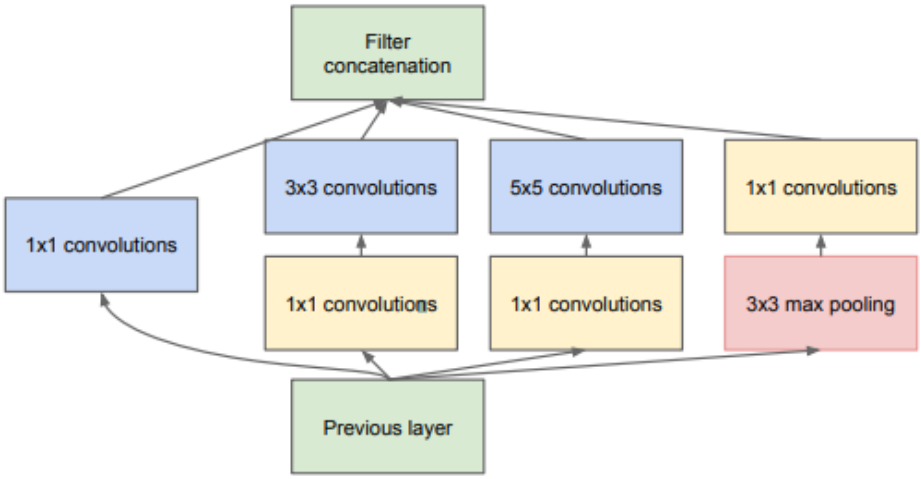
En este apartado, analizaremos en detalle las arquitecturas de redes neuronales mencionadas previamente, describiendo su estructura y funcionamiento.

### 4.1.1. Inceptionv1

GoogLeNet también conocida como Inception, el nombre "Inception" hace referencia a esta idea de tener varios puntos de comienzo o de inicio en el proceso de extracción de características, lo que se traduce en una red más profunda y más compleja que puede capturar características más precisas y complejas. Esto es una parte importante de la arquitectura de GoogLeNet, y es por eso que se le dio este nombre.

GoogLeNet está construida con una estructura modular, donde cada módulo Inception combina diferentes tipos de capas y operaciones para extraer diferentes características de la imagen de entrada. Estos módulos se organizan en varios bloques de Inception, que a su vez están conectados en una estructura jerárquica. Además, GoogLeNet utiliza una arquitectura de subredes donde cada subred se enfoca en extraer características a diferentes escalas de la imagen. La salida de todas las subredes se combina y se pasa a una capa final de clasificación para producir la predicción.

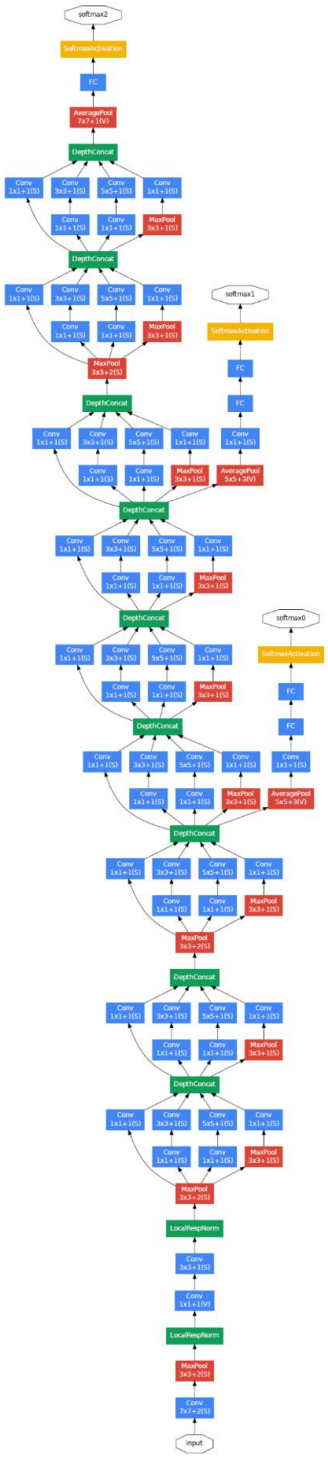
***Figura 31:*** *Módulo Inception con reducción de dimensiones.*




Fuente: (Szegedy et al., 2014)

Esta arquitectura encadena 9 módulos Inception, dentro de cada módulo Inception se producen convoluciones en paralelo que al final se unen en una capa de concatenación.

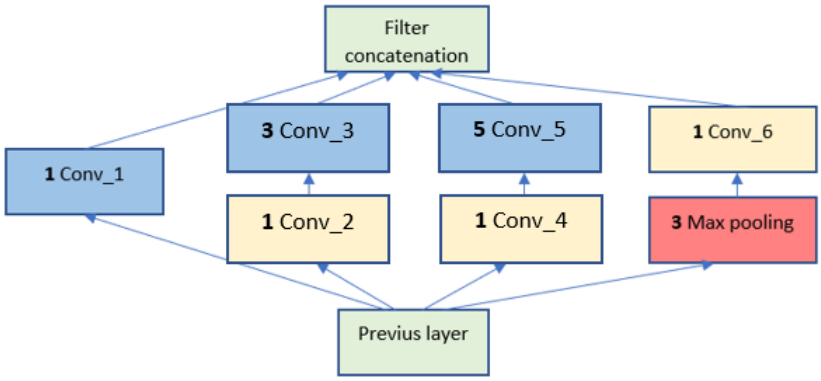
***Figura 32:*** *Arquitectura GoogLeNet.*




Fuente: (Szegedy et al., 2014)

En nuestra comparativa, recrearemos el módulo Inception y la arquitectura GoogLeNet adaptado a señales unidimensionales en lugar de imágenes.

***Figura 33:*** *Arquitectura del módulo Inception 1D.*




Cada módulo Inception tiene una configuración distinta de hiper parámetros. El tamaño de los kernels se mantiene, solo cambiando la cantidad de los mismos en cada módulo. Descritas en la Tabla 2.

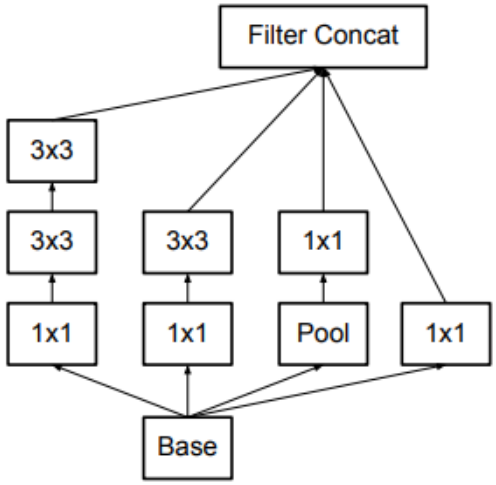
**Tabla 2:** *Número de kernels en las capas de convolución de los módulos Inception.*

| *Inception module* | *Conv_1* | *Conv_2* | *Conv_3* | *Conv_4* | *Conv_5* | *Conv_6* |
|---|---|---|---|---|---|---|
| Inception bloque 1 | 64 | 96 | 128 | 16 | 32 | 32 |
| Inception bloque 2 | 128 | 128 | 192 | 32 | 96 | 64 |
| Inception bloque 3 | 192 | 96 | 208 | 16 | 48 | 64 |
| Inception bloque 4 | 160 | 112 | 224 | 24 | 64 | 64 |
| Inception bloque 5 | 128 | 128 | 256 | 24 | 64 | 64 |
| Inception bloque 6 | 112 | 144 | 288 | 32 | 64 | 64 |
| Inception bloque 7 | 256 | 160 | 320 | 32 | 128 | 128 |
| Inception bloque 8 | 256 | 160 | 320 | 32 | 128 | 128 |
| Inception bloque 9 | 384 | 192 | 348 | 48 | 128 | 128 |

### 4.1.2. Inceptionv2

Para la versión posterior de Inception, hemos realizado modificaciones en el módulo Inception y se ha agregado un bloque de reducción a la red. La modificación más significativa es que se ha reemplazado una convolución con un kernel de tamaño 5x5 en una de las ramas, por dos convoluciones de tamaño 3x3. Esta actualización ayuda a optimizar la red al reducir la cantidad de parámetros que necesitan ser entrenados.

***Figura 34:*** *Módulo Inceptionv2.*




Fuente: (Szegedy et al., 2015)

Nuestra adaptación al procesamiento de señales unidimensionales estaría definida por:

***Figura 35:*** *Arquitectura del módulo Inceptionv2 1D.*

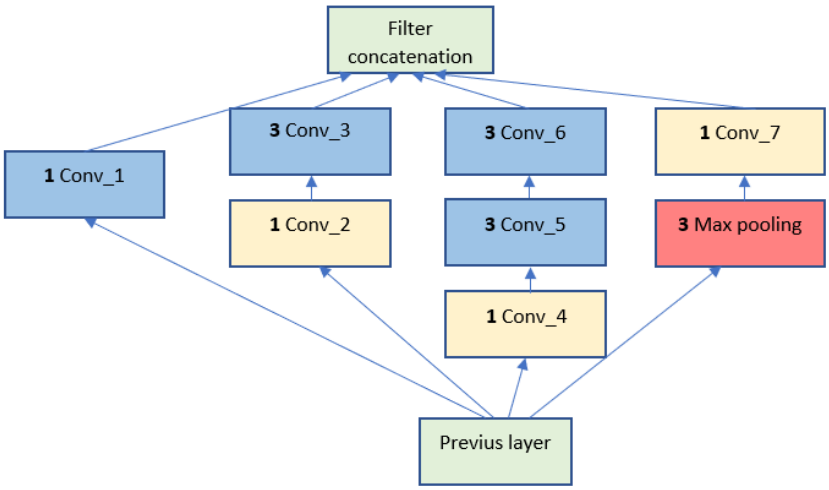
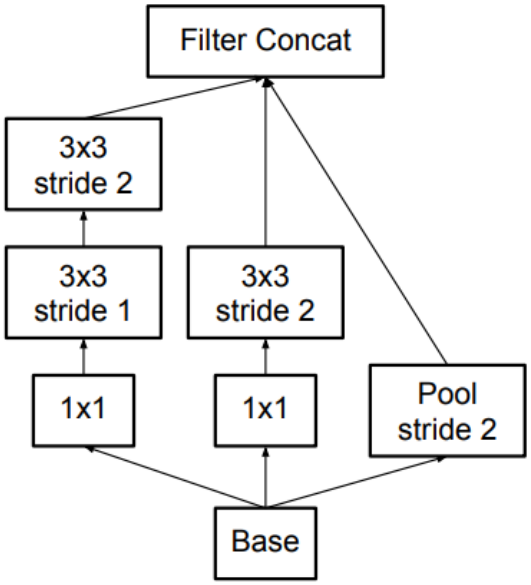


**Tabla 3:** *Número de kernels en las capas de convolución de los módulos Inception.*

| *Inception module* | *Conv_1* | *Conv_2* | *Conv_3* | *Conv_4* | *Conv_5* | *Conv_6* | *conv_7* |
|---|---|---|---|---|---|---|---|
| Inception bloque 1 | 64 | 64 | 64 | 64 | 96 | 96 | 32 |
| Inception bloque 2 | 64 | 64 | 96 | 64 | 96 | 96 | 64 |
| Inception bloque 3 | 224 | 64 | 96 | 96 | 128 | 128 | 128 |
| Inception bloque 4 | 196 | 96 | 128 | 96 | 128 | 128 | 128 |
| Inception bloque 5 | 160 | 128 | 160 | 128 | 160 | 160 | 96 |
| Inception bloque 6 | 96 | 128 | 196 | 160 | 192 | 192 | 96 |
| Inception bloque 7 | 352 | 192 | 320 | 160 | 224 | 224 | 128 |
| Inception bloque 8 | 352 | 192 | 320 | 192 | 224 | 224 | 128 |

El bloque de reducción (Reduction block) en la arquitectura Inception tiene como objetivo reducir el tamaño espacial de los mapas de características y disminuir la cantidad de parámetros en la red. Esto se logra mediante la aplicación de operaciones de agrupamiento y convoluciones con filtros de tamaño reducido en paralelo, lo que permite reducir la dimensión espacial de los mapas de características mientras se mantiene la información relevante.

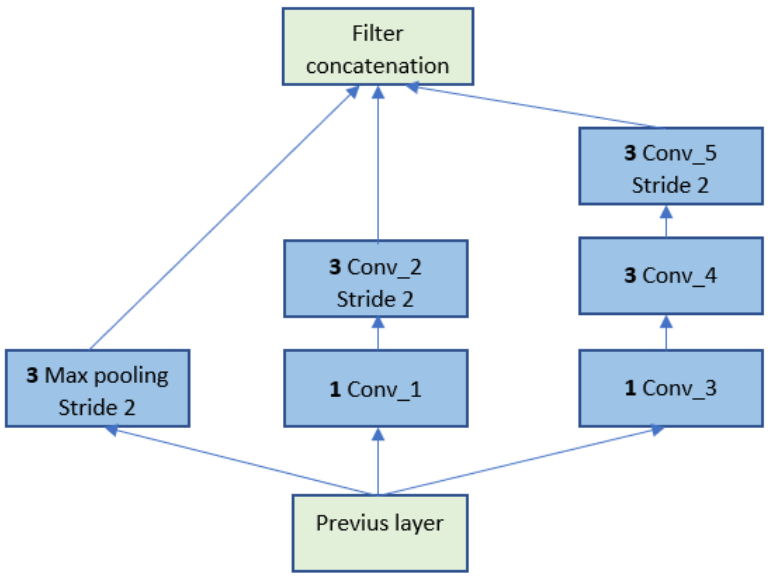
***Figura 36:*** *Reduction block.*




Fuente: (Szegedy et al., 2015)

La Figura 37 muestra la adaptación del Reduction Block en señales unidimensionales

***Figura 37:*** *Reduction block 1d InceptionV2.*




La implementación de InceptionV2 contiene 2 módulos de Reduction Block, la cantidad de kernels utilizados están descritos en la Tabla 4.

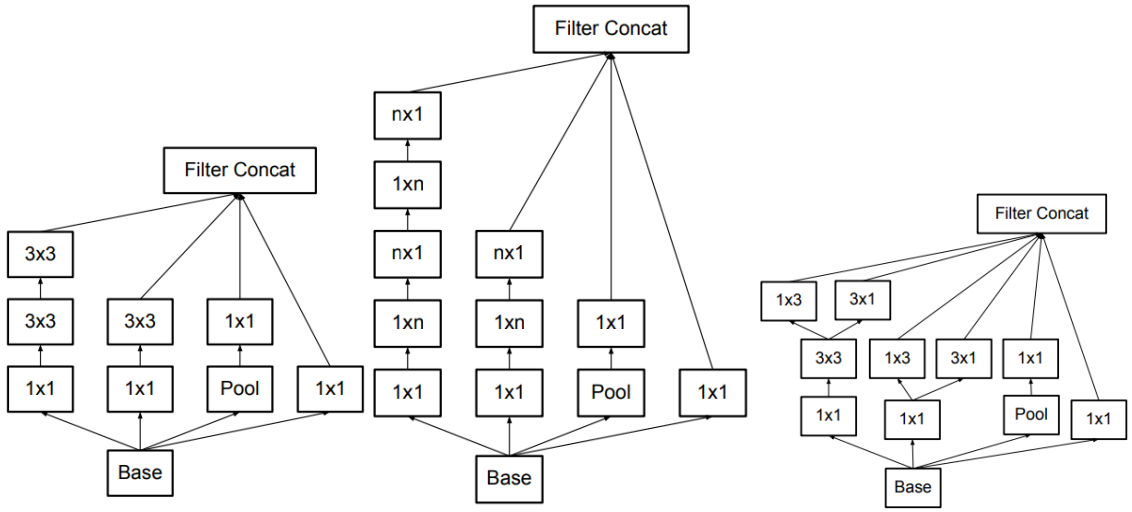
**Tabla 4:** *Número de kernels en las capas de convolución en los módulos Reduction* Block.

| *Reduction Block* | *Conv_1* | *Conv_2* | *Conv_3* | *Conv_4* | *Conv_5* |
|---|---|---|---|---|---|
| Reduction Block 1 | 128 | 160 | 64 | 96 | 96 |
| Reduction Block 2 | 128 | 192 | 192 | 256 | 256 |

### 4.1.3. Inceptionv3

Inceptionv3 contiene 3 módulos distintos de Inception, debido a que esta arquitectura está pensada para solucionar problemas bidimensionales. Algunos componentes de la arquitectura no podrán ser adaptados a una señal unidimensional.

***Figura 38:*** *Bloque Inception_A (izquierda), bloque Inception_B (Centro), bloque Inception_C (derecha).*




Fuente: (Szegedy et al., 2015)

Como se muestra en la figura 38, no es posible aplicar la mayoría de las últimas capas de los módulos Inception_B e Inception_C, debido a la imposibilidad de transponer el tamaño del kernel en una señal unidimensional. Debido a la semejanza morfológica de estos tres módulos, su aplicación a señales unidimensionales será el mismo módulo, con diferencias en el tamaño del kernel en algunas capas.

***Figura 39:*** *en esta figura observamos la adaptación a procesamiento de señales unidimensionales de los módulos Inception_A (izquierda) e Inception_B (derecha).*

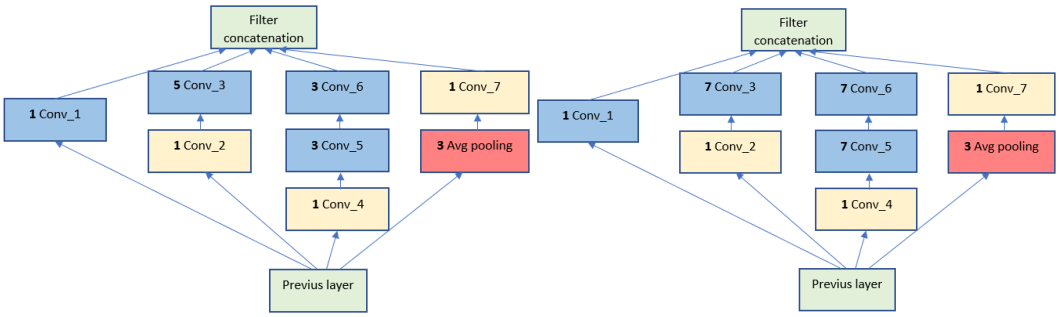
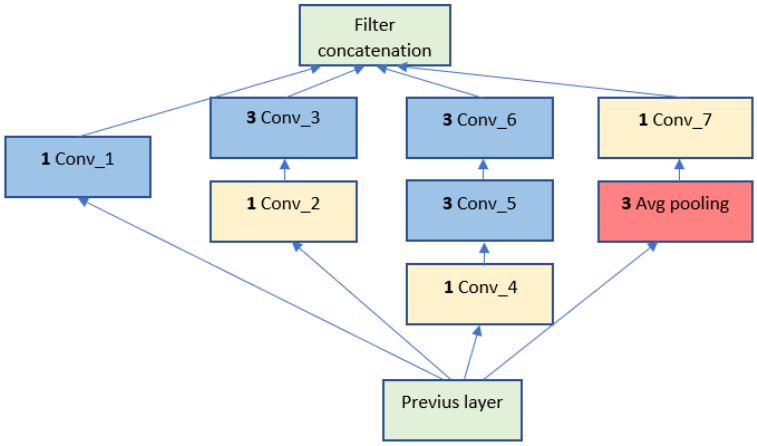



***Figura 40:*** *En esta figura observamos la adaptación a procesamiento de señales unidimensionales del módulo Inception_C.*




El bloque Inception_A 1D (Figura 39) mantiene la estructura del bloque Inception_A 2D (Figura 38), pero se pueden observar cambios significativos en los bloques Inception_B e Inception_C.

En el bloque Inception_B 2D de la figura 38, la rama izquierda presenta una secuencia de 4 convoluciones consecutivas 1xn → nx1 → 1xn → nx1. Sin embargo, en nuestra implementación, hemos reemplazado estas 4 convoluciones por solo 2 convoluciones. Por otro lado, en la segunda rama de izquierda a derecha, hemos sustituido las dos convoluciones 1xn → nx1 por una sola convolución. En ambos casos, el tamaño de kernel utilizado es de n=7.

En el bloque Inception_C 2D de la figura 38, las ramas 1 y 2, de izquierda a derecha, presentan dos convoluciones en paralelo 1x3 y 3x1. En nuestra implementación, hemos reemplazado estas dos convoluciones por una sola convolución con un tamaño de kernel de 3.

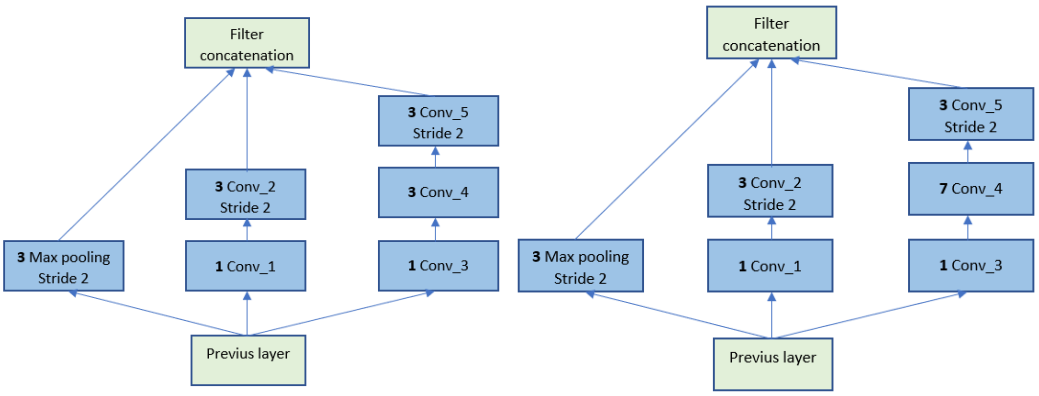
La cantidad de kernels utilizados en cada capa de convolución se detallan a continuación:

**Tabla 5:** *Número de kernels de convolución en los módulos Inception.*

| *Inception module* | *Conv_1* | *Conv_2* | *Conv_3* | *Conv_4* | *Conv_5* | *Conv_6* | *conv_7* |
|---|---|---|---|---|---|---|---|
| Inception_A 1 | 64 | 48 | 64 | 64 | 96 | 96 | 32 |
| Inception_A 2 | 64 | 48 | 64 | 64 | 96 | 96 | 64 |
| Inception_A 3 | 64 | 48 | 64 | 64 | 96 | 96 | 64 |
| Inception_B 1 | 192 | 128 | 192 | 128 | 128 | 192 | 192 |
| Inception_B 2 | 192 | 160 | 192 | 160 | 160 | 192 | 192 |
| Inception_B 3 | 192 | 160 | 192 | 160 | 160 | 192 | 192 |
| Inception_B 4 | 192 | 192 | 192 | 192 | 192 | 192 | 192 |
| Inception_C 1 | 320 | 384 | 384 | 448 | 384 | 384 | 192 |
| Inception_C 2 | 320 | 384 | 384 | 448 | 384 | 384 | 192 |

Dentro de la arquitectura InceptionV3, se han añadido dos bloques de reducción que cumplen la función de disminuir el tamaño espacial de los mapas de características, logrando de esta forma una optimización de la red.

***Figura 41:*** *Reduction_Block_A (izquierda) y Reduction_Block_A (derecha).*



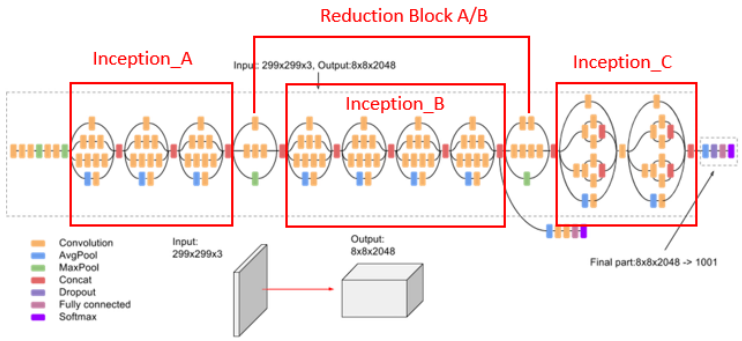

La cantidad kernels en cada capa de convolución de los bloques de reducción se detallan en la Tabla 6.

**Tabla 6:** *Número de kernels en las capas de convolución en los módulos Reduction Block.*

| *Reduction_Block* | *Conv_1* | *Conv_2* | *Conv_3* | *Conv_4* | *Conv_5* |
|---|---|---|---|---|---|
| Reduction_Block_A | 64 | 384 | 64 | 96 | 96 |
| Reduction_Block_B | 192 | 320 | 192 | 192 | 192 |

La estructura final de la arquitectura estaría compuesta de la siguiente forma (Puede diferir por las adaptaciones al tratamiento de señales unidimensionales).

***Figura 42:*** *Arquitectura original inceptionV3 para el tratamiento de imágenes.*

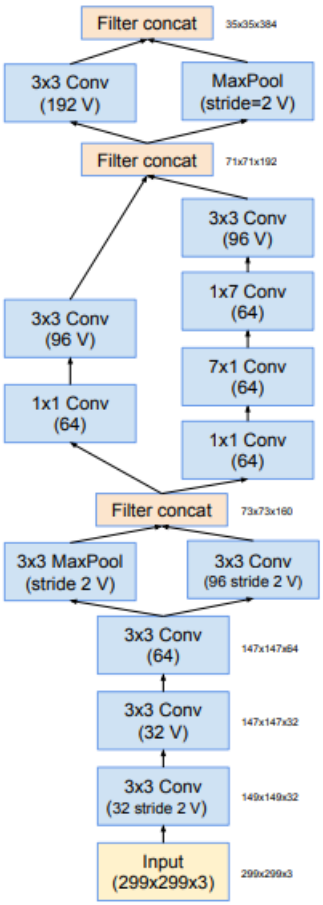



Fuente: (Google Cloud, s. f.)

### 4.1.4. InceptionV4

La principal diferencia entre InceptionV4 y su predecesor InceptionV3 radica en la base de la arquitectura, desde la entrada hasta el primer módulo de Inception. Además, otra diferencia notable es la cantidad de módulos de cada tipo que utiliza la arquitectura.

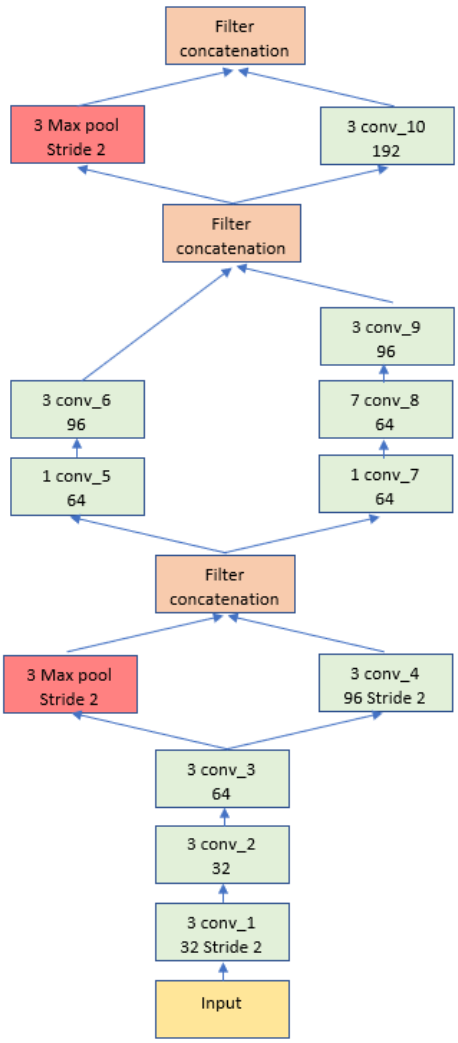
***Figura 43:*** *Base de la arquitectura InceptionV4.*




Fuente: (Szegedy et al., 2016)

Es importante destacar que la fiel adaptación al procesamiento de señales unidimensionales no es posible, ya que como se puede ver en la sección intermedia de la rama derecha antes de la segunda concatenación, el tamaño del kernel está transpuesto, lo cual no es aplicable a señales unidimensionales.

***Figura 44:*** *Adaptación de la base de la arquitectura InceptionV4 a señales unidimensionales.*




La configuración de kernels en los módulos Inception están descritos en la Tabla 7.

**Tabla 7:** *Número de kernels en las capas de convolución en los módulos Inception.*

| *Inception module* | *Conv_1* | *Conv_2* | *Conv_3* | *Conv_4* | *Conv_5* | *Conv_6* | *conv_7* |
|---|---|---|---|---|---|---|---|
| 4 x Inception_A | 96 | 64 | 96 | 64 | 96 | 96 | 96 |
| 7 x Inception_B | 384 | 192 | 256 | 192 | 224 | 256 | 128 |
| 3 x Inception_C | 256 | 384 | 512 | 384 | 512 | 512 | 256 |

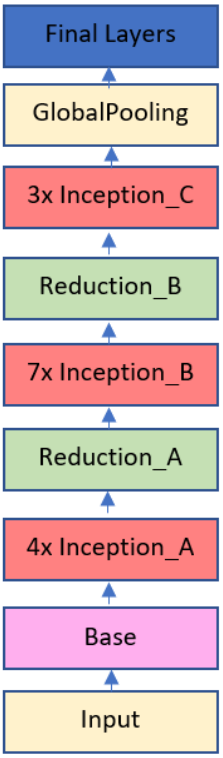
La cantidad kernels en cada capa de convolución se detallan en la Tabla 8.

**Tabla 8:** *Número de kernels en las capas de convolución en los módulos Reduction Block.*

| *Reduction_Block* | *Conv_1* | *Conv_2* | *Conv_3* | *Conv_4* | *Conv_5* |
|---|---|---|---|---|---|
| Reduction_Block_A | 64 | 384 | 192 | 224 | 256 |
| Reduction_Block_B | 192 | 192 | 256 | 320 | 320 |

La arquitectura InceptionV4 1d está representada en el siguiente diagrama:

***Figura 45:*** *Arquitectura InceptionV4 1D.*

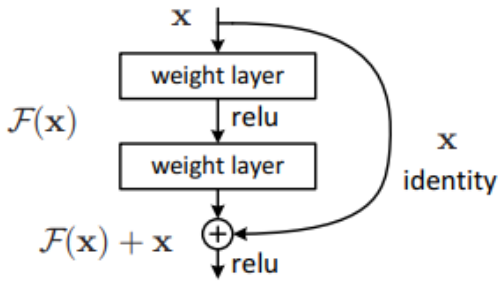



### 4.1.5. ResNetV1

La arquitectura ResNetV1 cuenta con 3 distintos módulos Inception modificados con una conexión residual. La base de la arquitectura es similar a InceptionV3.

La conexión residual funciona conectando la entrada directamente a la salida a través de una secuencia de operaciones adicionales. En lugar de tener una secuencia compleja de capas, la conexión residual permite que la información original pase directamente a la salida, facilitando la propagación del gradiente y mejorando la capacidad de la red para aprender características complejas (He et al., 2016).

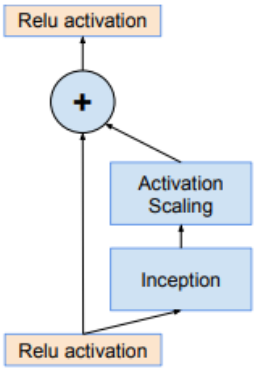
***Figura 46:*** *Aprendizaje residual en un bloque.*




Fuente: (He et al., 2016)

En la arquitectura ResNet la conexión residual sucede en cada bloque Inception.

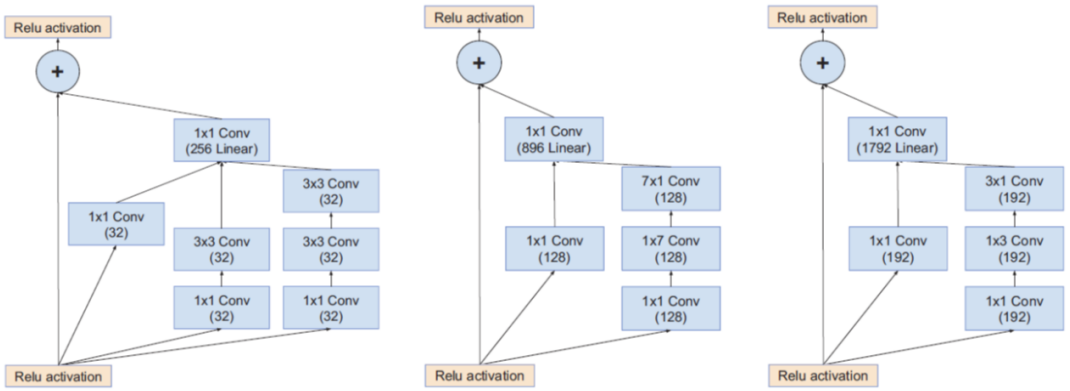
***Figura 47:*** *Conexión residual en módulo Inception.*




Fuente: (Szegedy et al., 2016)

Los módulos Inception modificados con conexiones residuales están descritas a continuación:

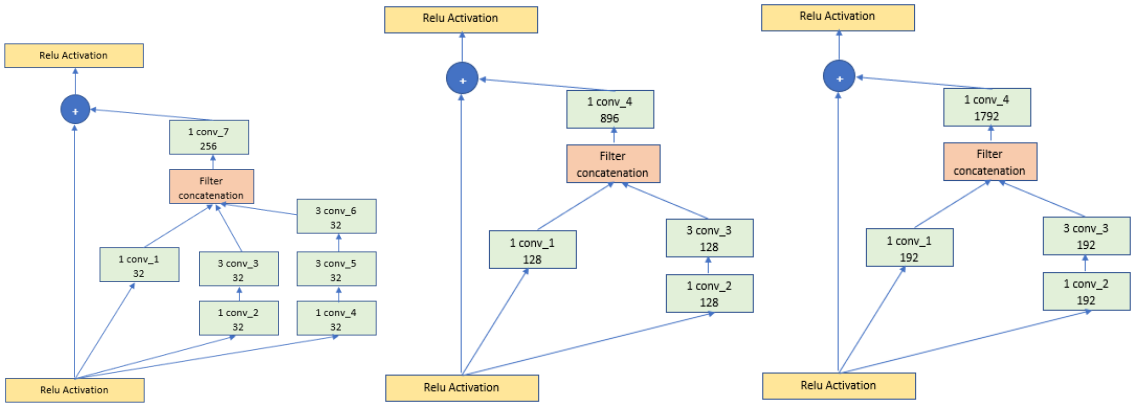
***Figura 48:*** *Módulo Inception-ResNet-A(izquierda), Inception-ResNet-B(centro) y Inception-ResNet-C (derecha).*




Fuente: (Szegedy et al., 2016)

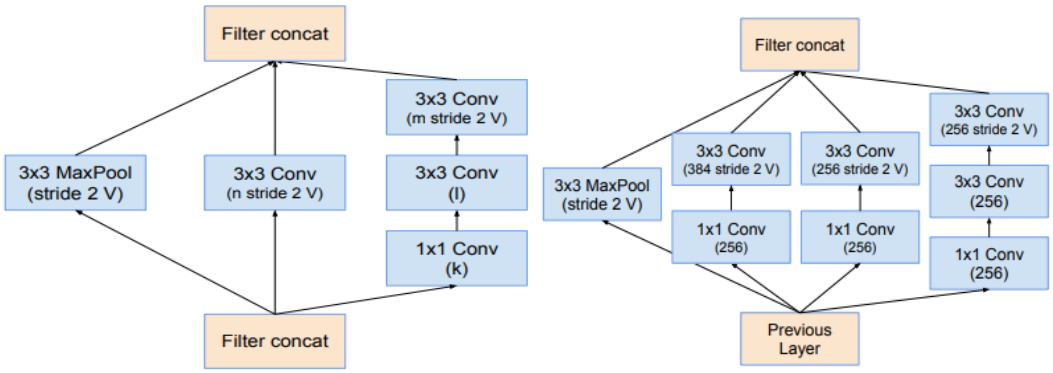
A continuación, veremos la adaptación de los módulos Inception-ResNet al procesamiento de señales unidimensionales.

***Figura 49:*** *En esta figura observamos la adaptación al procesamiento de señales unidimensionales de los módulos Inception-ResNet-A(izquierda), Inception-ResNet-B(centro) e Inception-ResNet-C (derecha).*



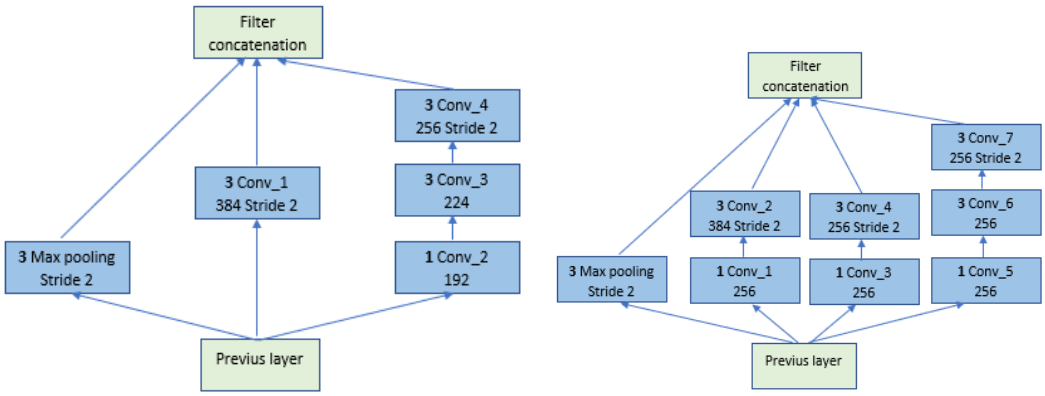

ResNet al igual que su antecesora posee 2 módulos de reducción.

***Figura 50:*** *Módulos de reducción de la arquitectura ResNetV1. Reduction_Block_A(izquierda) y Reduction_Block_A (derecha).*



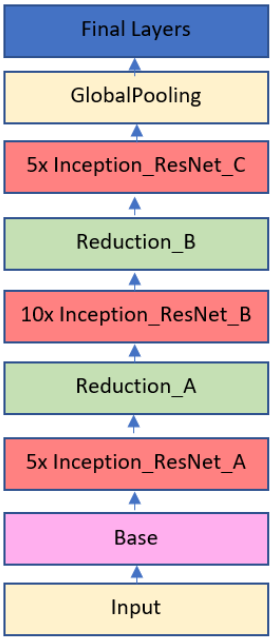

Fuente: (Szegedy et al., 2016)

***Figura 51:*** Módulos de reducción de la arquitectura ResNetV1 adaptados al procesamiento de señales unidimensionales. Reduction_Block_A(izquierda) y Reduction_Block_A (derecha).




La arquitectura ResNetv1 se resume en la Figura 52.

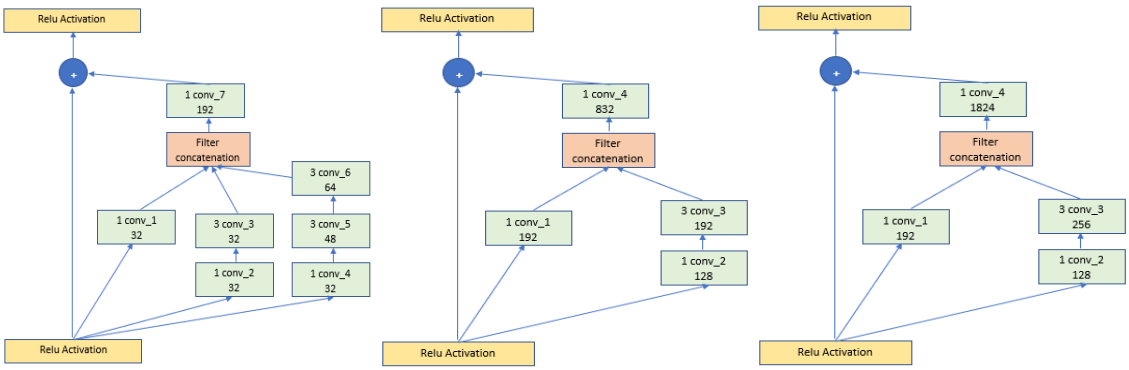
***Figura 52:*** *Arquitectura ResNetV1 1D.*




### 4.1.6. ResNetV2

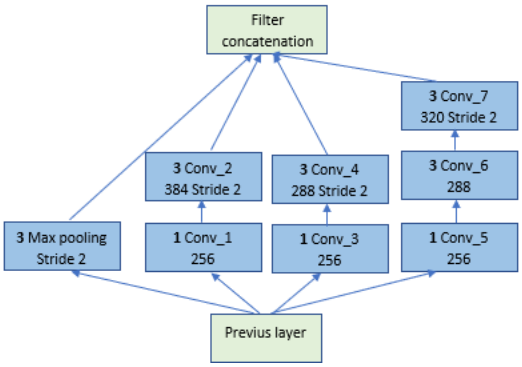
ResNetV2 posee la misma base que InceptionV4 y sus módulos inception tienen la misma estructura que ResNetV1, variando en la cantidad de kernels en sus capas convolucionales y sobre todo ResNetV2 es una red aún más profunda porque contiene 10 módulos de Inception_ResNet_A, 20 de Inception_ResNet_B y 10 de Inception_ResNet_C.

***Figura 53:*** *módulo Inception_ResNet_A(Izquierda), Inception_ResNet_B(centro) Inception_ResNet_C (derecha).*



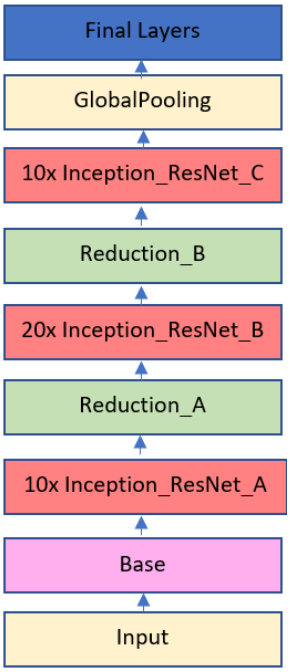
ResNetV2 también utiliza 2 bloques de reducción. El reduction_Block_A es el mismo que el de ResNetV1 y el Reduction_Block_B varía ligeramente la cantidad de kernels en algunas convoluciones.

***Figura 54:*** *Reduction_Block_B de la arquitectura ResNetV2.*




La Figura 55 resume la arquitectura ResNetV2.

***Figura 55:*** *Arquitectura ResNetV2 1D.*

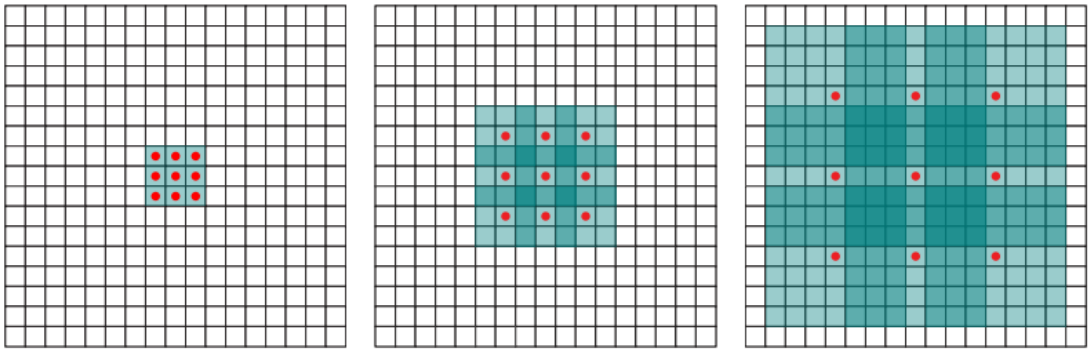


## 4.2. ARQUITECTURAS DEL EXPERIMENTO 2

### 4.2.1. Dilated convolutions

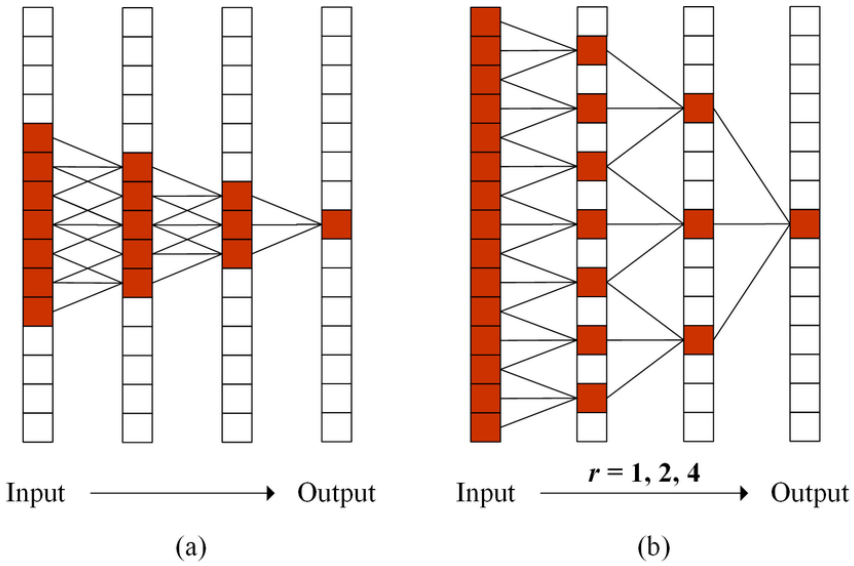
Las convoluciones dilatadas también conocidas como ‘à trous convolutions’ son un tipo de operación de convolución en la que se aplica un kernel a una imagen con un "dilatador", que espacia los elementos del kernel ampliando el campo receptivo.

***Figura 56:*** *En las imágenes podemos ver cómo funciona un kernel de tamaño 3x3 con diferentes tasas de dilatación. En la figura (a) tiene una tasa de dilatación de 1, lo que equivale a una convolución tradicional con un campo receptivo de 3x3. En la figura (b), con una tasa de dilatación de 2, el campo receptivo del kernel aumenta a 7x7 y captura más información espacial sin cambiar el tamaño del kernel. En la figura (c), con una tasa de dilatación de 4, el campo receptivo del kernel aumenta a 15x15.*




Fuente: (Yu & Koltun, 2016)

***Figura 57:*** *En la figura (a) vemos el árbol de capas de una red neuronal convolucional 1D convencional y en la figura (b) vemos el árbol de capas de una red neuronal convolucional 1D con un aumento exponencial del dilation rate en base 2.*




Fuente: (Hao et al., 2019)

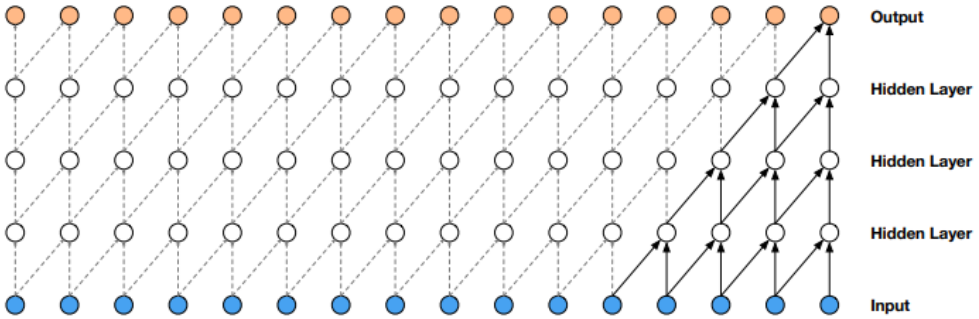
La figura muestra que las capas de convolución dilatadas permiten recopilar información espacial en la entrada sin tener que agrandar el tamaño del kernel. Esto optimiza las redes basadas en redes neuronales convolucionales, ya que podemos trabajar con kernels más pequeños.

### 4.2.2. Causal Convolutional Layers

Las Convoluciones Causales son aquellas que solo dependen del presente y del pasado, pero no del futuro, en el procesamiento de una señal temporal. Esto permite que las salidas de las operaciones sean solo dependientes de las entradas pasadas y no de futuras, lo que es útil en ciertas aplicaciones como la generación de audio o video e incluso series temporales.

Un ejemplo de aplicación es WaveNet un modelo generativo de audio. “El ingrediente principal de WaveNet son las convoluciones causales. Mediante el uso de convoluciones causales, nos aseguramos de que el modelo no pueda violar el orden en el que modelamos los datos.” (Oord et al., 2016).

***Figura 58:*** *Visualización del árbol de capas de una red con operadores de convolución causales.*

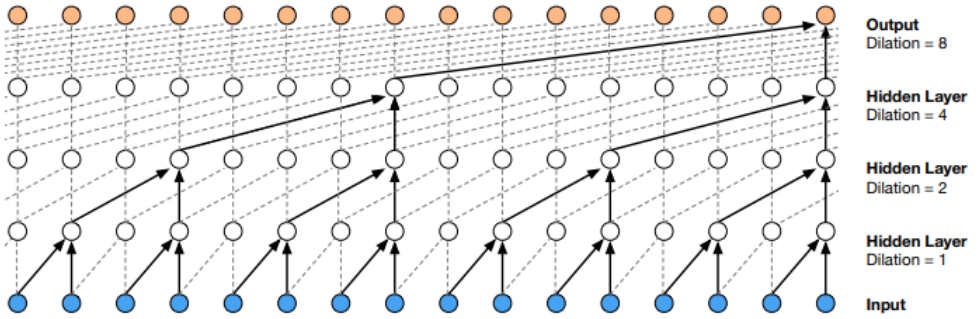



Fuente: (Oord et al., 2016)

### 4.2.3. Dilated Causal Convolutional layers

Las capas de Convolución Causal Dilatada combinan las bondades de las Convoluciones Dilatadas y las Convoluciones Causales, aumentando el campo receptivo del kernel sin incluir información futura.

***Figura 59:*** *Visualización del árbol de capas de una red con operadores de convolución causales dilatados también llamada Redes de Convoluciones Temporales (TCNs por sus siglas en inglés).*



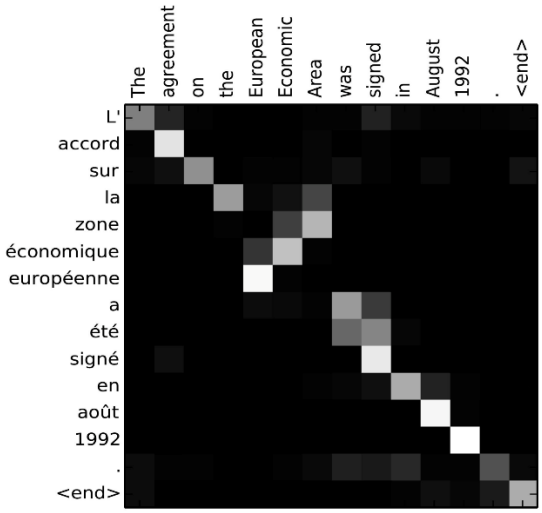

Fuente: (Oord et al., 2016)

### 4.2.4. Attention

Bahdanau, Cho y Bengio (2016) propusieron mecanismos de atención para solucionar el problema de información que enfrentaba la arquitectura de RNN codificador-decodificador inicial aplicada a traducción.

***Figura 60:*** *La figura muestra una matriz de alineación visualizada para una traducción del francés al inglés. Cada cuadrado representa la intensidad o peso de la alineación entre la*

*entrada y la salida. El color negro, con valores más bajos, se representa como 0 y el color blanco, con el valor más alto, se representa como 1.*




Fuente: (Bahdanau et al., 2016)

Si bien Bahdanau, Cho y Bengio (2016) fueron los primeros en usar mecanismos de atención, Loung et al. (2015) es el primero en proponer distintos mecanismos de atención.

***Figura 61:*** *Score function utilizados en los distintos modelos de atención: global, local-m, local-p.*

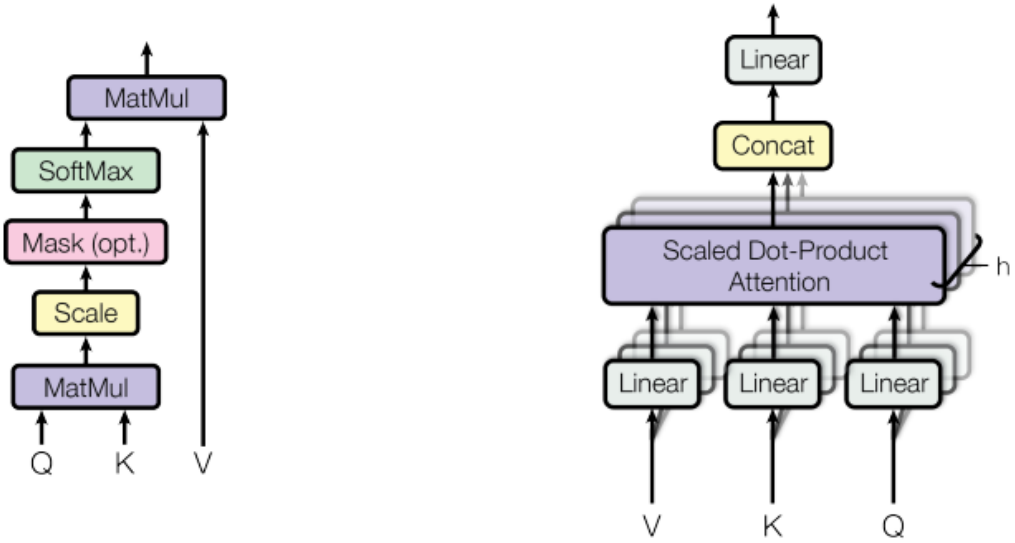
$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_a \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_a[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

$$\boldsymbol{a}_t = \text{softmax}(\boldsymbol{W}_a \boldsymbol{h}_t) \qquad \textit{location}$$

Fuente: (Luong et al., 2015)

Posteriormente, Vaswani el al. (2017) populariza los mecanismos de atención en el artículo "Attention Is All You Need" implementado en la novedosa arquitectura Transformers.

***Figura 62:*** *Scaled Dot-Product Attention (izquierda), Multi-Head Attention (derecha) utilizadas en la arquitectura Transformers.*




Fuente: (Vaswani et al., 2017)

El mecanismo de Scaled Dot-Product Attention se puede resumir en la siguiente expresión:

$$Attention(Q, K, V) = softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V$$

Donde $Q$ es el vector Query, $K$ es el vector key, $V$ el vector value y $d_k$ es la dimensión de los vectores. Scaled Dot-Product Attention hace referencia al score function "dot" de Bahdanau, Cho y Bengio (2016), donde su principal diferencia es el factor de escalado $\frac{1}{\sqrt{d_k}}$.

Multi-Head Attention utiliza Scaled Dot-Product Attention múltiples veces en paralelo, denominada heads. El mecanismo de Multi-Head Attention se resume en la siguiente expresión:

$$MultiHead(Q, K, V) = Concat(head_1, head_2, \dots, head_h)W^O$$

$$donde\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Cada $W$ es una matriz de parámetros que transforma los vectores $Q, K\ y\ V$ antes de entrar al módulo Scaled Dot-Product Attention.

$Q$ y $K$ son dos vectores diferentes, como en el caso de la traducción automática donde $Q$ es el vector de la oración en idioma 1 y $K$ el de la oración en idioma 2. Pero la atención también se puede usar en otros problemas cuando $Q$ y $K$ son el mismo vector, llamándose entonces autoatención.

### 4.2.5. Put it all together

En el segundo experimento, las capas de convolución dilatada son fundamentales para el éxito de este proyecto debido a su capacidad para aumentar el alcance receptor de la red sin aumentar su complejidad. Además, estas capas son compatibles y fácilmente integrables con las capas de convolución casual y de atención, por lo que serán incluidas en cada arquitectura.

El número de capas de convolución, cantidad de kernels y demás hiper parámetros se determinaron mediante experimentación.

***Figura 63:*** *arquitectura DilatedConv (izquierda), DilatedConv+attention (centro), DilatedConv+multihead attention (derecha).*

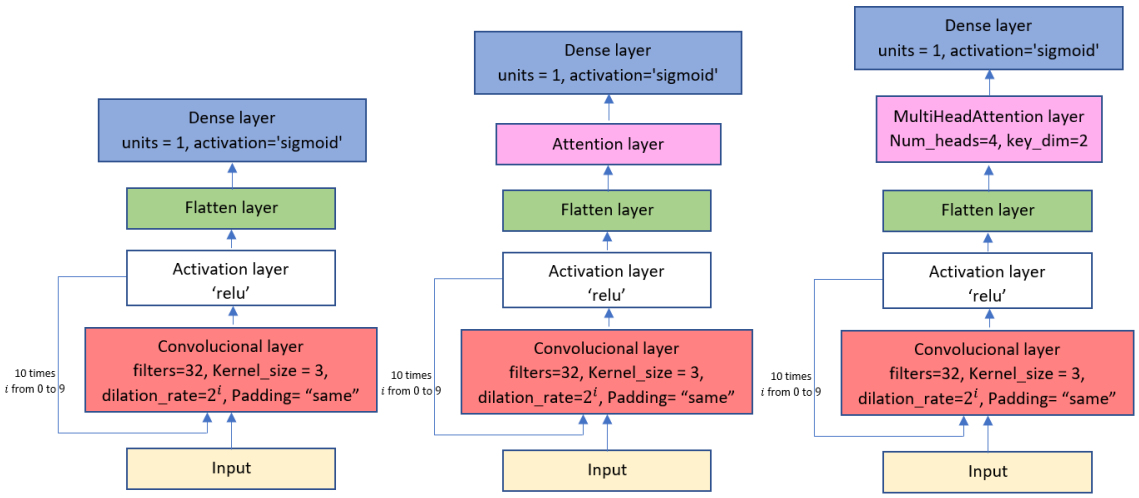
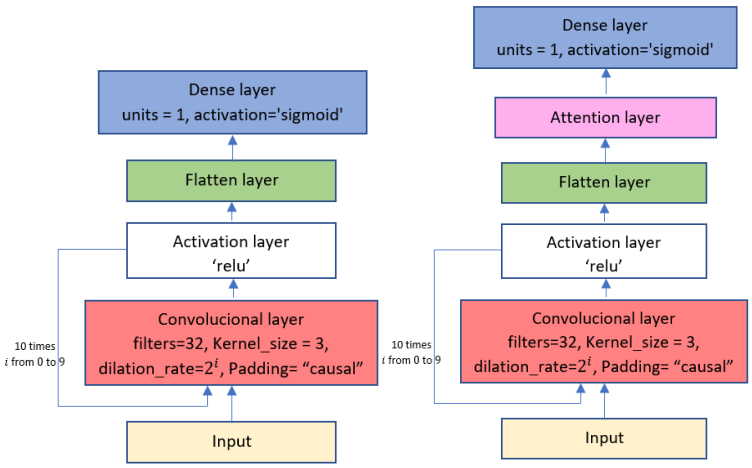


***Figura 64:*** *TCN (izquierda), TCN+attention (derecha).*



# 5. Desarrollo de la comparativa

En el desarrollo de esta comparativa utilizamos como base el algoritmo Stella en su configuración por defecto:

El algoritmo Stella implementa la división de la base de datos en tres conjuntos a través de la función "split_data": entrenamiento (80%), validación (10%) y prueba (10%). Para garantizar la aleatoriedad en la distribución de los datos, se utiliza la función "do_shuffle" con una semilla fija de 123. Además, el parámetro "frac_balance" se utiliza para equilibrar la relación entre fulguraciones (flares a partir de aquí) y no flares en los datos, con un valor predeterminado de 0.73. Finalmente, el entrenamiento de cada modelo se realiza con 200 cadencias y una semilla aleatoria de 2. Se establece "adam" como el optimizador, se utiliza la función de pérdida "binary_crossentropy" y las métricas seleccionadas son "accuracy", "precision" y "recall".

Para el desarrollo de ambos experimentos se utiliza Google Colab (*Google Collaboratory*, s. f.) para poder paralelizar los cálculos en una GPU.

## 5.1. PRIMER EXPERIMENTO

En el primer experimento comparamos la arquitectura de la red neuronal de Stella con las arquitecturas InceptionV1, InceptionV2, InceptionV3, InceptionV4, ResNetV1 y ResNetV2.

### 5.1.1. Comparativa de los modelos del experimento 1

#### 5.1.1.1. Resumen de los modelos resultantes del experimento 1.

**Tabla 9:** *Resumen métricas de los modelos del experimento 1 en el conjunto de validación.*

| | *Conjunto de validación* | | | | |
|---|---|---|---|---|---|
| Modelo | Accuracy | Precision (Flare – Non Flare) | Recall (Flare – Non Flare) | F1-Score (Flare – Non Flare) | ROC |
| Stella | 0.98 | 0.97-0.98 | 0.91-0.99 | 0.94-0.99 | 0.99 |
| Inceptionv1 | 0.98 | 0.98-0.99 | 0.94-1.00 | 0.96-0.99 | 0.99 |
| Inceptionv2 | 0.97 | 0.99-0.97 | 0.88-1.00 | 0.93-0.98 | 0.99 |
| Inceptionv3 | 0.94 | 0.97-0.93 | 0.71-0.99 | 0.82-0.96 | 0.97 |
| Inceptionv4 | 0.99 | 0.99-0.99 | 0.94-1.00 | 0.96-0.99 | 1 |
| ResNetv1 | 0.99 | 0.98-0.99 | 0.96-1.00 | 0.97-0.99 | 0.99 |
| ResNetv2 | 0.99 | 0.99-0.99 | 0.96-1.00 | 0.97-0.99 | 0.99 |

En la Tabla 9, del conjunto de validación, se puede apreciar que cuatro de los modelos evaluados en este experimento han mejorado los resultados de referencia (Stella). Uno de los problemas más graves en el desempeño de Stella era la presencia de falsos positivos.

En la Tabla 9, se puede observar que los modelos InceptionV1, InceptionV4, ResNetV1 y ResNetV2 han mejorado las métricas de precisión, sensibilidad y F1, lo cual indica que estos cuatro modelos clasifican de manera más precisa los verdaderos positivos en comparación con Stella.

**Tabla 10:** *Resumen métricas de los modelos del experimento 1 en el conjunto de test.*

| | *Conjunto de test* | | | | |
|---|---|---|---|---|---|
| Modelo | Accuracy | Precision[1] (Flare – Non Flare) | Recall[2] (Flare – Non Flare) | F1-Score[3] (Flare – Non Flare) | ROC |
| Stella | 0.98 | 0.98-0.98 | 0.92-1.00 | 0.95-0.99 | 0.99 |
| Inceptionv1 | 0.99 | 0.99-0.99 | 0.94-1.00 | 0.96-0.99 | 0.99 |
| Inceptionv2 | 0.98 | 0.99-0.97 | 0.89-1.00 | 0.94-0.99 | 0.99 |
| Inceptionv3 | 0.95 | 0.97-0.94 | 0.74-1.00 | 0.84-0.97 | 0.97 |
| Inceptionv4 | 0.99 | 0.99-0.99 | 0.95-1.00 | 0.97-0.99 | 1 |
| ResNetv1 | 0.99 | 0.99-0.99 | 0.97-1.00 | 0.98-0.99 | 0.99 |
| ResNetv2 | 0.99 | 0.99-0.99 | 0.95-1.00 | 0.97-0.99 | 0.99 |

En la Tabla 10 en el conjunto de test también observamos el mismo comportamiento de los modelos que en la Tabla 9.

**Tabla 11:** *Resumen de matriz de confusión de los modelos del experimento 1.*

| | *Conjunto de validación* | | | | *Conjunto de test* | | | |
|---|---|---|---|---|---|---|---|---|
| Modelo | TP | TN | FP | FN | TP | TN | FP | FN |
| Stella | 801 | 3550 | 28 | 80 | 780 | 3597 | 17 | 65 |
| Inceptionv1 | 828 | 3564 | 14 | 53 | 797 | 3604 | 10 | 48 |
| Inceptionv2 | 773 | 3573 | 5 | 108 | 748 | 3610 | 4 | 97 |
| Inceptionv3 | 625 | 3558 | 20 | 256 | 624 | 3598 | 16 | 221 |
| Inceptionv4 | 830 | 3566 | 12 | 51 | 803 | 3605 | 9 | 42 |
| ResNetv1 | 844 | 3562 | 16 | 37 | 816 | 3602 | 12 | 29 |
| ResNetv2 | 843 | 3567 | 11 | 38 | 805 | 3607 | 7 | 40 |

[1] la métrica precision desagregada por clases nos indica el porcentaje de aciertos en esa clase que predice nuestro modelo. Tomando en cuenta las instancias en las que se equivocó clasificándolas como esa clase.
[2] la métrica recall desagregada por clases nos indica el porcentaje de aciertos en esa clase que predice nuestro modelo. Tomando en cuenta las instancias las instancias que debió haber clasificado como esa clase.
[3] la métrica F1-Score desagregada por clases nos indica si hay o no equilibrio entre las métricas de precision y recall clasificando esa clase.

En la Tabla 11 se observa que todos los modelos propuestos disminuyen los falsos positivos, mientras que los modelos Inceptionv1, Inceptionv4, ResNetv1 y ResNetv2 también logran disminuir los falsos negativos.

5.1.1.2. Rendimiento de los modelos del experimento 1 durante el entrenamiento.

***Figura 65:*** *Comparativa del accuracy durante el entrenamiento en el conjunto de entrenamiento (izquierda), Comparativa del accuracy durante el entrenamiento en el conjunto de validación (derecha).*

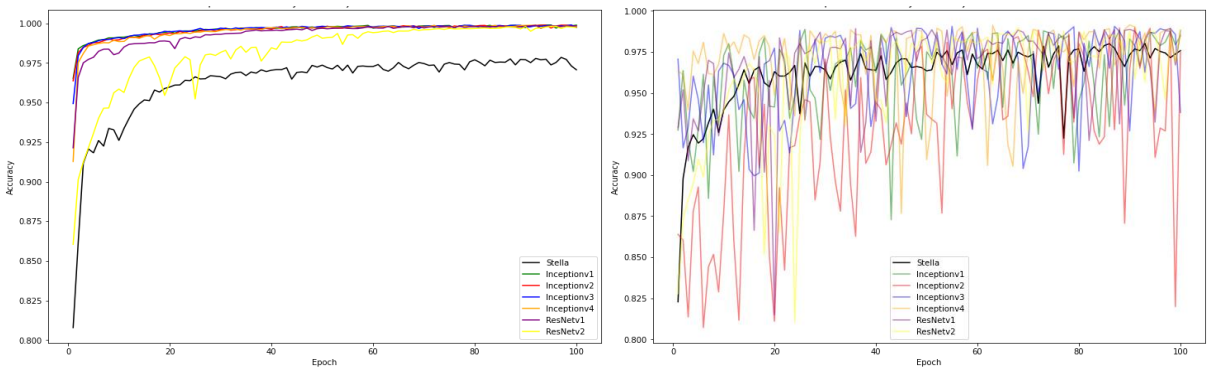



***Figura 66:*** *Comparativa de Precision durante el entrenamiento en el conjunto de entrenamiento (izquierda), Comparativa de Precision durante el entrenamiento en el conjunto de validación (derecha).*

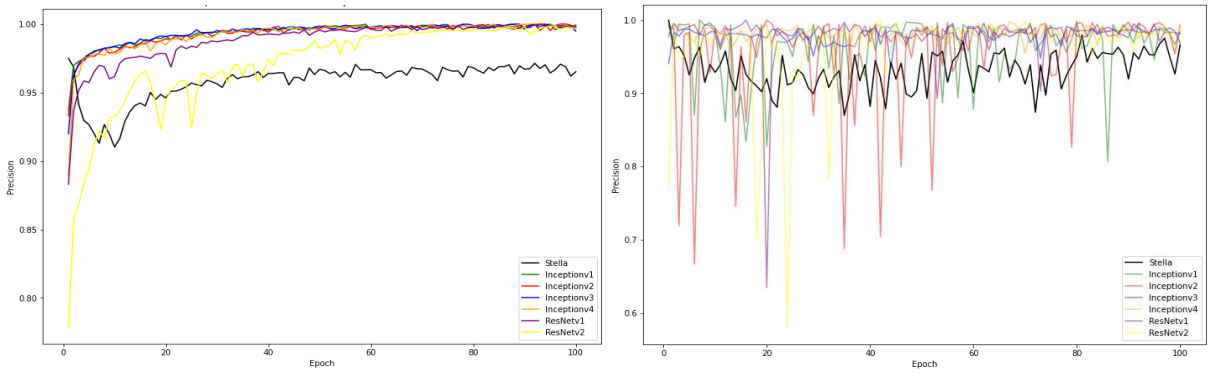
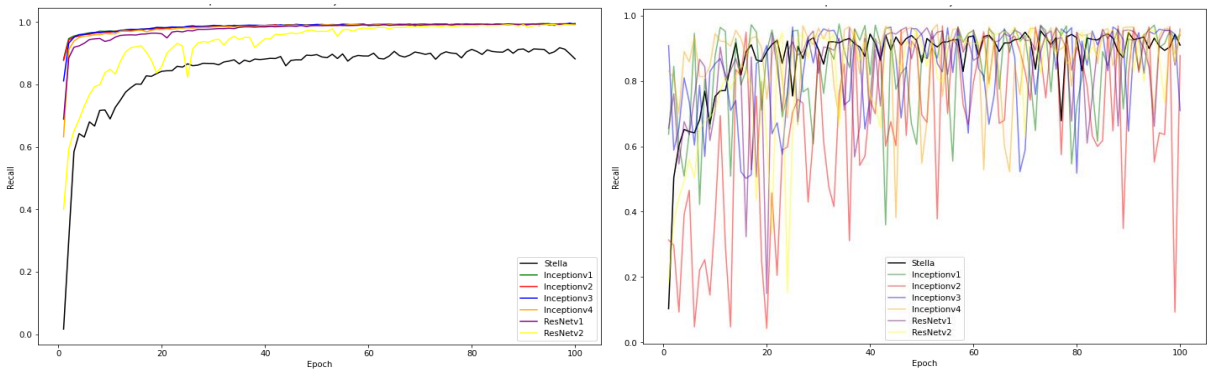


***Figura 67:*** *Comparativa de recall durante el entrenamiento en el conjunto de entrenamiento (izquierda), Comparativa del recall durante el entrenamiento en el conjunto de validación (derecha).*




En las figuras 65, 66 y 67 podemos observar que todos los modelos se están sobre ajustando al conjunto de entrenamiento, incluso algunos más que Stella.

**Tabla 12:** *Resumen del rendimiento de los modelos del experimento 1 durante el entrenamiento (media ± desviación estándar).*

| *Modelo* | *accuracy* | *val_accuracy* | *precision* | *val_precision* | *recall* | *val_recall* |
|---|---|---|---|---|---|---|
| Stella | 0.96±0.02 | 0.96±0.02 | 0.96±0.01 | 0.93±0.02 | 0.85±0.12 | 0.87±0.11 |
| Inceptionv1 | 0.996±0.004 | 0.96±0.03 | 0.99±0.01 | 0.96±0.04 | 0.99±0.01 | 0.84±0.14 |
| Inceptionv2 | 0.996±0.005 | 0.93±0.05 | 0.99±0.01 | 0.96±0.07 | 0.98±0.01 | 0.67±0.26 |
| Inceptionv3 | 0.996±0.006 | 0.96±0.03 | 0.99±0.01 | 0.98±0.01 | 0.98±0.02 | 0.83±0.13 |
| Inceptionv4 | 0.995±0.009 | 0.97±0.02 | 0.99±0.01 | 0.98±0.01 | 0.98±0.04 | 0.88±0.11 |
| ResNetv1 | 0.993±0.009 | 0.97±0.03 | 0.99±0.02 | 0.97±0.03 | 0.98±0.03 | 0.85±0.14 |
| ResNetv2 | 0.98±0.02 | 0.96±0.04 | 0.97±0.04 | 0.96±0.06 | 0.93±0.09 | 0.83±0.17 |

En la tabla 12, se puede observar que hay una mayor variabilidad en las métricas de validación que en las de entrenamiento, lo que sugiere que los modelos pueden estar sobre ajustando al conjunto de entrenamiento.

**Tabla 13:** *Coste computacional basados en la cantidad de parámetros a entrenar.*

| *Modelo* | *Total de parámetros* | *Parámetros entrenables* | *Parámetros no entrenables* |
|---|---|---|---|
| Stella | 105,729 | 105,729 | 0 |
| Inceptionv1 | 3,451,121 | 3,436,561 | 14,560 |
| Inceptionv2 | 5,760,680 | 5,740,392 | 20,288 |
| Inceptionv3 | 12,569,041 | 12,540,977 | 28,064 |
| Inceptionv4 | 28,558,465 | 28,507,457 | 51,008 |
| ResNetv1 | 16,051,537 | 15,985,137 | 66,400 |
| ResNetv2 | 35,151,841 | 35,027,041 | 124,800 |

En la tabla 13 se describe el número de parámetros entrenables y no entrenables de cada modelo.

## 5.2. SEGUNDO EXPERIMENTO

### 5.2.1. Comparativa de los modelos del experimento 2.

#### 5.2.1.1. Resumen de los modelos resultantes del experimento 2.

**Tabla 14:** *Resumen métricas de los modelos del experimento 2 en el conjunto de validación.*

| | *Conjunto de validación* | | | | |
|---|---|---|---|---|---|
| Modelo | Accuracy | Precision (Flare – Non Flare) | Recall (Flare – Non Flare) | F1-Score (Flare – Non Flare) | ROC |
| Stella | 0.98 | 0.97-0.98 | 0.91-0.99 | 0.94-0.99 | 0.99 |
| DilatedConv | 0.99 | 0.97-0.99 | 0.97-0.99 | 0.97-0.99 | 0.99 |
| DilatedConv +attention | 0.99 | 0.98-0.99 | 0.96-1.00 | 0.97-0.99 | 0.99 |
| DilatedConv +multihead attention | 0.99 | 0.98-0.99 | 0.96-0.99 | 0.97-0.99 | 1.00 |
| TCN | 0.99 | 0.97-0.99 | 0.95-0.99 | 0.96-0.99 | 0.99 |
| TCN +attention | 0.99 | 0.97-0.99 | 0.95-0.99 | 0.96-0.99 | 0.99 |

En la tabla 14 en el conjunto de validación podemos observar que los 5 modelos propuestos para este experimento mejoran las métricas que indican la clasificación de ambas clases. En particular mejoran en +1% el accuracy.

**Tabla 15:** *Resumen métricas de los modelos del experimento 2 en el conjunto de test.*

| | *Conjunto de test* | | | | |
|---|---|---|---|---|---|
| Modelo | Accuracy | Precision (Flare – Non Flare) | Recall (Flare – Non Flare) | F1-Score (Flare – Non Flare) | ROC |
| Stella | 0.98 | 0.98-0.98 | 0.92-1.00 | 0.95-0.99 | 0.99 |
| DilatedConv | 0.99 | 0.97-0.99 | 0.96-0.99 | 0.96-0.99 | 0.99 |
| DilatedConv +attention | 0.99 | 0.98-0.99 | 0.97-1.00 | 0.97-0.99 | 0.99 |
| DilatedConv +multihead attention | 0.99 | 0.98-0.99 | 0.96-1.00 | 0.97-0.99 | 0.99 |
| TCN | 0.99 | 0.97-0.99 | 0.96-0.99 | 0.97-0.99 | 0.99 |
| TCN +attention | 0.99 | 0.97-0.99 | 0.96-0.99 | 0.96-0.99 | 0.99 |

En la Tabla 15 del conjunto de pruebas, se observa un comportamiento similar al encontrado en el conjunto de validación. Los cinco modelos evaluados muestran una mejoría en la identificación de verdaderos positivos en comparación con el modelo de referencia (Stella).

**Tabla 16:** *Resumen de matriz de confusión de los modelos del experimento 2.*

| | *Conjunto de validación* | | | | *Conjunto de test* | | | |
|---|---|---|---|---|---|---|---|---|
| Modelo | TP | TN | FP | FN | TP | TN | FP | FN |
| Stella | 801 | 3550 | 28 | 80 | 780 | 3597 | 17 | 65 |
| DilatedConv | 851 | 3551 | 27 | 30 | 813 | 3587 | 27 | 32 |
| DilatedConv +attention | 845 | 3563 | 15 | 36 | 817 | 3597 | 17 | 28 |
| DilatedConv +multihead attention | 844 | 3557 | 21 | 37 | 810 | 3601 | 13 | 35 |
| TCN | 841 | 3552 | 26 | 40 | 814 | 3588 | 26 | 31 |
| TCN +attention | 840 | 3553 | 25 | 41 | 811 | 3587 | 27 | 34 |

En la tabla 16 podemos observar que todos los modelos mejoran la clasificación de las instancias en el conjunto de validación y en el conjunto de test mejora en la mayoría de clasificaciones.

5.2.1.2. Rendimiento de los modelos del experimento 2 durante el entrenamiento.

***Figura 68:*** *Comparativa del accuracy durante el entrenamiento en el conjunto de train (izquierda), Comparativa del accuracy durante el entrenamiento en el conjunto de validación (derecha).*

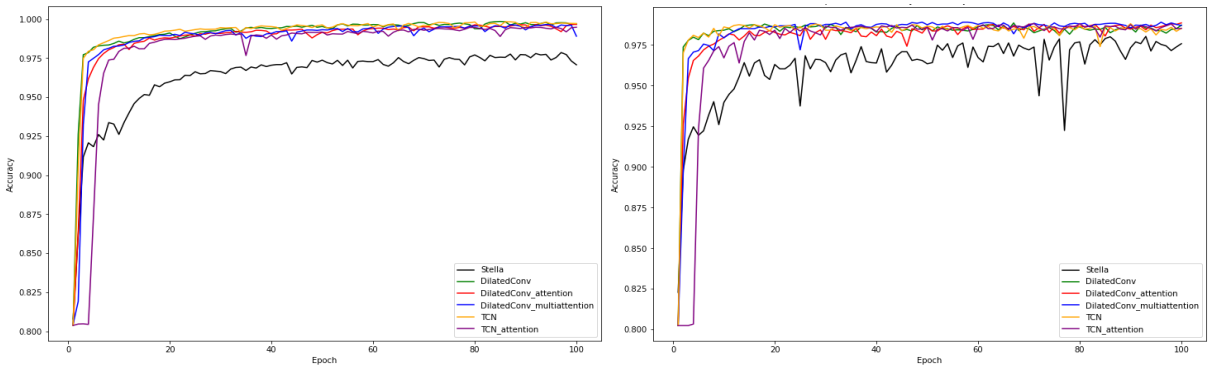



***Figura 69:*** *Comparativa de Precision durante el entrenamiento en el conjunto de train (izquierda), Comparativa de Precision durante el entrenamiento en el conjunto de validación (derecha).*

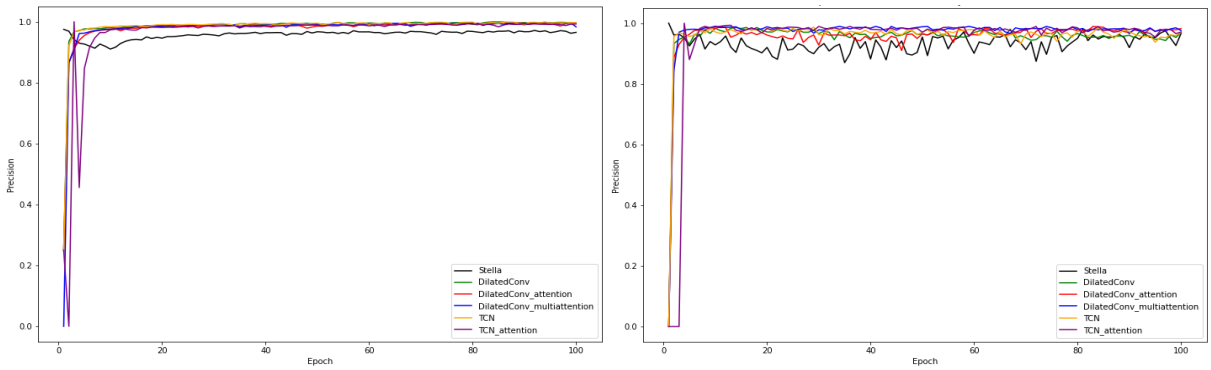
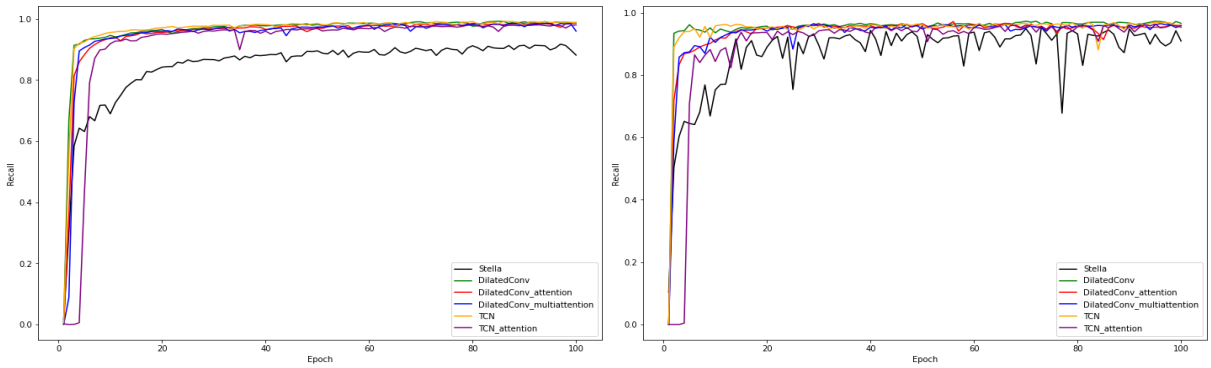



***Figura 70:*** *Comparativa de recall durante el entrenamiento en el conjunto de train (izquierda), Comparativa del recall durante el entrenamiento en el conjunto de validación (derecha).*



En las figuras 68, 69 y 70 podemos observar que los 5 modelos propuestos se ajustan muy bien a los datos, reduciendo la variabilidad de las métricas tanto al conjunto de entrenamiento como al conjunto de validación y no hay indicios de sobre ajuste. Incluso, sus gráficas son más suaves que las de Stella.

Esto lo podemos contrastar con la tabla 16 en donde en promedio mejoran todas las métricas tanto en el conjunto de entrenamiento, como en el conjunto de validación.

**Tabla 17:** *Resumen del rendimiento de los modelos del experimento 2 durante el entrenamiento (media ± desviación estándar).*

| *Modelo* | *accuracy* | *val_accuracy* | *precision* | *val_precision* | *recall* | *val_recall* |
|---|---|---|---|---|---|---|
| Stella | 0.96±0.02 | 0.96±0.02 | 0.96±0.01 | 0.93±0.02 | 0.85±0.12 | 0.87±0.11 |
| DilatedConv | 0.99±0.02 | 0.98±0.02 | 0.98±0.07 | 0.95±0.10 | 0.96±0.10 | 0.94±0.10 |
| DilatedConv +attention | 0.99±0.02 | 0.98±0.02 | 0.98±0.07 | 0.96±0.10 | 0.95±0.12 | 0.94±0.10 |
| DilatedConv +multihead attention | 0.99±0.02 | 0.98±0.02 | 0.97±0.10 | 0.97±0.10 | 0.94±0.13 | 0.93±0.10 |
| TCN | 0.99±0.02 | 0.98±0.02 | 0.98±0.07 | 0.96±0.10 | 0.96±0.11 | 0.95±0.10 |
| TCN +attention | 0.98±0.04 | 0.98±0.04 | 0.96±0.13 | 0.95±0.17 | 0.92±0.20 | 0.90±0.19 |

**Tabla 18:** *Coste computacional basados en la cantidad de parámetros a entrenar.*

| *Modelo* | *Total de parámetros* | *Parámetros entrenables* | *Parámetros no entrenables* |
|---|---|---|---|
| Stella | 105,729 | 105,729 | 0 |
| DilatedConv | 34,465 | 34,465 | 0 |
| DilatedConv +attention | 34,465 | 34,465 | 0 |
| DilatedConv +multihead attention | 35,545 | 35,545 | 0 |
| TCN | 34,465 | 34,465 | 0 |
| TCN +attention | 34,465 | 34,465 | 0 |

## 5.3. ENSAMBLE DE LOS MODELOS DEL SEGUNDO EXPERIMENTO

Para este ensamble utilizamos los modelos DilatedConv, DilatedConv+attention, DilatedConv+multihead attention, TCN, TCN+attention ya entrenados en el experimento 2.

Con el propósito de establecer la clasificación de flare o no flare, se realiza un proceso de votación suave en el cual cada modelo individual genera una distribución de probabilidad para cada clase. Posteriormente, se calcula un promedio ponderado de las distribuciones de probabilidad emitidas por cada modelo. Para determinar la predicción final, se establece un umbral de clasificación en 0.5, en donde cualquier valor por encima del umbral se clasifica como flare y cualquier valor por debajo se clasifica como no flare.

**Tabla 19:** *Métricas del ensamble de los 5 modelos.*

| *Conjunto* | *Accuracy* | *Precision (Flare-no flare)* | *Recall (Flare-no flare)* | *F1-score (Flare-no flare)* | *ROC* |
|---|---|---|---|---|---|
| Validación | 0.99 | 0.99-0.99 | 0.96-1.00 | 0.97-0.99 | 0.98 |
| Test | 0.99 | 0.99-0.99 | 0.97-1.00 | 0.98-1.00 | 0.98 |

En la tabla 19 observamos la mejoría que obtienen las métricas del ensamble con respecto a los modelos individuales del experimento 2.

**Tabla 20:** *Comparación de la matriz de confusión entre Stella y el ensamble de los 5 modelos.*

| | *Conjunto de validación* | | | | *Conjunto de test* | | | |
|---|---|---|---|---|---|---|---|---|
| Modelo | TP | TN | FP | FN | TP | TN | FP | FN |
| Stella | 801 | 3550 | 28 | 80 | 780 | 3597 | 17 | 65 |
| Ensamble | 847 | 3567 | 11 | 34 | 816 | 3608 | 6 | 29 |

En la tabla 20 observamos una mejora considerable en la clasificación de nuestro ensamble con respecto a Stella.

# 6. Discusión y análisis de resultados

En este capítulo discutiremos el significado de los resultados presentados en el Capítulo 5. Para ello lo dividiremos en 3 secciones:

- Experimento 1
- Experimento 2
- Ensamble de los 5 modelos propuestos en el experimento 2

## 6.1. EXPERIMENTO 1

En esta sección, se llevará a cabo un análisis de los modelos Inceptionv1, Inceptionv4, ResNetv1 y ResNetv4 del experimento 1, ya que estos modelos presentan una mejora significativa en la clasificación de los verdaderos positivos en comparación con el modelo Stella.

**Tabla 21:** *Modelos que clasifican mejor los verdaderos positivos que Stella en el conjunto de validación.*

| *Modelo* | *Precision* | *%* $\frac{modelo\ -Stella}{Stella}\cdot 100$ | *Recall* | *%* $\frac{modelo\ -Stella}{Stella}\cdot 100$ | *F1-Score* | *%* $\frac{modelo\ -Stella}{Stella}\cdot 100$ |
|---|---|---|---|---|---|---|
| Stella | 97 | | 91 | | 94 | |
| InceptionV1 | 98 | +1 | 94 | +3.2 | 96 | +2.1 |
| InceptionV4 | 99 | +2 | 94 | +3.2 | 96 | +2.1 |
| ResNetV1 | 98 | +1 | 96 | +5.5 | 97 | +3.2 |
| ResNetV4 | 99 | +2 | 96 | +5.5 | 97 | +3.2 |
| Promedio | | +1.5 | | +4.35 | | +2.65 |

En la Tabla 21 se muestra el porcentaje de mejora en el conjunto de validación de cada modelo en las métricas de precisión, Recall y puntuación F1. Es importante destacar que se ha observado una mejora significativa en el Recall, lo que indica que estos modelos han mejorado su capacidad para identificar verdaderos positivos, teniendo en cuenta la cantidad de falsos negativos, es decir, los eventos de flares que inicialmente habían sido clasificados incorrectamente como no flares.

**Tabla 22:** *Modelos que clasifican mejor los verdaderos positivos que Stella en el conjunto de test.*

| *Modelo* | *Precision* | *%* $\frac{modelo\ -Stella}{Stella}\cdot 100$ | *Recall* | *%* $\frac{modelo\ -Stella}{Stella}\cdot 100$ | *F1-Score* | *%* $\frac{modelo\ -Stella}{Stella}\cdot 100$ |
|---|---|---|---|---|---|---|
| Stella | 98 | | 92 | | 95 | |
| InceptionV1 | 99 | +1 | 94 | +2.2 | 96 | +1.1 |
| InceptionV4 | 99 | +1 | 95 | +3.3 | 97 | +2.1 |
| ResNetV1 | 99 | +1 | 97 | +5.4 | 98 | +3.2 |
| ResNetV4 | 99 | +1 | 95 | +3.3 | 97 | +2.1 |
| Promedio | | +1 | | +3.55 | | +2.13 |

En la Tabla 22, se puede observar un patrón similar en el desempeño de los modelos en el conjunto de test en comparación con el conjunto de validación de la Tabla 20.

**Tabla 23:** *Comparación de parámetros entrenables.*

| *Modelo* | *Parámetros entrenables* | *Incremento porcentual de parámetros entrenables* |
|---|---|---|
| Stella | 105,729 | |
| InceptionV1 | 3,436,561 | +3,150.3 |
| InceptionV4 | 28,507,457 | +27,862.8 |
| ResNetV1 | 15,985,137 | +15,019 |
| ResNetV4 | 35,027,041 | +33,029.1 |
| Promedio | | +16385.8 |

Debido a la naturaleza compleja de estos modelos, existe un alto riesgo de sufrir sobre ajuste durante el proceso de entrenamiento, lo cual se evidencia en las Figuras 65, 66 y 67, así como en el resumen presentado en la Tabla 12. Los resultados obtenidos por estos modelos no son lo suficientemente significativos como para cumplir con los objetivos establecidos en el presente trabajo. Es decir, los modelos tienden a sobre ajustarse al conjunto de entrenamiento y aumentan drásticamente la cantidad de parámetros.

## 6.2. EXPERIMENTO 2

En esta sección se analizará detalladamente el rendimiento de los modelos DilatedConv, DilatedConv+attention, DilatedConv+multihead attention, TCN, TCN+attention.

**Tabla 24:** *Modelos que clasifican mejor los verdaderos positivos que Stella en el conjunto de validación.*

| *Modelo* | *Precision (flare-no flare)* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *Recall (flare-no flare)* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *F1-Score (flare-no flare)* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ |
|---|---|---|---|---|---|---|
| Stella | 97-98 | | 91-99 | | 94-99 | |
| DilatedConv | 97-99 | 0, +1 | 97-99 | +6.6, 0 | 97-99 | +3.2, 0 |
| DilatedConv | 98-99 | +1, +1 | 96-100 | +5.5, +1 | 97-99 | +3.2, 0 |
| +attention | 98-99 | +1, +1 | 96-99 | +5.5, 0 | 97-99 | +3.2, 0 |
| DilatedConv | 97-99 | 0, +1 | 95-99 | +4.4, 0 | 96-99 | +2.1, 0 |
| +multihead | 97-99 | 0, +1 | 95-99 | +4.4, 0 | 96-99 | +2.1, 0 |
| attention | | | | | | |
| Promedio | | +0.4, +1 | | +5.28, +0.2 | | +2.76, 0 |

En la Tabla 24 se puede observar que los cinco modelos evaluados en el Experimento 2 en el conjunto de entrenamiento mejoraron tanto la tasa de verdaderos positivos como la tasa de verdaderos negativos, lo cual se refleja en las métricas Recall y F1-Score. En promedio, estos modelos lograron una mejora del 5.28% en la tasa de verdaderos positivos, considerando los eventos de flares que inicialmente habían sido clasificados incorrectamente como no flares, y una mejora del 2.76% en el F1-Score.

**Tabla 25:** *Modelos que clasifican mejor los verdaderos positivos que Stella en el conjunto de test.*

| *Modelo* | *Precision* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *Recall* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *F1-Score* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ |
|---|---|---|---|---|---|---|
| Stella | 98-98 | | 92-100 | | 95-99 | |
| DilatedConv | 97-99 | -1, +1 | 96-99 | +4.3, -1 | 96-99 | +1.1, 0 |
| DilatedConv | 98-99 | 0, +1 | 97-100 | +5.4, 0 | 97-99 | +2.1, 0 |
| +attention | 98-99 | 0, +1 | 96-100 | +4.3, 0 | 97-99 | +2.1, 0 |
| DilatedConv | 97-99 | -1, +1 | 96-99 | +4.3, -1 | 97-99 | +2.1, 0 |
| +multihead | 97-99 | -1, +1 | 96-99 | +4.3, -1 | 96-99 | +1.1, 0 |
| attention | | | | | | |
| Promedio | | -0.6, +1 | | +4.52, -0.6 | | +1.7,0 |

En la Tabla 25 se puede apreciar un equilibrio más favorable entre las métricas de precisión y recall. Esto se refleja en un aumento del puntaje F1, lo cual indica una mejora en la capacidad de los modelos para identificar tanto verdaderos positivos como verdaderos negativos.

**Tabla 26:** *Comparación de matrices de confusión en el conjunto de validación.*

| *Modelo* | *TP* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *TN* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *FP* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *FN* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ |
|---|---|---|---|---|---|---|---|---|
| Stella | 801 | | 3550 | | 28 | | 80 | |
| DilatedConv | 851 | +6.2 | 3551 | +0.02 | 27 | -3.6 | 30 | -62.5 |
| DilatedConv + attention | 845 | +5.5 | 3563 | +0.37 | 15 | -46.4 | 36 | -55 |
| DilatedConv + multihead attention | 844 | +5.3 | 3557 | +0.2 | 21 | -25 | 37 | -53.8 |
| TCN | 841 | +5 | 3552 | +0.06 | 26 | -7.1 | 40 | -50 |
| TCN + attention | 840 | +5 | 3553 | +0.08 | 25 | -10.7 | 41 | -49 |
| Promedio | | +5.4 | | +0.15 | | -18.6 | | -54.1 |

En la Tabla 26 se puede observar que los cinco modelos evaluados en el conjunto de validación lograron mejorar la clasificación de eventos, aumentando la cantidad de eventos que son

clasificados correctamente como verdaderos positivos y verdaderos negativos, mientras que disminuyen los eventos clasificados incorrectamente como falsos positivos y falsos negativos.

**Tabla 27:** *Comparación de matrices de confusión en el conjunto de test.*

| *Modelo* | *TP* | *%* $\frac{modelo\ -Stella}{Stella}\cdot 100$ | *TN* | *%* $\frac{modelo\ -Stella}{Stella}\cdot 100$ | *FP* | *%* $\frac{modelo\ -Stella}{Stella}\cdot 100$ | *FN* | *%* $\frac{modelo\ -Stella}{Stella}\cdot 100$ |
|---|---|---|---|---|---|---|---|---|
| Stella | 780 | | 3597 | | 17 | | 65 | |
| DilatedConv | 813 | +4.2 | 3587 | -0.28 | 27 | +58.8 | 32 | -50.8 |
| DilatedConv + attention | 817 | +4.7 | 3597 | 0 | 17 | 0 | 28 | -56.9 |
| DilatedConv + multihead attention | 810 | +4.4 | 3601 | +0.11 | 13 | | 35 | -46.2 |
| TCN | 814 | +4.4 | 3588 | -0.25 | 26 | +52.9 | 31 | -52.3 |
| TCN + attention | 811 | +4.4 | 3587 | -0.28 | 27 | +58.8 | 34 | -47.7 |
| Promedio | | +4.42 | | -0.14 | | +34.1 | | -50.8 |

En la Tabla 27 se puede observar que, en el conjunto de prueba, los modelos DilatedConv, TCN y TCN+Attention lograron mejorar la clasificación de instancias como verdaderos positivos y falsos negativos, lo cual significa que fueron capaces de identificar más eventos de flares correctamente. Sin embargo, estos modelos también sacrificaron la clasificación de verdaderos negativos y falsos positivos, lo que significa que se clasificaron erróneamente algunos eventos que no eran flares como flares.

**Tabla 28:** *Comparativa de métricas en conjunto de train durante el entrenamiento.*

| *Modelo* | *accuracy* | *%* $\frac{modelo\ -Stella}{Stella}\cdot 100$ | *precision* | *%* $\frac{modelo\ -Stella}{Stella}\cdot 100$ | *recall* | *%* $\frac{modelo\ -Stella}{Stella}\cdot 100$ |
|---|---|---|---|---|---|---|
| Stella | 96 | | 96 | | 85 | |
| DilatedConv | 99 | +3.1 | 98 | +2.1 | 96 | +12.9 |
| DilatedConv +attention | 99 | +3.1 | 98 | +2.1 | 95 | +11.8 |
| DilatedConv +multihead attention | 99 | +3.1 | 97 | +1 | 94 | +10.6 |
| TCN | 99 | +3.1 | 98 | +2.1 | 96 | +11.8 |
| TCN +attention | 98 | +2.1 | 96 | 0 | 92 | +8.2 |
| Promedio | | +2.9 | | +1.46 | | +11.06 |

**Tabla 29:** *Comparativa de métricas en conjunto de validación durante el entrenamiento.*

| *Modelo* | *accuracy* | *%* $\frac{modelo\ -Stella}{Stella}\cdot 100$ | *precision* | *%* $\frac{modelo\ -Stella}{Stella}\cdot 100$ | *recall* | *%* $\frac{modelo\ -Stella}{Stella}\cdot 100$ |
|---|---|---|---|---|---|---|
| Stella | 96 | | 93 | | 87 | |
| DilatedConv | 98 | +2.1 | 95 | +2.2 | 94 | +8 |
| DilatedConv +attention | 98 | +2.1 | 96 | +3.2 | 94 | +8 |
| DilatedConv +multihead attention | 98 | +2.1 | 97 | +4.3 | 93 | +6.9 |
| TCN | 98 | +2.1 | 96 | +3.2 | 95 | +9.2 |
| TCN +attention | 98 | +2.1 | 95 | +2.2 | 90 | +3.4 |
| Promedio | | +2.1 | | +3.02 | | +7.1 |

Se puede observar en la Tabla 28 y 29 que a medida que avanza el entrenamiento, los 5 modelos del experimento 2 presentan mejoras en el accuracy, Precision y Recall. Esto indica que los modelos están aprendiendo y ajustándose cada vez mejor a los datos de entrenamiento, mejorando su capacidad para clasificar correctamente las instancias en verdaderos positivos y verdaderos negativos, y disminuyendo la cantidad de falsos positivos y falsos negativos.

Estas métricas nos indica 2 cosas:

- Un ensamble del mismo modelo entrenado con diferentes semillas aleatorias podría mejorar los resultados de las tablas 24, 25, 26 y 27.
- Un ensamble de los 5 modelos puede mejorar el rendimiento individual, porque reduciría la variabilidad presente en un solo modelo.

Como último punto en este apartado analizamos la Tabla 18. En la Tabla 18 observamos que las 5 arquitecturas en este experimento se reducen en un gran porcentaje.

**Tabla 30:** *Comparación de parámetros entrenables.*

| *Modelo* | *Parámetros entrenables* | *Incremento porcentual de parámetros entrenables* |
|---|---|---|
| Stella | 105,729 | |
| DilatedConv | 34,465 | -67.4 |
| DilatedConv +attention | 34,465 | -67.4 |
| DilatedConv +multihead attention | 35,545 | -66.4 |
| TCN | 34,465 | -67.4 |
| TCN +attention | 34,465 | -67.4 |
| Promedio | | -67.2 |

Debido a los resultados expuestos en esta sección, se puede concluir que cada uno de los modelos propuestos es una buena alternativa con respecto a Stella, ya que todos muestran una mejora en el accuracy, precision y recall. Aumentando la tasa de clasificación de verdaderos positivos y sobre todo disminuyendo la cantidad de parámetros entrenables.

## 6.3. ENSAMBLE DE LOS 5 MODELOS DEL EXPERIMENTO 2

**Tabla 31:** *Comparativa de métricas en el conjunto de validación del ensamble de los 5 modelos del experimento 2 con respecto a Stella.*

| *Modelo* | *Accuracy* | *%* | *Precision (Flare-no flare)* | *%* | *Recall (Flare-no flare)* | *%* | *F1-score (Flare-no flare)* | *%* |
|---|---|---|---|---|---|---|---|---|
| Stella | 98 | | 97-98 | | 91-99 | | 94-99 | |
| Ensamble | 99 | +1 | 99-99 | +2.1, +1 | 96-100 | +5.5, +1 | 97-99 | +3.2,0 |

**Tabla 32:** *Comparativa de métricas en el conjunto de test del ensamble de los 5 modelos del experimento 2 con respecto a Stella.*

| *Modelo* | *Accuracy* | *%* | *Precision (Flare-no flare)* | *%* | *Recall (Flare-no flare)* | *%* | *F1-score (Flare-no flare)* | *%* |
|---|---|---|---|---|---|---|---|---|
| Stella | 98 | | 98-98 | | 92-100 | | 95-99 | |
| Ensamble | 99 | +1 | 99-99 | +1, +1 | 97-100 | +5.4, 0 | 98-100 | +3.2, +1 |

También podemos observar mejoras notorias en la matriz de confusión del ensamble con respecto a los modelos individuales

**Tabla 33:** *Comparación matriz de confusión en el conjunto de validación*.

| *Modelo* | *TP* | *%* | *TN* | *%* | *FP* | *%* | *FN* | *%* |
|---|---|---|---|---|---|---|---|---|
| Stella | 801 | | 3550 | | 28 | | 80 | |
| Ensamble | 847 | +5.7 | 3567 | +0.5 | 11 | -60.7 | 34 | -57.5 |

**Tabla 34:** *Comparación matriz de confusión en el conjunto de test.*

| *Modelo* | *TP* | *%* | *TN* | *%* | *FP* | *%* | *FN* | *%* |
|---|---|---|---|---|---|---|---|---|
| Stella | 780 | | 3597 | | 17 | | 65 | |
| Ensamble | 816 | +4.6 | 3608 | +0.3 | 6 | -64.7 | 29 | -55.4 |

Se puede apreciar en la Tabla 33 y 34 que la utilización de un ensamble de los cinco modelos del experimento 2 resulta en una disminución significativa de la tasa de error de clasificación, así como en un aumento de la tasa de clasificación correcta de verdaderos positivos y verdaderos negativos en comparación con el modelo Stella.

Lo destacable de este ensamble es que, como se puede apreciar en las gráficas suaves presentadas en las figuras 68, 69 y 70 para el conjunto de entrenamiento y validación durante el proceso de entrenamiento, es posible obtener resultados similares con una menor cantidad de épocas de entrenamiento.

# 7. Conclusiones y trabajo futuro

## 7.1. Conclusiones

A continuación, se reúnen las conclusiones alcanzadas en el presente trabajo.

En el Capítulo 3 establecimos los objetivos de este trabajo y sus criterios de éxito. El primero consistía en identificar y explorar las principales herramientas utilizadas para clasificar fulguraciones estelares en curvas de luz y el tipo de técnicas que emplean.

Identificamos dos tipos de técnicas principales:

- Basadas en técnicas en transformaciones para detectar valores atípicos
- Basadas en técnicas de Inteligencia artificial

Particularmente centramos nuestro punto de atención en las herramientas basadas en técnicas de inteligencia artificial y nos permitió examinar la estructura de:

- FLATW'RM que se basa en el algoritmo RANSAC para la detección de valores atípicos.
- FLATW'RM2 que se basa en redes neuronales recurrentes de tipo LSTM.
- Stella que se basa en redes neuronales convolucionales.

El segundo objetivo consistía en analizar y profundizar en el entendimiento del algoritmo Stella. Explorando todos sus módulos y funcionalidades y por supuesto experimentar en su módulo de Neural Network que nos permitió entrenar nuevos modelos con diferentes arquitecturas de redes neuronales. Este trabajo se basa en los resultados obtenidos en este módulo.

Como tercer objetivo nos planteamos modificar el algoritmo Stella con otras arquitecturas de redes neuronales. Para cumplir con este objetivo, entrenamos 6 modelos distintos de arquitecturas conocidas adaptadas al procesamiento de señales unidimensionales. enumerados a continuación:

- GoogLeNet o InceptionV1
- InceptionV2

- InceptionV3
- InceptionV4
- ResNetV1
- ResNetV2

Y además propusimos 5 nuevas arquitecturas que llamamos:

- DilatedConv
- DilatedConv + attention
- DilatedConv + multihead attention
- TCN
- TCN + attention

En donde exploramos una diversidad de mecanismos que nos pudiese ayudar a abordar este problema.

Nuestro último objetivo fue evaluar estos algoritmos con diversas métricas como el accuracy, precision, recall, F1-score, ROC-AUC y por supuesto su coste computacional basándonos en la cantidad de parámetros. También utilizamos como métrica el análisis visual y las métricas del entrenamiento.

Con el análisis visual de las métricas del entrenamiento, descubrimos que los modelos entrenados con las arquitecturas Inception y ResNet estaban sobre ajustados. Esto puede deberse a distintos factores: complejidad de la arquitectura, no había suficientes datos de entrenamiento, desbalance entre las clases, ajuste de hiper parámetros, etc. Particularmente no exploramos la búsqueda de hiper parámetros adecuados e intentar reducir el sobreajuste porque cada entrenamiento tomaba mucho tiempo y no contábamos con el equipo adecuado para agilizar el proceso. Por lo que nos era inviable seguir con esa ruta.

Por lo que exploramos nuevas vías, tomando como objetivo reducir la complejidad de la arquitectura. Mediante la investigación sistemática que nos permitiera resolver el problema de clasificación de fulguraciones estelares en curvas de luz encontramos 3 técnicas que nos podían ayudar.

- Capas de convoluciones dilatadas

- Capas de convoluciones causales
- Capas de atención

Esta investigación nos permitió encontrar casos de éxito como por ejemplo WaveNet, que combina las capas convoluciones dilatadas con las capas convolucionales causales en capas convolucionales causales dilatadas y por supuesto, los transformers con sus mecanismos de atención.

Las 5 arquitecturas propuestas en el experimento 2 que combinan estas tres técnicas mejoraron los resultados de Stella que fue nuestro punto de referencia. Mediante las capas convolucionales nos ayudaba a ampliar nuestro campo receptivo sin añadir complejidad al modelo, las capas convolucionales causales nos ayudaron a aprovechar el componente temporal presente en las curvas de luz y las capas de atención nos ayudaron a que el modelo aprendiera qué característica nos ayudaba a clasificar mejor.

Precisamente estas 5 arquitecturas nos ayudaron a conseguir cumplir con el objetivo general de este trabajo de investigación "Optimizar el algoritmo Stella para clasificar fulguraciones estelares en curvas de luz proporcionadas por la misión TESS utilizando nuevas arquitecturas de redes neuronales obteniendo una mejora en la clasificación de verdaderos positivos con un menor coste computacional"

Estas son las métricas que soportan esta afirmación:

- Los modelos finales en el conjunto de validación: precision: +0.4% en promedio, recall: +5.28 en promedio, f1-score: +2.76 en promedio **(Tabla 24)**
- Los modelos finales en el conjunto de test: precision: -0.6% en promedio, recall: +4.52% en promedio, f1-score: +1.7% en promedio **(Tabla 25)**
- Métricas del conjunto de train durante el entrenamiento: accuracy: +2.9% en promedio, precision: +1.46% en promedio, recall: +11.06% en promedio **(Tabla 28)**
- Métricas del conjunto de validación durante el entrenamiento: accuracy: +2.1% en promedio, precision: +3.02% en promedio, recall: +7.1% en promedio **(Tabla 29)**
- En promedio logramos un 5.4% de mejora en la clasificación de verdaderos positivos según la matriz de confusión en el conjunto de validación **(Tabla 26)** y 4.42% en el conjunto de test **(Tabla 27)** con los modelos obtenidos.

- En promedio una reducción de parámetros de 67.2% **(Tabla 30)**

Por último, el ensamble de estos 5 modelos del segundo experimento obtuvo mejores resultados que todos los modelos individuales (incluido Stella), reduciendo en un 64.7% la clasificación de los falsos positivos y un 55.4 de reducción en la clasificación de los falsos negativos.

## 7.2. Líneas de trabajo futuro

La clasificación de fulguraciones estelares es de gran relevancia en el campo de astrofísica, por lo que este trabajo puede servir de base a varias líneas de trabajo futuras.

- Análisis detallado de las instancias clasificadas incorrectamente por estos modelos: Nos permitirá identificar en los casos particulares en donde nuestros modelos fallan. De esa forma poder tomar acción en la búsqueda e implementación de métodos que eviten que nuestros modelos tengan esos fallos.
- Utilizar nuestros modelos para clasificar fulguraciones estelares en curvas de luz de los demás sectores que proporciona la misión TESS. Aplicar filtros para disminuir falsos positivos de la forma en que Feinstein et al. (2022) plantea en su artículo.
- Utilizar las arquitecturas propuestas en el experimento 2 para la clasificar fulguraciones observadas en otras longitudes de onda (UV) y otras cadencias temporales.

Las arquitecturas CNN son bastante flexibles, por lo que puede utilizarse para la clasificación de señales unidimensionales en otros ámbitos. Por ejemplo:

- Electrocardiogramas y encefalogramas en medicina
- Señales bursátiles en economía.
- Clasificación de sonidos.
- Etc.

# Referencias bibliográficas

Administración Nacional de Aeronáutica y el Espacio [NASA]. (2021). *What is an Exoplanet?* Exoplanet Exploration: Planets Beyond our Solar System. https://exoplanets.nasa.gov/what-is-an-exoplanet/overview

Amazon Web Services [AWS]. (s. f.-a). *Model Fit: Underfitting vs. Overfitting—Amazon Machine Learning*. Recuperado 29 de enero de 2023, de https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

Amazon Web Services [AWS]. (s. f.-b). *¿Qué es el sobreajuste? - Sobreajuste - AWS*. Amazon Web Services, Inc. Recuperado 29 de enero de 2023, de https://aws.amazon.com/es/what-is/overfitting/

Baglin, A., Auvergne, M., Barge, P., Deleuil, M., Catala, C., Michel, E., Weiss, W., & COROT Team. (2006). *Scientific Objectives for a Minisat: CoRoT*. *1306*, 33.

Bahdanau, D., Cho, K., & Bengio, Y. (2016). *Neural Machine Translation by Jointly Learning to Align and Translate* (arXiv:1409.0473). https://doi.org/10.48550/arXiv.1409.0473

Borucki, W. J., Koch, D. G., Lissauer, J. J., Basri, G. B., Caldwell, J. F., Cochran, W. D., Dunham, E. W., Geary, J. C., Latham, D. W., Gilliland, R. L., Caldwell, D. A., Jenkins, J. M., & Kondo, Y. (2003). *The Kepler mission: A wide-field-of-view photometer designed to determine the frequency of Earth-size planets around solar-like stars*. *4854*, 129-140. https://doi.org/10.1117/12.460266

*COROT*. (s. f.). Recuperado 6 de octubre de 2022, de https://www.esa.int/Science_Exploration/Space_Science/COROT_overview

Davenport, J. R. A. (2016). The Kepler Catalog of Stellar Flares. *The Astrophysical Journal*, *829*(1), 23. https://doi.org/10.3847/0004-637X/829/1/23

Davenport, J. R. A., Hawley, S. L., Hebb, L., Wisniewski, J. P., Kowalski, A. F., Johnson, E. C., Malatesta, M., Peraza, J., Keil, M., Silverberg, S. M., Jansen, T. C., Scheffler, M. S., Berdis, J. R., Larsen, D. M., & Hilton, E. J. (2014). Kepler Flares II: The Temporal Morphology of White-Light Flares on GJ 1243. *The Astrophysical Journal*, *797*(2), 122. https://doi.org/10.1088/0004-637X/797/2/122

Dharmaraj. (2021, septiembre 21). Zero-Padding in Convolutional Neural Networks. *Medium*. https://medium.com/@draj0718/zero-padding-in-convolutional-neural-networks-bf1410438e99

Feinstein, A. D., Montet, B. T., & Ansdell, M. (2020). \texttt{stella}: Convolutional Neural Networks for Flare Identification in \textit{TESS}. *Journal of Open Source Software*, *5*(52), 2347. https://doi.org/10.21105/joss.02347

Feinstein, A. D., Montet, B. T., Ansdell, M., Nord, B., Bean, J. L., Günther, M. N., Gully-Santiago, M. A., & Schlieder, J. E. (2020). Flare Statistics for Young Stars from a Convolutional Neural Network Analysis of TESS Data. *The Astronomical Journal*, *160*(5), 219. https://doi.org/10.3847/1538-3881/abac0a

Feinstein, A. D., Seligman, D. Z., Günther, M. N., & Adams, F. C. (2022). Testing Self-organized Criticality across the Main Sequence Using Stellar Flares from TESS. *The Astrophysical Journal*, *925*, L9. https://doi.org/10.3847/2041-8213/ac4b5e

Fischer, D. A., Howard, A. W., Laughlin, G. P., Macintosh, B., Mahadevan, S., Sahlmann, J., & Yee, J. C. (2014). *Exoplanet Detection Techniques* (pp. 715-737). https://doi.org/10.2458/azu_uapress_9780816531240-ch031

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381-395. https://doi.org/10.1145/358669.358692

Glen, S. (2019, septiembre 3). *ROC Curve Explained in One Picture—DataScienceCentral.com*. Data Science Central. https://www.datasciencecentral.com/roc-curve-explained-in-one-picture/

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

Google Cloud. (s. f.). *Guía avanzada de Inception v3 | Cloud TPU*. Google Cloud. Recuperado 30 de enero de 2023, de https://cloud.google.com/tpu/docs/inception-v3-advanced?hl=es-419

*Google Collaboratory*. (s. f.). Recuperado 31 de enero de 2023, de https://colab.research.google.com/

Günther, M. N., Zhan, Z., Seager, S., Rimmer, P. B., Ranjan, S., Stassun, K. G., Oelkers, R. J., Daylan, T., Newton, E., Kristiansen, M. H., Olah, K., Gillen, E., Rappaport, S., Ricker, G. R., Vanderspek, R. K., Latham, D. W., Winn, J. N., Jenkins, J. M., Glidden, A., … Ting, E. B. (2020). Stellar Flares from the First *TESS* Data Release: Exploring a New Sample of M Dwarfs. *The Astronomical Journal*, *159*(2), 60. https://doi.org/10.3847/1538-3881/ab5d3a

Hao, X., Su, X., Wang, Z., Zhang, H., & Batushiren, A. (2019). *UNetGAN: A Robust Speech Enhancement Approach in Time Domain for Extremely Low Signal-to-Noise Ratio Condition*. 1786-1790. https://doi.org/10.21437/Interspeech.2019-1567

He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep Residual Learning for Image Recognition*. 770-778. https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html

Ilin, E., Poppenhäger, K., & Alvarado-Gómez, J. D. (2022). Localizing flares to understand stellar magnetic fields and space weather in exo-systems. *Astronomische Nachrichten*, *343*(4), e210111. https://doi.org/10.1002/asna.20210111

Ilin, E., Schmidt, S. J., Davenport, J. R. A., & Strassmeier, K. G. (2019). Flares in open clusters with K2: I. M 45 (Pleiades), M 44 (Praesepe), and M 67. *Astronomy & Astrophysics*, *622*, A133. https://doi.org/10.1051/0004-6361/201834400

Ilin, E., Schmidt, S. J., Poppenhäger, K., Davenport, J. R. A., Kristiansen, M. H., & Omohundro, M. (2022). *Ekaterinailin/AltaiPony* [Jupyter Notebook]. https://github.com/ekaterinailin/AltaiPony/blob/319cf3b5a685ae3fbdc3befb51031c7fc76cd637/notebooks/01_Getting_Started.ipynb

Julin, F. (2019). *Vision based facial emotion detection using deep convolutional neural networks*. https://doi.org/10.13140/RG.2.2.12567.62881

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., … Hassabis, D. (2021).

Highly accurate protein structure prediction with AlphaFold. *Nature*, *596*(7873), Art. 7873. https://doi.org/10.1038/s41586-021-03819-2

KeepCoding, R. (2020, noviembre 11). *¿Qué son las Redes Neuronales Convolucionales? | KeepCoding Tech School*. https://keepcoding.io/blog/redes-neuronales-convolucionales/

KeepCoding, R. (2022, agosto 22). *¿Qué es la convolución? | KeepCoding Tech School*. https://keepcoding.io/blog/que-es-la-convolucion/

Khan Academy. (s. f.). *Neuron action potentials: The creation of a brain signal (article)*. Khan Academy. Recuperado 22 de enero de 2023, de https://www.khanacademy.org/test-prep/mcat/organ-systems/neuron-membrane-potentials/a/neuron-action-potentials-the-creation-of-a-brain-signal

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, *25*. https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

Lorente, Ò., Riera, I., & Rana, A. (2021). *Image Classification with Classic and Deep Learning Techniques* (arXiv:2105.04895). https://doi.org/10.48550/arXiv.2105.04895

Luong, M.-T., Pham, H., & Manning, C. D. (2015). *Effective Approaches to Attention-based Neural Machine Translation* (arXiv:1508.04025). https://doi.org/10.48550/arXiv.1508.04025

NASA. (2013). *Light Curves—Introduction*. https://imagine.gsfc.nasa.gov/science/toolbox/timing1.html#

Oord, A. van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016). *WaveNet: A Generative Model for Raw Audio* (arXiv:1609.03499; Versión 2). arXiv. http://arxiv.org/abs/1609.03499

Ricker, G. R., Winn, J. N., Vanderspek, R., Latham, D. W., Bakos, G. Á., Bean, J. L., Berta-Thompson, Z. K., Brown, T. M., Buchhave, L., Butler, N. R., Butler, R. P., Chaplin, W. J., Charbonneau, D., Christensen-Dalsgaard, J., Clampin, M., Deming, D., Doty, J., De Lee, N., Dressing, C., … Villasenor, J. (2015). Transiting Exoplanet Survey Satellite (TESS). *Journal of Astronomical Telescopes, Instruments, and Systems*, *1*, 014003. https://doi.org/10.1117/1.JATIS.1.1.014003

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), Art. 6088. https://doi.org/10.1038/323533a0

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, *115*(3), 211-252. https://doi.org/10.1007/s11263-015-0816-y

Stanford. (2022). *CS231n Convolutional Neural Networks for Visual Recognition*. https://cs231n.github.io/convolutional-networks/

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2016). *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning* (arXiv:1602.07261). https://doi.org/10.48550/arXiv.1602.07261

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). *Going Deeper with Convolutions* (arXiv:1409.4842). arXiv. https://doi.org/10.48550/arXiv.1409.4842

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). *Rethinking the Inception Architecture for Computer Vision* (arXiv:1512.00567). arXiv. https://doi.org/10.48550/arXiv.1512.00567

Tarek, Z., Shams, M., Elshewey, A., El-kenawy, E.-S., Ibrahim, A., Abdelhamid, A., & El-dosuky, M. (2022). Wind Power Prediction Based on Machine Learning and Deep Learning Models. *Computers, Materials and Continua*, *74*, 715-732. https://doi.org/10.32604/cmc.2023.032533

Teimourian, H., & Dimililer, K. (2019). La inteligencia artificial en la detección de radiogalaxias. *Dilemas contemporáneos: Educación, Política y Valores*. https://doi.org/10.46377/dilemas.v30i1.1260

Tu, Z.-L., Wu, Q., Wang, W., Zhang, G. Q., Liu, Z.-K., & Wang, F. Y. (2022). Convolutional Neural Networks for Searching Superflares from Pixel-level Data of the Transiting Exoplanet Survey Satellite. *The Astrophysical Journal*, *935*(2), 90. https://doi.org/10.3847/1538-4357/ac7f2c

Vásquez López, J. P. (2014). RED NEURONAL FEEDFORWARD COMO ESTIMADOR DE PATRONES DE CORRIENTES EN EL INTERIOR DEL PUERTO DE MANZANILLO SUJETO A LA ACCION DE TSUNAMIS. *PUBLICACION TECNICA*, *406*. https://trid.trb.org/view/1366350

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need* (arXiv:1706.03762). https://doi.org/10.48550/arXiv.1706.03762

Vida, K., Bódi, A., Szklenár, T., & Seli, B. (2021). Finding flares in Kepler and TESS data with recurrent deep neural networks. *Astronomy & Astrophysics*, *652*, A107. https://doi.org/10.1051/0004-6361/202141068

Vida, K., & Roettenbacher, R. M. (2018). Finding flares in Kepler data using machine-learning tools. *Astronomy and Astrophysics*, *616*, A163. https://doi.org/10.1051/0004-6361/201833194

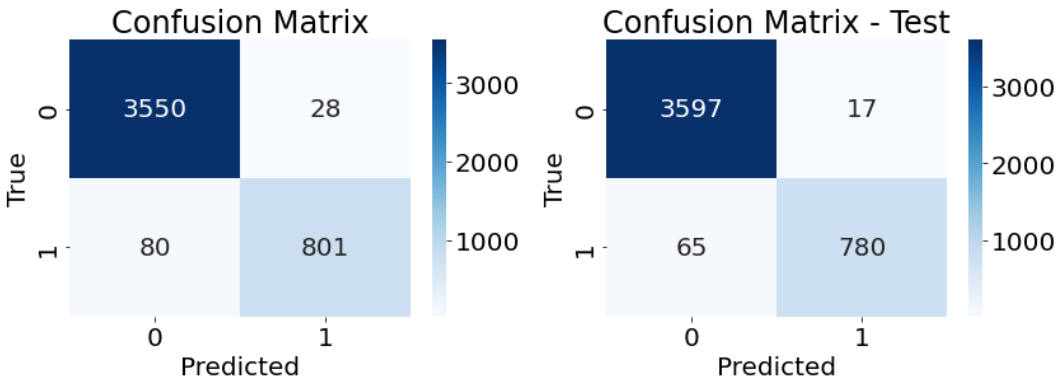
Yu, F., & Koltun, V. (2016). *Multi-Scale Context Aggregation by Dilated Convolutions* (arXiv:1511.07122). https://doi.org/10.48550/arXiv.1511.07122

# Anexo A. Métricas de los modelos del experimento 1

Modelo entrenado con la arquitectura base de Stella

En la parte izquierda podemos observar las métricas obtenidas en el conjunto de validación por el modelo entrenado con la arquitectura Stella y en la parte derecha las métricas obtenidas en el conjunto de test.

Matriz de confusión

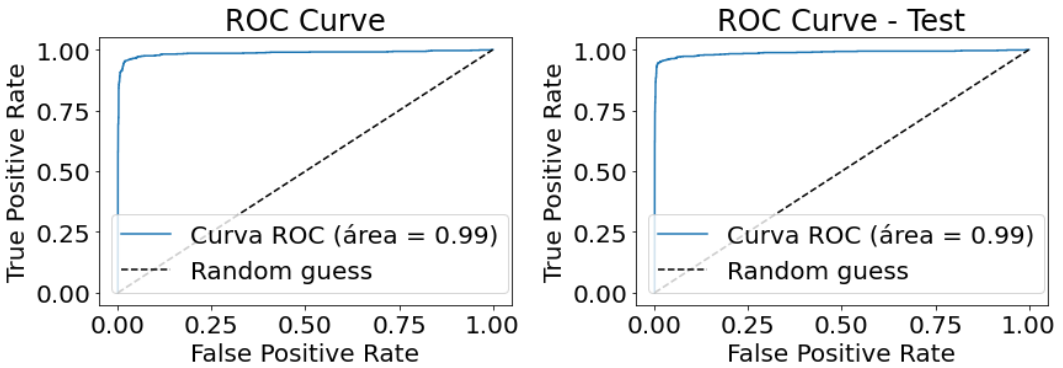



Métricas

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.98 | 0.99 | 0.99 | 3578 |
| 1 | 0.97 | 0.91 | 0.94 | 881 |
| accuracy | | | 0.98 | 4459 |
| macro avg | 0.97 | 0.95 | 0.96 | 4459 |
| weighted avg | 0.98 | 0.98 | 0.98 | 4459 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.98 | 1.00 | 0.99 | 3614 |
| 1 | 0.98 | 0.92 | 0.95 | 845 |
| accuracy | | | 0.98 | 4459 |
| macro avg | 0.98 | 0.96 | 0.97 | 4459 |
| weighted avg | 0.98 | 0.98 | 0.98 | 4459 |

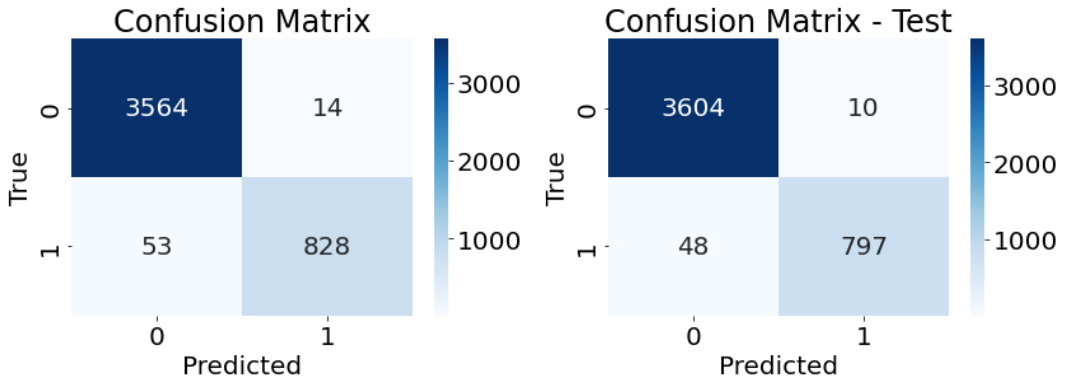
Área bajo la curva ROC-AUC



Modelo entrenado con la arquitectura InceptionV1

En la parte izquierda podemos observar las métricas obtenidas en el conjunto de validación por el modelo entrenado con la arquitectura InceptionV1 y en la parte derecha las métricas obtenidas en el conjunto de test.

Matriz de confusión

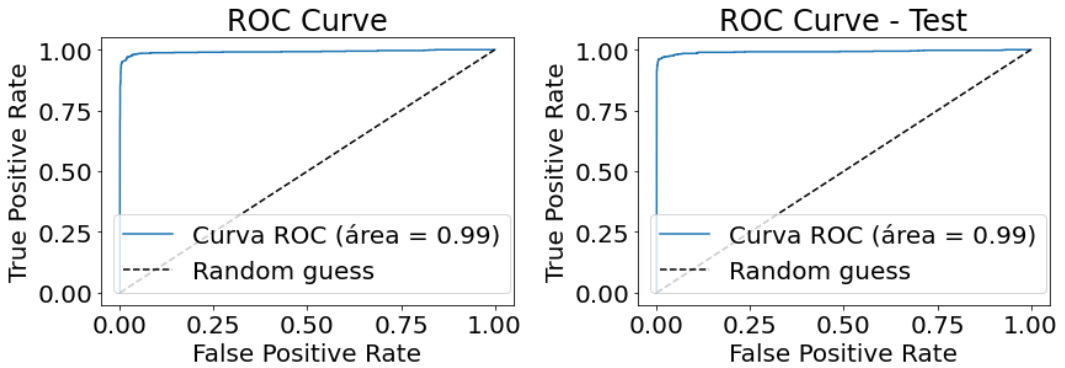



Métricas

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 1.00 | 0.99 | 3578 |
| 1 | 0.98 | 0.94 | 0.96 | 881 |
| accuracy | | | 0.98 | 4459 |
| macro avg | 0.98 | 0.97 | 0.98 | 4459 |
| weighted avg | 0.98 | 0.98 | 0.98 | 4459 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 1.00 | 0.99 | 3614 |
| 1 | 0.99 | 0.94 | 0.96 | 845 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.99 | 0.97 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

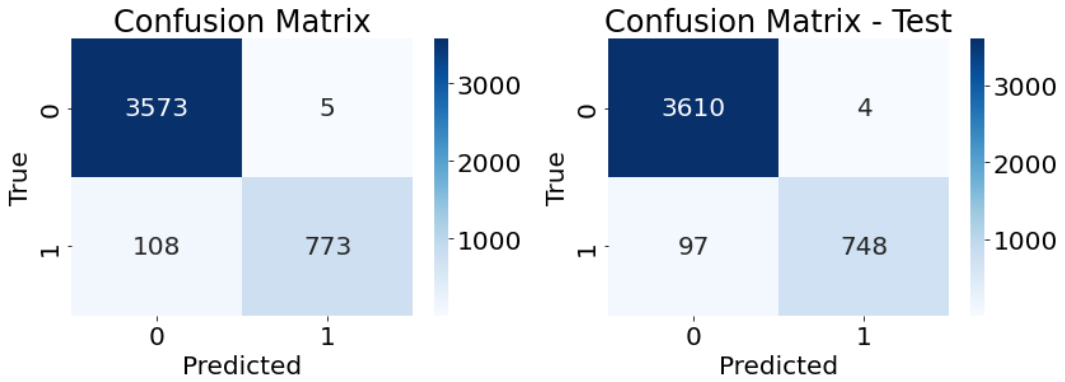
Área bajo la curva ROC-AUC



Modelo entrenado con la arquitectura InceptionV2

En la parte izquierda podemos observar las métricas obtenidas en el conjunto de validación por el modelo entrenado con la arquitectura InceptionV2 y en la parte derecha las métricas obtenidas en el conjunto de test.

Matriz de confusión:

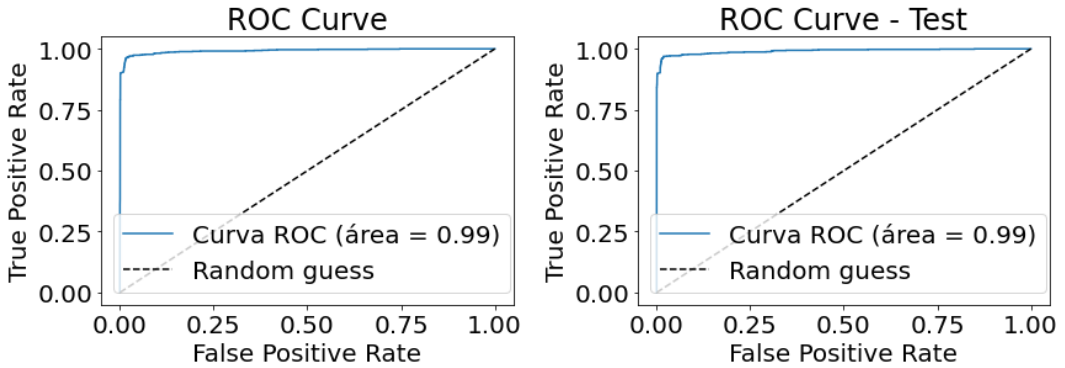



Métricas

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.97 | 1.00 | 0.98 | 3578 |
| 1 | 0.99 | 0.88 | 0.93 | 881 |
| accuracy | | | 0.97 | 4459 |
| macro avg | 0.98 | 0.94 | 0.96 | 4459 |
| weighted avg | 0.98 | 0.97 | 0.97 | 4459 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.97 | 1.00 | 0.99 | 3614 |
| 1 | 0.99 | 0.89 | 0.94 | 845 |
| accuracy | | | 0.98 | 4459 |
| macro avg | 0.98 | 0.94 | 0.96 | 4459 |
| weighted avg | 0.98 | 0.98 | 0.98 | 4459 |

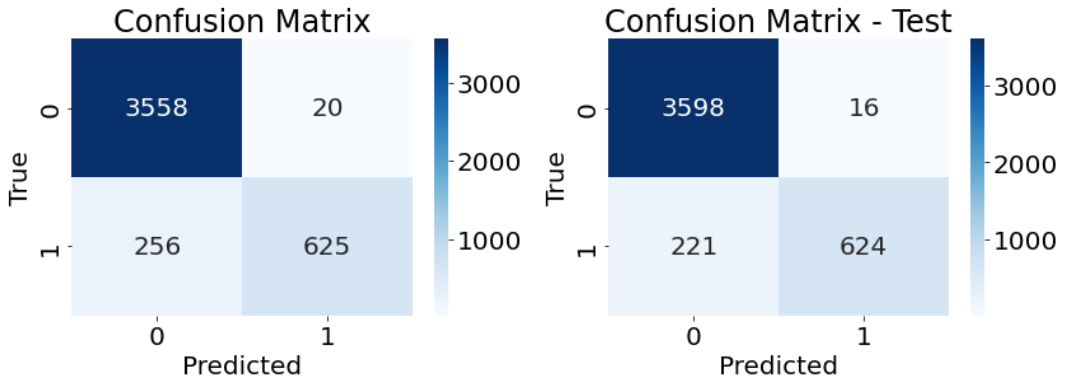
Área bajo la curva ROC-AUC



Modelo entrenado con la arquitectura InceptionV3

En la parte izquierda podemos observar las métricas obtenidas en el conjunto de validación por el modelo entrenado con la arquitectura InceptionV3 y en la parte derecha las métricas obtenidas en el conjunto de test

Matriz de confusión

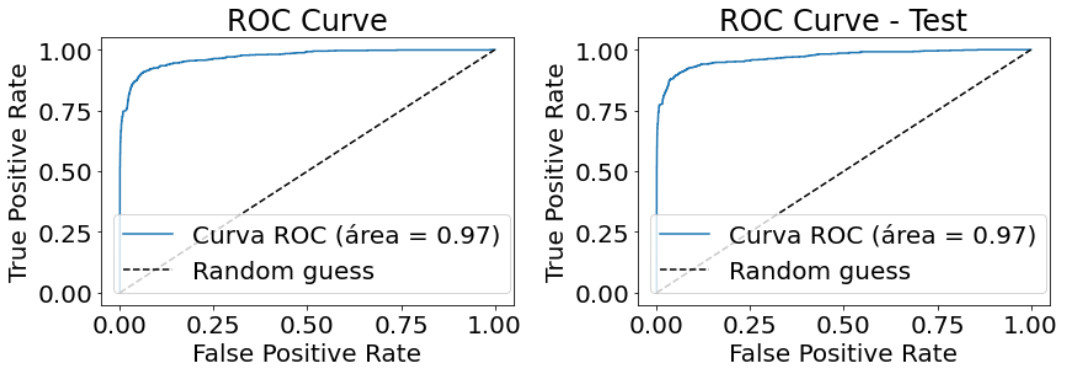



Métricas

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.99 | 0.96 | 3578 |
| 1 | 0.97 | 0.71 | 0.82 | 881 |
| accuracy | | | 0.94 | 4459 |
| macro avg | 0.95 | 0.85 | 0.89 | 4459 |
| weighted avg | 0.94 | 0.94 | 0.93 | 4459 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.94 | 1.00 | 0.97 | 3614 |
| 1 | 0.97 | 0.74 | 0.84 | 845 |
| accuracy | | | 0.95 | 4459 |
| macro avg | 0.96 | 0.87 | 0.90 | 4459 |
| weighted avg | 0.95 | 0.95 | 0.94 | 4459 |

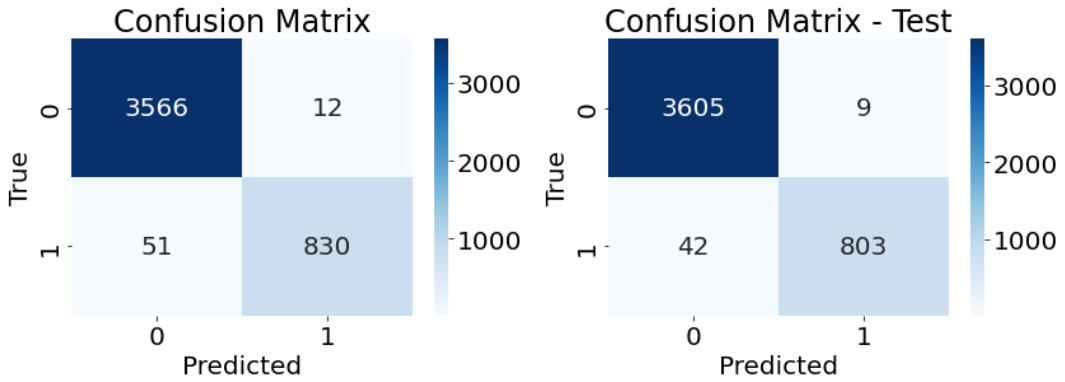
Área bajo la curva ROC-AUC



Modelo entrenado con la arquitectura InceptionV4

En la parte izquierda podemos observar las métricas obtenidas en el conjunto de validación por el modelo entrenado con la arquitectura InceptionV4 y en la parte derecha las métricas obtenidas en el conjunto de test

Matriz de confusión

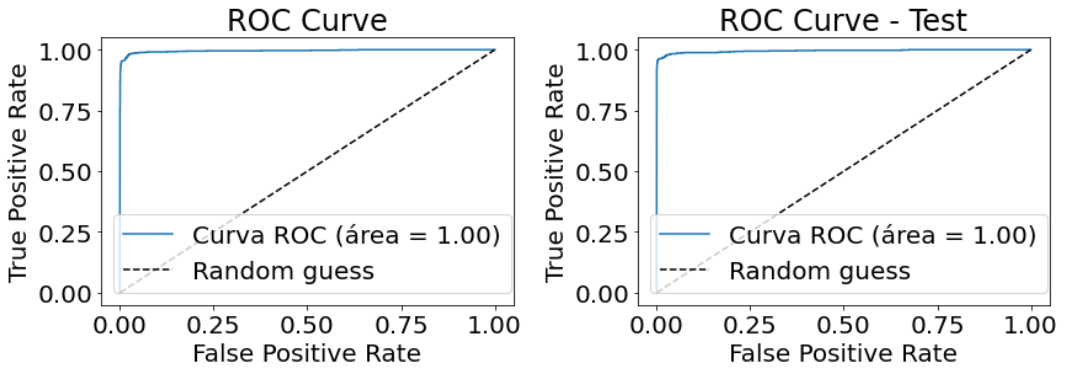



Métricas

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 1.00 | 0.99 | 3578 |
| 1 | 0.99 | 0.94 | 0.96 | 881 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.99 | 0.97 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 1.00 | 0.99 | 3614 |
| 1 | 0.99 | 0.95 | 0.97 | 845 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.99 | 0.97 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

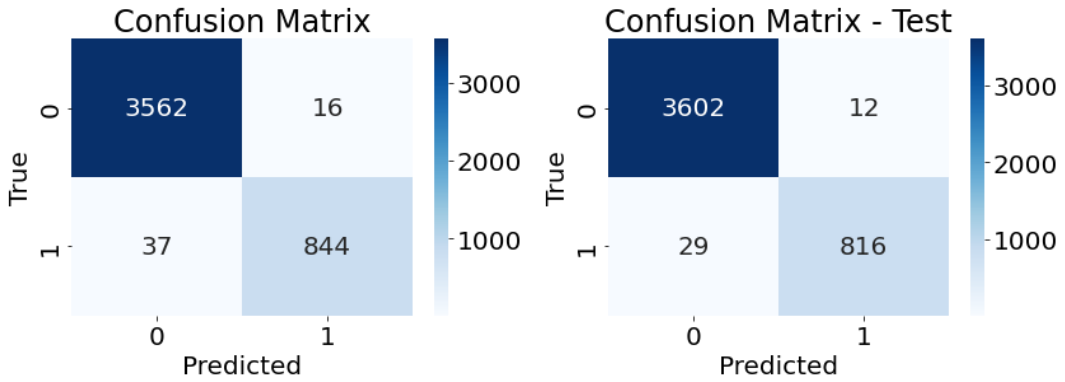
Área bajo la curva ROC-AUC



Modelo entrenado con la arquitectura ResNetV1

En la parte izquierda podemos observar las métricas obtenidas en el conjunto de validación por el modelo entrenado con la arquitectura ResNetV1 y en la parte derecha las métricas obtenidas en el conjunto de test.

Matriz de confusión

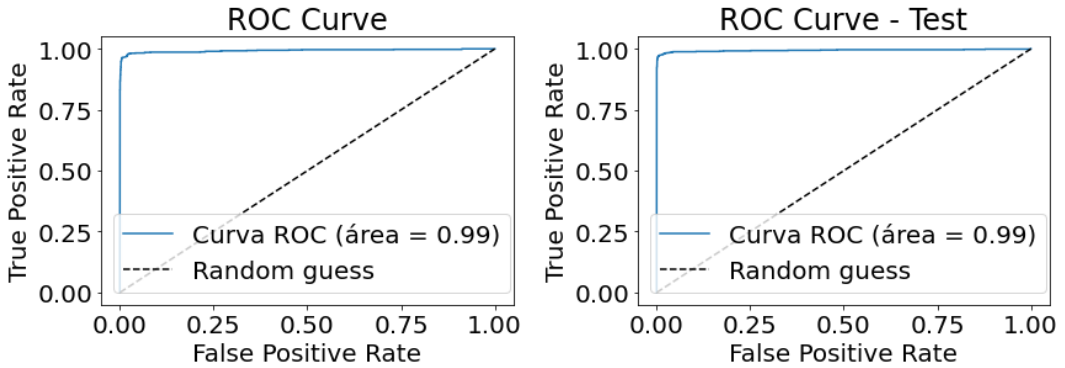



Métricas

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 1.00 | 0.99 | 3578 |
| 1 | 0.98 | 0.96 | 0.97 | 881 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.99 | 0.98 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 1.00 | 0.99 | 3614 |
| 1 | 0.99 | 0.97 | 0.98 | 845 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.99 | 0.98 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

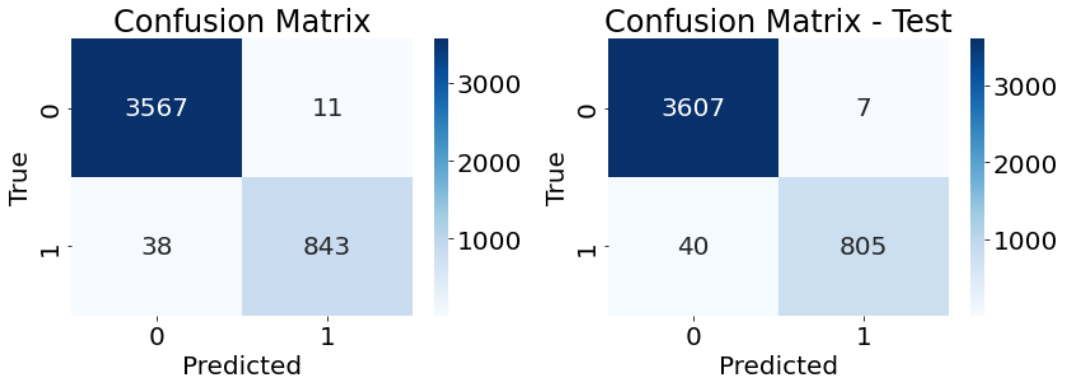
Área bajo la curva ROC-AUC



Modelo entrenado con la arquitectura ResNetV2

En la parte izquierda podemos observar las métricas obtenidas en el conjunto de validación por el modelo entrenado con la arquitectura ResNetV1 y en la parte derecha las métricas obtenidas en el conjunto de test.

Matriz de confusión

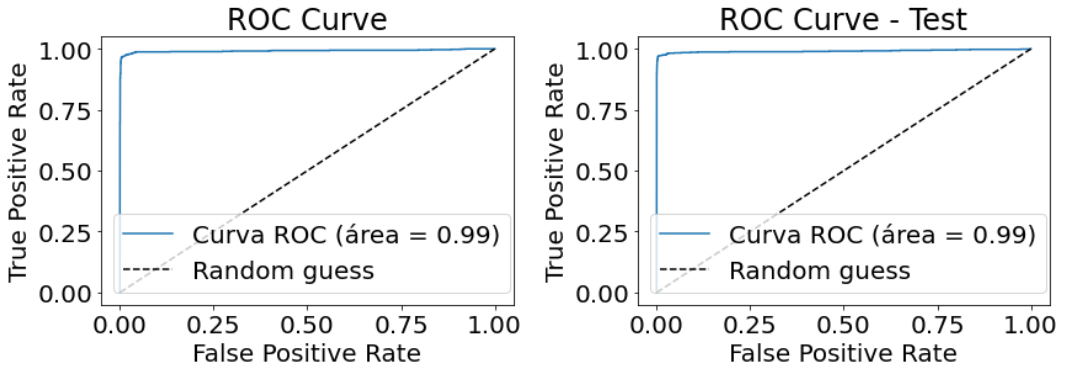



Métricas

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 1.00 | 0.99 | 3578 |
| 1 | 0.99 | 0.96 | 0.97 | 881 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.99 | 0.98 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 1.00 | 0.99 | 3614 |
| 1 | 0.99 | 0.95 | 0.97 | 845 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.99 | 0.98 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

Área bajo la curva ROC-AUC

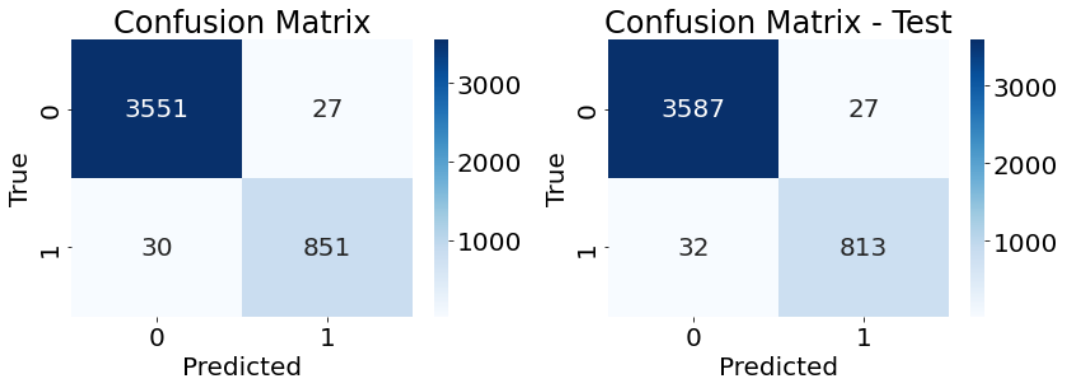


# Anexo B. Métricas de los modelos del experimento 2

Modelo entrenado con convoluciones dilatadas.

En la parte izquierda podemos observar las métricas obtenidas en el conjunto de validación por el modelo entrenado con convoluciones dilatadas y en la parte derecha las métricas obtenidas en el conjunto de test.

Matriz de confusión

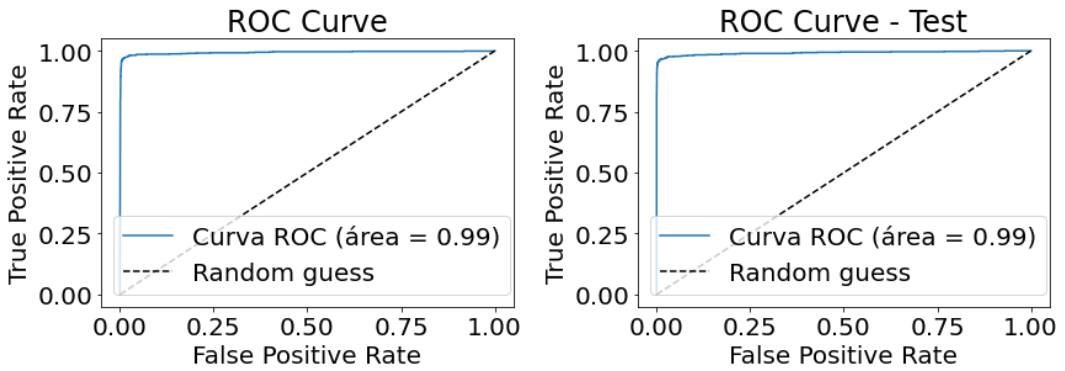



Métricas

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 3578 |
| 1 | 0.97 | 0.97 | 0.97 | 881 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.98 | 0.98 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 3614 |
| 1 | 0.97 | 0.96 | 0.96 | 845 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.98 | 0.98 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

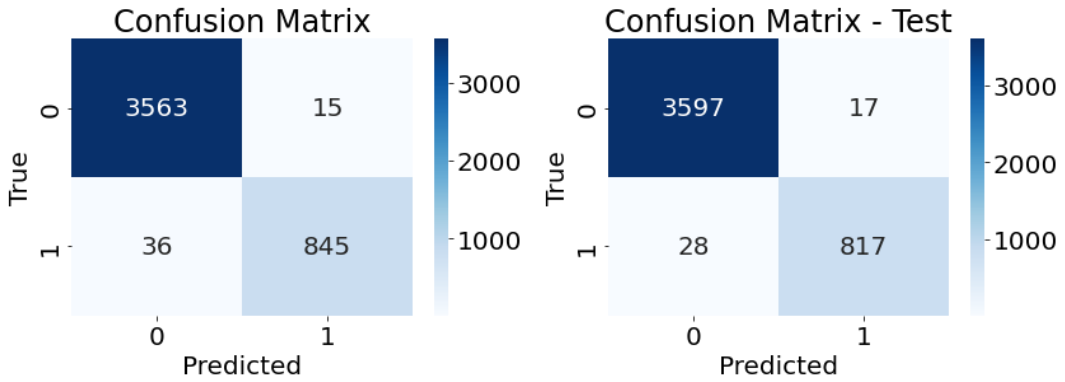
Área bajo la curva ROC-AUC



Modelo entrenado con convoluciones dilatadas + capa de atención.

En la parte izquierda podemos observar las métricas obtenidas en el conjunto de validación por el modelo entrenado con convoluciones dilatadas + capa de atención y en la parte derecha las métricas obtenidas en el conjunto de test.

Matriz de confusión

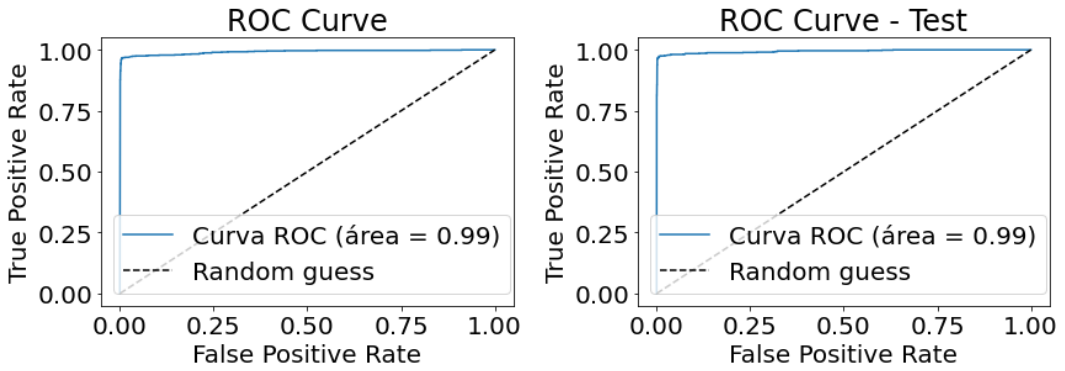



Métricas

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 1.00 | 0.99 | 3578 |
| 1 | 0.98 | 0.96 | 0.97 | 881 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.99 | 0.98 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 1.00 | 0.99 | 3614 |
| 1 | 0.98 | 0.97 | 0.97 | 845 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.99 | 0.98 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

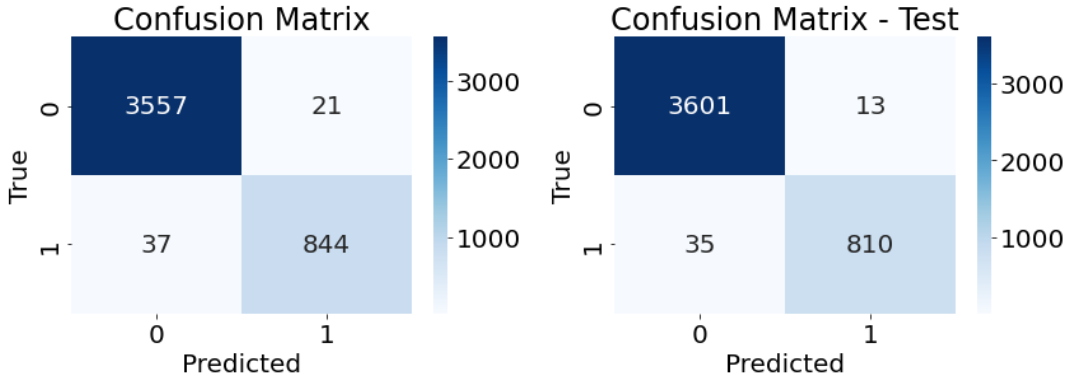
Área bajo la curva ROC-AUC



Modelo entrenado con convoluciones dilatadas + capa de multihead atención.

En la parte izquierda podemos observar las métricas obtenidas en el conjunto de validación por el modelo entrenado con convoluciones dilatadas + capa de multihead atención y en la parte derecha las métricas obtenidas en el conjunto de test.

Matriz de confusión

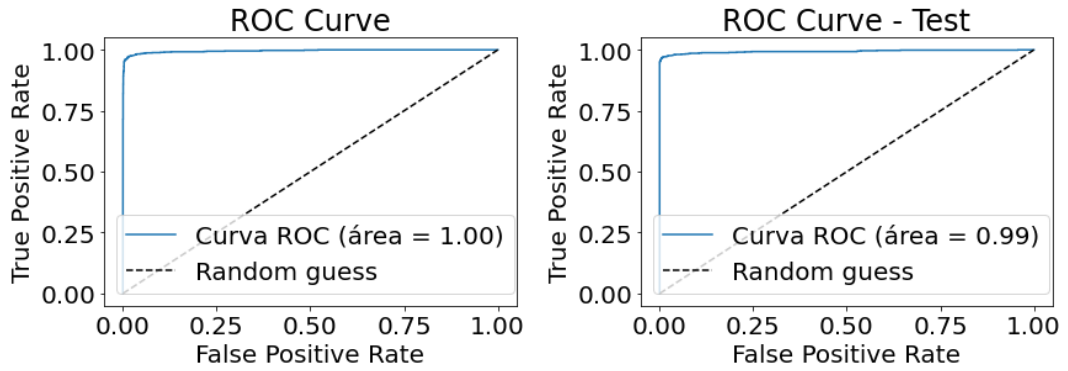



Métricas

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 3578 |
| 1 | 0.98 | 0.96 | 0.97 | 881 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.98 | 0.98 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 1.00 | 0.99 | 3614 |
| 1 | 0.98 | 0.96 | 0.97 | 845 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.99 | 0.98 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

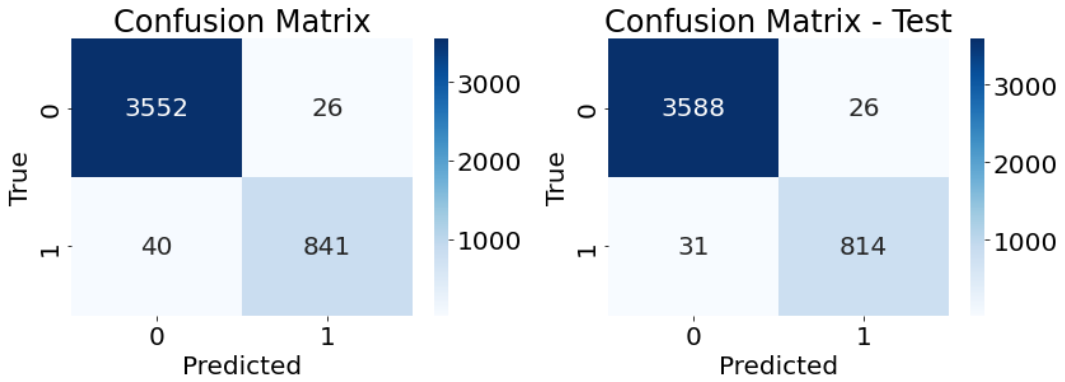
Área bajo la curva ROC-AUC



Modelo entrenado con una red neuronal temporal convolucional (TCN)

En la parte izquierda podemos observar las métricas obtenidas en el conjunto de validación por el modelo entrenado una red neuronal temporal convolucional y en la parte derecha las métricas obtenidas en el conjunto de test.

Matriz de confusión

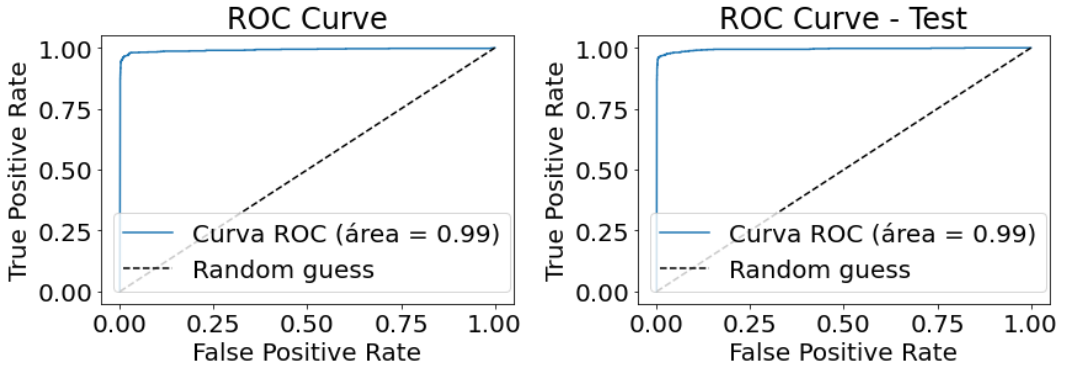



Métricas

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 3578 |
| 1 | 0.97 | 0.95 | 0.96 | 881 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.98 | 0.97 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 3614 |
| 1 | 0.97 | 0.96 | 0.97 | 845 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.98 | 0.98 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

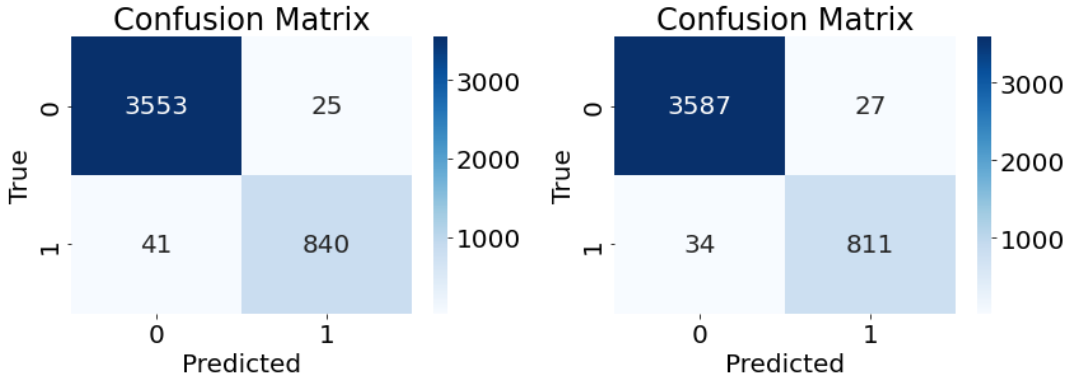
Área bajo la curva ROC-AUC



Modelo entrenado con una red neuronal temporal convolucional (TCN) + capa de atención.

En la parte izquierda podemos observar las métricas obtenidas en el conjunto de validación por el modelo entrenado una red neuronal temporal convolucional + capa de atención y en la parte derecha las métricas obtenidas en el conjunto de test.

Matriz de confusión

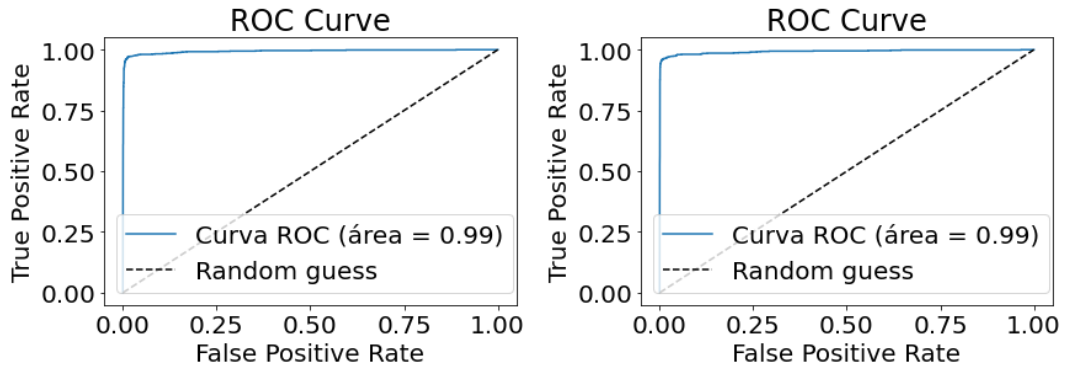



Métricas

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 3578 |
| 1 | 0.97 | 0.95 | 0.96 | 881 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.98 | 0.97 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 3614 |
| 1 | 0.97 | 0.96 | 0.96 | 845 |
| accuracy | | | 0.99 | 4459 |
| macro avg | 0.98 | 0.98 | 0.98 | 4459 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4459 |

Área bajo la curva ROC-AUC



# Anexo C. Código Fuente de las arquitecturas

**Arquitecturas Inceptionv1, Inceptionv2, Inceptionv3, Inceptionv4, ResNetv1 y Resnetv2:**

https://github.com/ansony1691/Inception-ResNet/blob/main/TFM.ipynb

**Arquitectura DilatedConv**

```
input_layer = tf.keras.layers.Input(shape=(200, 1))
x = input_layer
for i in range(10):
      x = tf.keras.layers.Conv1D(32, 3, dilation_rate=2**i,
      padding='same')(x)
      x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dense(1, activation='sigmoid')(x)
model = keras.Model(inputs=input_layer, outputs=x, name='DilatedConv')
```

**Arquitectura DilatedConv +Attention**

```
input_layer = tf.keras.layers.Input(shape=(200, 1))
x = input_layer
for i in range(10):
      x = tf.keras.layers.Conv1D(32, 3, dilation_rate=2**i,
      padding='same')(x)
      x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Attention()([x, x]) # temporal attention layer
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dense(1, activation='sigmoid')(x)
model = keras.Model(inputs=input_layer, outputs=x, name='DilatedConv
+Attention')
```

**Arquitectura DilatedConv + MultiHeadAttention**

```
input_layer = tf.keras.layers.Input(shape=(200, 1))
x = input_layer
for i in range(10):
      x = tf.keras.layers.Conv1D(32, 3, dilation_rate=2**i,
      padding='same')(x)
      x = tf.keras.layers.Activation('relu')(x)
attention = tf.keras.layers.MultiHeadAttention(num_heads=4, key_dim=2)
x = attention(x, x, x) # temporal attention layer
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dense(1, activation='sigmoid')(x)
model = keras.Model(inputs=input_layer, outputs=x, name='DilatedConv +
MultiHeadAttention')
```

**Arquitectura TCN**

```
input_layer = tf.keras.layers.Input(shape=(200, 1))
x = input_layer
for i in range(10):
      x = tf.keras.layers.Conv1D(32, 3, dilation_rate=2**i,
      padding='causal')(x)
      x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dense(1, activation='sigmoid')(x)
model = keras.Model(inputs=input_layer, outputs=x, name='TCN')
```

**Arquitectura TCN+Attention**

```
input_layer = tf.keras.layers.Input(shape=(200, 1))
x = input_layer
for i in range(10):
      x = tf.keras.layers.Conv1D(32, 3, dilation_rate=2**i,
      padding='causal')(x)
      x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Attention()([x, x]) # temporal attention layer
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dense(1, activation='sigmoid')(x)
model = keras.Model(inputs=input_layer, outputs=x, name='TCN+Attention'
)
```

# Anexo D. Artículo de investigación

Al final de la memoria y como un anexo obligatorio deberá incluirse un artículo de investigación que resuma el trabajo realizado y los principales resultados obtenidos. Este artículo deberá seguir la plantilla proporcionada a continuación y tendrá una extensión de entre 6 y 8 páginas. El artículo se podrá desarrollar en español o en inglés (para ello utilizar la plantilla adecuada).

# Rethinking Stella: comparativa de redes neuronales convolucionales para la clasificación de fulguraciones estelares en curvas de luz de la misión TESS.

Ansony Rolando Medina Baca

Universidad Internacional de la Rioja, Logroño (España)

01/03/2023

## Resumen

Este estudio tiene como objetivo mejorar el desempeño del algoritmo Stella en la clasificación de fulguraciones estelares en curvas de luz obtenidas a través de la misión TESS. Para ello, se evaluaron 11 modelos con diferentes arquitecturas de redes neuronales convolucionales (CNN), incluyendo 6 arquitecturas conocidas y 5 nuevas implementaciones. Los resultados demuestran que los 5 modelos propuestos en el segundo experimento mejoran significativamente las métricas de accuracy, precision y recall, logrando una mejora de 2.1%, 3.02% y 7.1% respectivamente en el conjunto de validación. Además, se obtuvo una mejora de 5.4% en la clasificación de verdaderos positivos en el conjunto de validación y 4.42% en el conjunto de test según sus matrices de confusión. Además, las arquitecturas optimizadas permiten reducir en un 67.2% el número de parámetros entrenables con respecto al algoritmo Stella.

## Palabras Clave

Astrofísica, CNN, curvas de luz, Fulguraciones Estelares, TESS.

## I. Introducción

La clasificación de estas curvas de luz es esencial para conocer cuan activa es una estrella a partir de sus fulguraciones. Tales fenómenos, que se acontecen en la fotosfera de la estrella, pueden tener importantes implicaciones sobre la composición de la atmosfera de los planetas que orbitan esas estrellas activas.

En la actualidad, existen diversas herramientas para detectar fulguraciones estelares basadas en métodos estadísticos que detectan valores atípicos, pero estos requieren ajustes manuales por parte del usuario. Por lo tanto, es crucial automatizar el proceso de clasificación para mejorar la eficiencia y la precisión de los resultados.

El algoritmo Stella [5] es uno de los primeros intentos para solucionar la problemática de la clasificación automática de fulguraciones estelares. Basado en redes neuronales convolucionales, reduce la interacción entre el usuario y el ajuste manual de parámetros, lo que permite una clasificación más eficiente en curvas de luz.

En la actualidad las redes neuronales y algoritmos de aprendizaje automático han invadido el campo de la química, física y astronomía para dar solución a diversos problemas, que puede incluir desde predecir la estructura de una proteína, como lo hace AlphaFold [11], hasta la detección de radiogalaxias [18].

De la misma forma la inteligencia artificial está siendo utilizada para la búsqueda y clasificación de exoplanetas. En este trabajo de fin de máster, se aborda el uso de la inteligencia artificial para la clasificación de fulguraciones estelares a partir de curvas de luz proporcionadas por la misión TESS de la NASA [14]. Se compararán diferentes arquitecturas basadas en redes neuronales para alcanzar una solución eficaz y automatizada a la problemática. La información para el entrenamiento será obtenida a partir del catálogo de fulguraciones TESS creado por Günther et al. [8].

## II. Estado del Arte

Las fulguraciones estelares son erupciones de energía de alta intensidad que se producen en la superficie de una estrella. Estas erupciones son causadas por la liberación repentina y masiva de energía en forma de luz, radiación y partículas cargadas. Las fulguraciones estelares son eventos temporales, generalmente duran unos minutos, aunque en algunos casos pueden durar horas.

Hay varios enfoques existentes para la detección de fulguraciones estelares utilizando curvas de luz. Como Flares By-Eye [4] que incorpora algoritmos de muchos autores y el trabajo de múltiples proyectos de tesis doctorales. El algoritmo FBEYE es una técnica de detección manual, lo que significa que requiere la intervención humana para seleccionar eventos transitorios y calcular los parámetros de las fulguraciones estelares detectadas.

Appaloosa [3] es otro algoritmo escrito en Python para la clasificación de fulguraciones estelares en curvas de luz proporcionadas por la misión Kepler [2]. AltaiPony [10] desarrollado en Python, es una caja de herramientas para estudios estadísticos de fulguraciones.

FLATW'RM [21] es un modelo de inteligencia artificial entrenado con el algoritmo RANdom SAmple Consensus (RANSAC) [6] con curvas de luz proporcionadas por la misión Kepler. RANSAC es un algoritmo de aprendizaje automático que se utiliza para estimar modelos en conjuntos de datos con outliers.

FLATW'RM2 [20] es un modelo de inteligencia artificial entrenado con una arquitectura RNN (Red Neuronal Recurrente) para la clasificación de fulguraciones estelares en curvas de luz proporcionadas por la misión Kepler y TESS.

Stella [5] es un paquete open-source de Python para identificar fulguraciones en los datos de dos minutos de TESS con redes neuronales convolucionales (CNN).

## III. Objetivos y Metodología

El objetivo general de este estudio es Optimizar el algoritmo Stella para clasificar fulguraciones estelares en curvas de luz proporcionadas por la misión TESS utilizando nuevas arquitecturas de redes neuronales obteniendo una mejora en la clasificación de verdaderos positivos con un menor coste computacional.

Para alcanzar el objetivo general, ha sido necesario alcanzar los siguientes objetivos específicos:

- Identificar y explorar las principales herramientas utilizadas para clasificar fulguraciones estelares en curvas de luz y el tipo de técnicas que emplean.
- Analizar y profundizar en el entendimiento del algoritmo Stella.
- Modificar y Entrenar el algoritmo Stella con otras arquitecturas de redes neuronales: Inception [16], ResNet [15], basados en convoluciones dilatadas [22], basadas en redes convolucionales temporales (TCN) [13] y Redes Neuronales convolucionales con mecanismos de atención [19].
- Evaluar el modelo utilizando las métricas: Accuracy, precisión, recall, F1-score, ROC-AUC y coste computacional.

Para lograr los objetivos específicos implica el desarrollo exitoso de 4 fases:

Fase 1: comprensión del algoritmo Stella y su funcionamiento: Esta fase se llevó a cabo mediante la instalación del módulo Stella para entender su funcionamiento a nivel usuario, posteriormente se clonó el repositorio de github para ahondar su funcionamiento a nivel código y la solución de algunos bugs ocasionados por la actualización de algunas dependencias.

Fase 2: revisión del estado del arte en la clasificación automática de fulguraciones estelares en curvas de luz: la revisión del estado del arte en la clasificación automática de fulguraciones estelares en curvas de luz nos brinda una comprensión del problema que Stella busca resolver, enfatizando la importancia de contar con un sistema automático para clasificar estas fulguraciones estelares.

Fase 3: implementación y evaluación de arquitecturas conocidas: la implementación y evaluación de las arquitecturas de redes neuronales conocidas como Inception en sus cuatro versiones y ResNet en sus dos versiones permitió ampliar nuestro conocimiento sobre la evolución histórica de las redes neuronales convolucionales. La evaluación de estas arquitecturas permitió identificar sus fortalezas y debilidades, y especialmente la aplicabilidad de redes tan complejas en la tarea de clasificación de fulguraciones estelares en curvas de luz.

Fase 4: desarrollo de nuevas arquitecturas: Durante esta fase, nos enfocamos en la investigación de soluciones innovadoras para abordar el problema de la clasificación de fulguraciones estelares en curvas de luz. La implementación de capas especializadas para ampliar el campo receptivo de las convoluciones, aprovechar el componente temporal presente en las curvas de luz y mecanismos de atención que permiten a la red generalizar de manera más efectiva, son algunos de los aspectos que se abordaron en esta fase. Estos esfuerzos permitieron obtener mejoras significativas en las métricas de evaluación de la red con un menor coste computacional.

## IV. Contribución

Dentro del dominio de la inteligencia artificial existen 4 principales trabajos publicados para la detección de fulguraciones estelares utilizando curvas de luz. El modelo de Vida & Roettenbacher [21] entrenado con el algoritmo RANSAC para la detección de outliers en curvas de luz proporcionadas por la misión Kepler; el modelo de Vida et al. [20] que está entrenado con una arquitectura de redes neuronales recurrentes aplicado en curvas de luz proporcionadas por las misiones Kepler y TESS; y el algoritmo Stella por Feinstein, Montet, & Ansdel [5] entrenado con una arquitectura basada en redes neuronales convolucionales 1D aplicados a curvas de luz proporcionados por la misión TESS.

En esta comparativa se plantea clasificar fulguraciones estelares en curvas de luz mediante redes neuronales convolucionales 1D por lo que nuestro punto de referencia será el algoritmo Stella.

El algoritmo Stella, no solamente consta de la arquitectura de la CNN. Este algoritmo posee un ecosistema de descarga, preprocesamiento, visualización, etc. Por lo que será nuestra base para el entrenamiento de nuevos modelos con arquitecturas distintas a las propuestas por Feinstein, Montet, & Ansdel [5].

Se plantea evaluar 11 arquitecturas distintas, enumeradas a continuación:

- Inceptionv1 1D
- Inceptionv2 1D
- Inceptionv3 1D
- Inceptionv4 1D
- ResNetv1 1D
- ResNetv2 1D
- Arquitectura con convoluciones dilatadas
- Arquitectura con convoluciones dilatadas + capa de atención
- Arquitectura con convoluciones dilatadas + capa multihead de atención
- TCN
- TCN + capa de atención

Para evaluar cada modelo se utilizarán distintas métricas como:

- Matriz de confusión
- Accuracy
- Precision
- Recall

- F1-score
- Curva ROC
- Cantidad de parámetros

Se evaluarán 3 estados del modelo:

- Entrenamiento
- Validación
- Test

Para la implementación del primer experimento recurrimos al código de las 4 versiones de Inception y las dos versiones de ResNet adaptados para el tratamiento de señales unidimensionales desarrollado por Mahmud[1].

Inceptionv1(GoogLeNet) [16]: está construida con una estructura modular, donde cada módulo Inception combina diferentes tipos de capas y operaciones para extraer diferentes características de la imagen de entrada. Estos módulos se organizan en varios bloques de Inception, que a su vez están conectados en una estructura jerárquica. Además, GoogLeNet utiliza una arquitectura de subredes donde cada subred se enfoca en extraer características a diferentes escalas de la imagen. La salida de todas las subredes se combina y se pasa a una capa final de clasificación para producir la predicción.

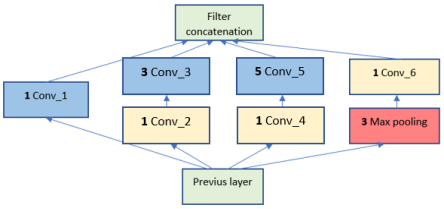



Figura 1: Módulo Inceptionv1 con reducción de dimensiones adaptado a señales unidimencionales

La arquitectura Inceptionv1 posee 9 módulos inception como los descritos en la Figura 1. Variando solamente la cantidad de kernels en cada capa de convolución. La cantidad de kernels están descritas en la Tabla 1.

Tabla 1

| Inception module | Conv _1 | Conv _2 | Conv _3 | Conv _4 | Conv _5 | Conv _6 |
|---|---|---|---|---|---|---|
| bloque 1 | 64 | 96 | 128 | 16 | 32 | 32 |
| bloque 2 | 128 | 128 | 192 | 32 | 96 | 64 |
| bloque 3 | 192 | 96 | 208 | 16 | 48 | 64 |
| bloque 4 | 160 | 112 | 224 | 24 | 64 | 64 |
| bloque 5 | 128 | 128 | 256 | 24 | 64 | 64 |
| bloque 6 | 112 | 144 | 288 | 32 | 64 | 64 |
| bloque 7 | 256 | 160 | 320 | 32 | 128 | 128 |
| bloque 8 | 256 | 160 | 320 | 32 | 128 | 128 |
| bloque 9 | 384 | 192 | 348 | 48 | 128 | 128 |

Cantidad de kernels en cada capa de convolución de la arquitectura Inceptionv1.

[1] https://github.com/Sakib1263/Inception-InceptionResNet-SEInception-SEInceptionResNet-1D-2D-Tensorflow-Keras

Inceptionv2 [17]: Para la versión posterior de Inception, hemos realizado modificaciones en el módulo Inception y se ha agregado un bloque de reducción a la red. La modificación más significativa es que se ha reemplazado una convolución con un kernel de tamaño 5x5 en una de las ramas, por dos convoluciones de tamaño 3x3. Esta actualización ayuda a optimizar la red al reducir la cantidad de parámetros que necesitan ser entrenados.

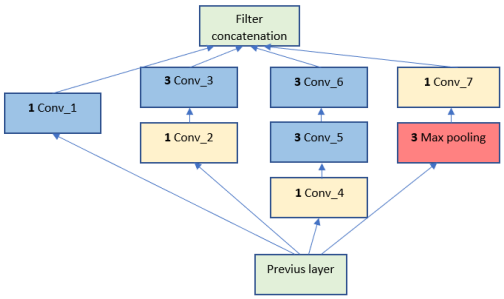



Figura 2: Módulo Inceptionv2 con reducción de dimensiones adaptado a señales unidimencionales.

La arquitectura Inceptionv2 posee 8 módulos inception como los descritos en la Figura 2. Variando solamente la cantidad de kernels en cada capa de convolución. La cantidad de kernels están descritas en la Tabla 2.

Tabla 2

| Inception module | Conv _1 | Conv _2 | Conv _3 | Conv _4 | Conv _5 | Conv _6 | Conv _7 |
|---|---|---|---|---|---|---|---|
| bloque 1 | 64 | 64 | 64 | 64 | 96 | 96 | 32 |
| bloque 2 | 64 | 64 | 96 | 64 | 96 | 96 | 64 |
| bloque 3 | 224 | 64 | 96 | 96 | 128 | 128 | 128 |
| bloque 4 | 196 | 96 | 128 | 96 | 128 | 128 | 128 |
| bloque 5 | 160 | 128 | 160 | 128 | 160 | 160 | 96 |
| bloque 6 | 96 | 128 | 196 | 160 | 192 | 192 | 96 |
| bloque 7 | 352 | 192 | 320 | 160 | 224 | 224 | 128 |
| bloque 8 | 352 | 192 | 320 | 192 | 224 | 224 | 128 |

Cantidad de kernels en cada capa de convolución de la arquitectura Inceptionv2.

La arquitectura Inceptionv2 incluye un módulo para reducir las dimensiones del mapa de características.

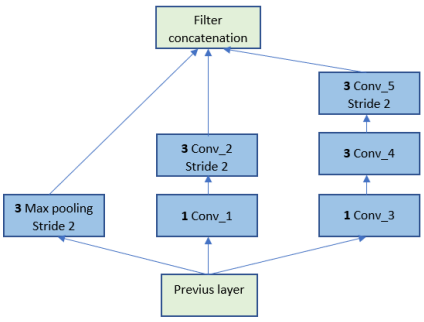



Figura 3: Módulo reduction block adaptado a señales unidimensionales.

La cantidad de kernels en cada capa de convolución del bloque de reducción están descritos en la Tabla 3.

Tabla 3

| Reduction Block | Conv_1 | Conv_2 | Conv_3 | Conv_4 | Conv_5 |
|---|---|---|---|---|---|
| Bloque 1 | 128 | 160 | 64 | 96 | 96 |
| Bloque 2 | 128 | 192 | 192 | 256 | 256 |

Inceptionv3 [17]: esta arquitectura contiene 3 módulos distintos de Inception, debido a que esta arquitectura está pensada para solucionar problemas bidimensionales. Algunos componentes de la arquitectura no podrán ser adaptados a una señal unidimensional.

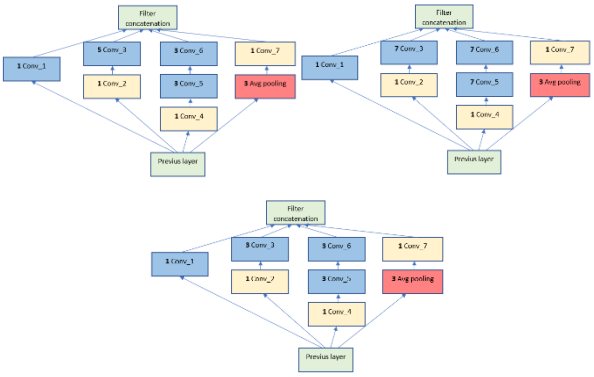


Figura 4: módulos Inception de la arquitectura Inceptionv3 adaptados a señales unidimensionales. Inception_A (arriba izquierda), Inception_B (arriba derecha) e Inception_C (abajo).

La cantidad de kernels en cada capa de convolución del bloque de Inception están descritos en la Tabla 4.

Tabla 4

| Inception module | Conv _1 | Conv _2 | Conv _3 | Conv _4 | Conv _5 | Conv _6 | Conv _7 |
|---|---|---|---|---|---|---|---|
| Incep_A 1 | 64 | 48 | 64 | 64 | 96 | 96 | 32 |
| Incep_A 2 | 64 | 48 | 64 | 64 | 96 | 96 | 64 |
| Incep_A 3 | 64 | 48 | 64 | 64 | 96 | 96 | 64 |
| Incep_B 1 | 192 | 128 | 192 | 128 | 128 | 192 | 192 |
| Incep_B 2 | 192 | 160 | 192 | 160 | 160 | 192 | 192 |
| Incep_B 3 | 192 | 160 | 192 | 160 | 160 | 192 | 192 |
| Incep_B 4 | 192 | 192 | 192 | 192 | 192 | 192 | 192 |
| Incep_C 1 | 320 | 384 | 384 | 448 | 384 | 384 | 192 |
| Incep_C 2 | 320 | 384 | 384 | 448 | 384 | 384 | 192 |

Esta arquitectura incluye 2 tipos de bloques de reducción.

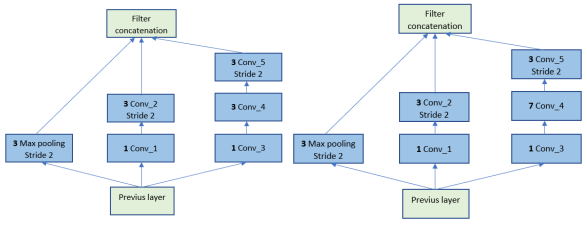


Figura 5: Reduction_Block_A (izquierda) y Reduction_Block_A (derecha) adaptados a señales unidimensionales.

Inceptionv4 [15]: La principal diferencia entre la arquitectura de Inceptionv4 y su predecesor Inceptionv3 radica en la base de la arquitectura, desde la entrada hasta el primer módulo de Inception. Además, otra diferencia notable es la cantidad de módulos de cada tipo que utiliza la arquitectura.

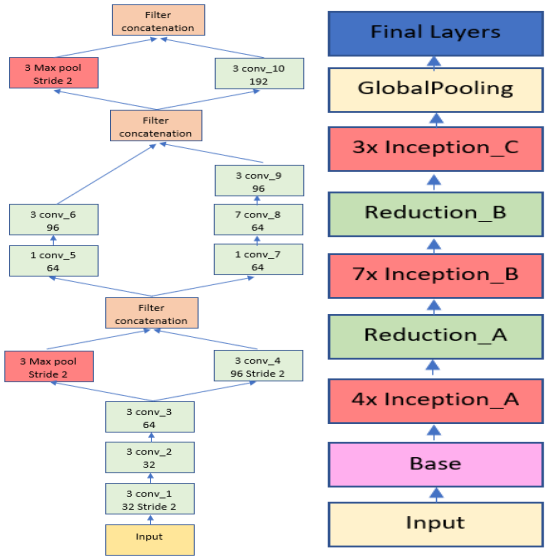



Figura 6: Base de la arquitectura Inceptionv4(izquierda) y arquitectura Inceptionv4 (derecha).

La cantidad de kernels en cada capa de convolución de los módulos Inception y Recuction Block están descritos en la Tabla 5 y Tabla 6 respectivamente.

Tabla 5

| Inception module | Conv _1 | Conv _2 | Conv _3 | Conv _4 | Conv _5 | Conv _6 | conv _7 |
|---|---|---|---|---|---|---|---|
| 4 x Inception_A | 96 | 64 | 96 | 64 | 96 | 96 | 96 |
| 7 x Inception_B | 384 | 192 | 256 | 192 | 224 | 256 | 128 |
| 3 x Inception_C | 256 | 384 | 512 | 384 | 512 | 512 | 256 |

Tabla 6

| Reduction_Block | Conv_1 | Conv_2 | Conv_3 | Conv_4 | Conv_5 |
|---|---|---|---|---|---|
| Block_A | 64 | 384 | 192 | 224 | 256 |
| Block_B | 192 | 192 | 256 | 320 | 320 |

ResNetv1 [15]: La arquitectura ResNetV1 cuenta con 3 distintos módulos Inception modificados con una conexión residual. La base de la arquitectura es similar a InceptionV3.

La conexión residual funciona conectando la entrada directamente a la salida a través de una secuencia de operaciones adicionales. En lugar de tener una secuencia compleja de capas, la conexión residual permite que la información original pase directamente a la salida, facilitando la propagación del gradiente y mejorando la capacidad de la red para aprender características complejas.

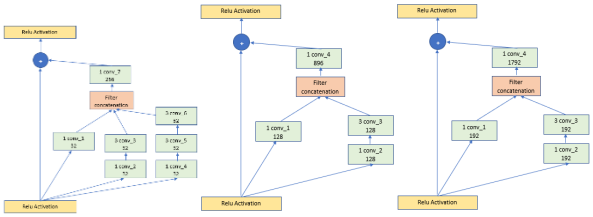


Figura 7: módulos Inception-ResNet-A(izquierda), Inception-ResNet-B(centro) e Inception-ResNet-C (derecha) de la arquitectura ResNetv1 adaptados al procesamiento de señales unidimensionales.

La arquitectura ResNetv1 al igual que su antecesora posee 2 bloques de reducción.

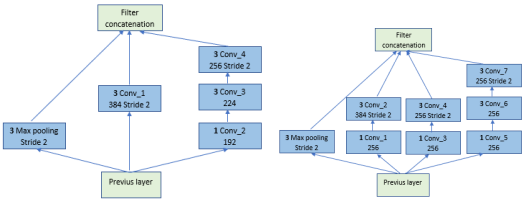


Figura 8: Reduction_Block_A(izquierda) y Reduction_Block_A (derecha) adaptado al procesamiento de señales unidimencionales.

ResNetv2 [15]: posee la misma base que InceptionV4 y sus módulos inception tienen la misma estructura que ResNetV1, variando en la cantidad de kernels en sus capas convolucionales y sobre todo ResNetV2 es una red aún más profunda porque contiene 10 módulos de Inception_ResNet_A, 20 de Inception_ResNet_B y 10 de Inception_ResNet_C.

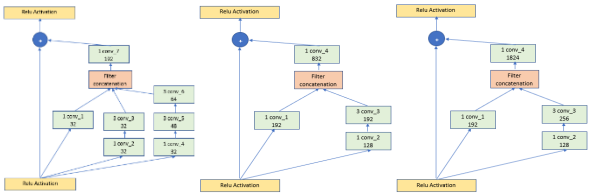


Figura 9: módulo Inception_ResNet_A (Izquierda), Inception_ResNet_B (centro) Inception_ResNet_C (derecha) de la arquitectura ResNetv2 adaptada al procesamiento de señales unidimencionales.

ResNetV2 también utiliza 2 bloques de reducción. El reduction_Block_A es el mismo que el de ResNetV1 y el Reduction_Block_B varía ligeramente la cantidad de kernels en algunas convoluciones.

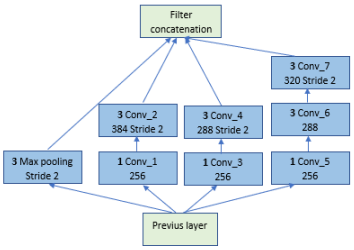



Figura 10: Reduction_Block_B de la arquitectura ResNetv2 adaptada al procesamiento de señales unidimensionales.

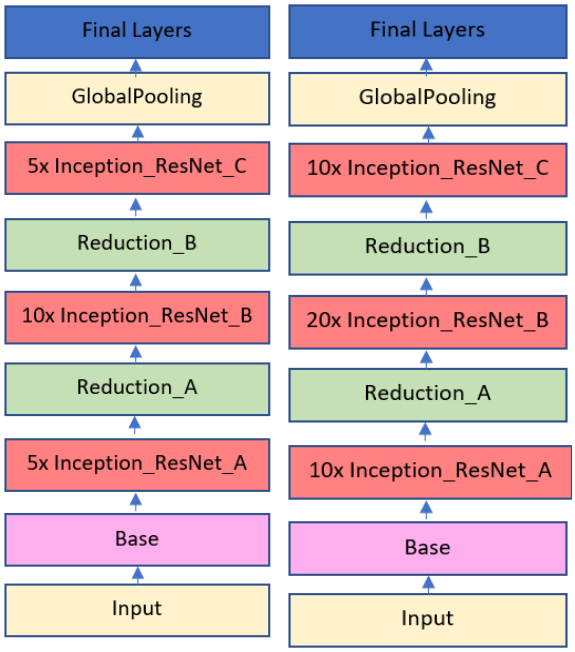



Figura 11: arquitectura ResNetv1 (izquierda) y ResNetv2 (derecha).

Dilated convolutions [22]: Las convoluciones dilatadas también conocidas como ‘à trous convolutions’ son un tipo de operación de convolución en la que se aplica un kernel a una imagen con un "dilatador", que espacia los elementos del kernel ampliando el campo receptivo.

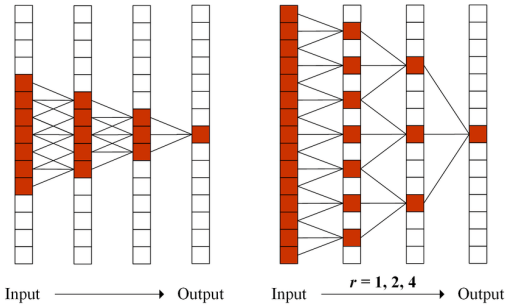



Figura 12: Red neuronal convolucional 1D convencional (izquierda), red neuronal convolucional dilatada 1D con aumento exponencial del dilation_rate en base 2 (derecha) [9].

Causal Convolutional Layers [13]: Las Convoluciones Causales son aquellas que solo dependen del presente y del pasado, pero no del futuro, en el procesamiento de una señal temporal. Esto permite que las salidas de las operaciones sean solo dependientes de las entradas pasadas y no de futuras, lo que es útil en ciertas aplicaciones como la generación de audio o video e incluso series temporales.

Dilated Causal Convolutional Layers [13]: Las capas de Convolución Causal Dilatada combinan las bondades de las Convoluciones Dilatadas y las Convoluciones Causales, aumentando el campo receptivo del kernel sin incluir información futura.

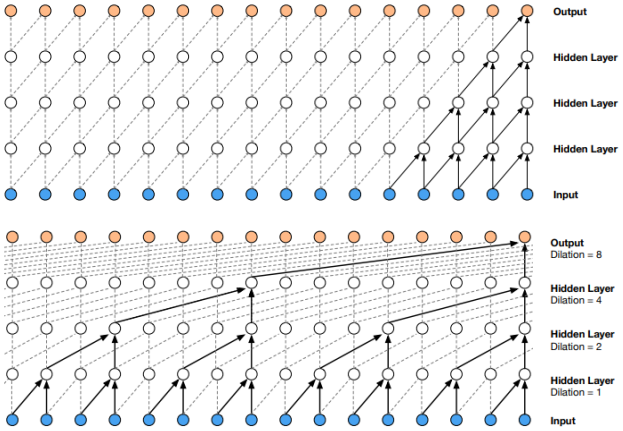



Figura 14: visualización del árbol de capas de una red con operadores convolucionales causales (arriba), Visualización del árbol de capas de una red con operadores de convolución causales dilatados (abajo) también llamada Redes de Convoluciones Temporales (TCNs por sus siglas en inglés) [13].

Attention: Bahdanau, Cho y Bengio [1] propusieron mecanismos de atención para solucionar el problema de información que enfrentaba la arquitectura de RNN codificador-decodificador inicial aplicada a traducción. Si bien fueron los primeros en usar mecanismos de atención, Loung et al. [12] es el primero en proponer distintos mecanismos de atención.

Posteriormente, Vaswani el al. [19] populariza los mecanismos de atención en el artículo "Attention Is All You Need" implementado en la novedosa arquitectura Transformers.

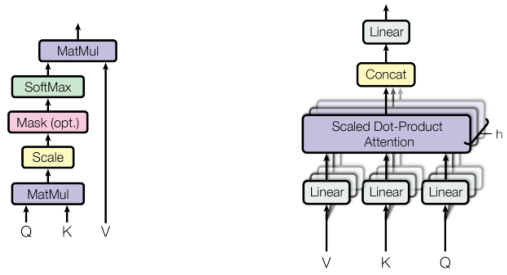



Figura 15: Scaled Dot-Product Attention (izquierda), Multi-Head Attention (derecha) utilizadas en la arquitectura Transformers [19].

El mecanismo de Scaled Dot-Product Attention se puede resumir en la siguiente expresión:

$$Attention(Q, K, V) = softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V$$

Donde Q es el vector Query, K es el vector key, V el vector value y d_k es la dimensión de los vectores. Scaled Dot-Product Attention hace referencia al score function "dot" de Bahdanau, Cho y Bengio [1], donde su principal diferencia es el factor de escalado $\frac{1}{\sqrt{d_k}}$.

Multi-Head Attention utiliza Scaled Dot-Product Attention múltiples veces en paralelo, denominada heads. El mecanismo de Multi-Head Attention se resume en la siguiente expresión:

$$MultiHead(Q, K, V) = Concat(head_1, head_2, \dots, head_h)W^O$$

Donde $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$

Cada $W$ es una matriz de parámetros que transforma los vectores $Q, K$ y V antes de entrar al módulo Scaled Dot-Product Attention.

En el segundo experimento, las capas de convolución dilatada son fundamentales para el éxito de este proyecto debido a su capacidad para aumentar el alcance receptor de la red sin aumentar su complejidad. Además, estas capas son compatibles y fácilmente integrables con las capas de convolución casual y de atención, por lo que serán incluidas en cada arquitectura.

El número de capas de convolución, cantidad de kernels y demás hiper parámetros se determinaron mediante experimentación.

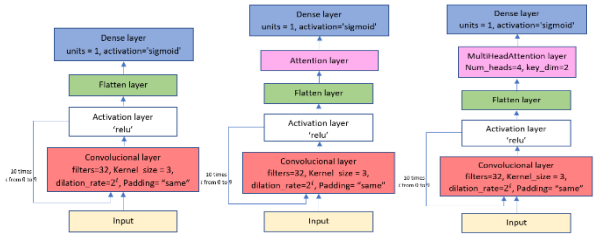



Figura 16: arquitectura DilatedConv (izquierda), DilatedConv+attention (centro), DilatedConv+multihead attention (derecha).

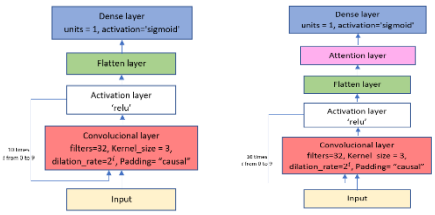



Figura 17: TCN (izquierda), TCN+attention (derecha).

## V. Evaluación y Resultados

En el desarrollo de esta comparativa utilizamos como base el algoritmo Stella en su configuración por defecto:

El algoritmo Stella implementa la división de la base de datos en tres conjuntos a través de la función "split_data": entrenamiento (80%), validación (10%) y prueba (10%). Para garantizar la aleatoriedad en la distribución de los datos, se utiliza la función "do_shuffle" con una semilla fija de 123. Además, el parámetro "frac_balance" se utiliza para equilibrar la relación entre fulguraciones (flares a partir de aquí) y no flares en los datos, con un valor predeterminado de 0.73. Finalmente, el entrenamiento de cada modelo se realiza con 200 cadencias y una semilla aleatoria de 2. Se establece "adam" como el optimizador, se utiliza la función de pérdida "binary_crossentropy" y las métricas seleccionadas son "accuracy", "precision" y "recall". Utilizaremos el TESS Catalog of Stellar Flare [8] para entrenar los modelos.

Para el desarrollo de ambos experimentos se utiliza Google Colab [7] para poder paralelizar los cálculos en una GPU.

### *Primer Experimento*

En el primer experimento comparamos la arquitectura de la red neuronal de Stella con las arquitecturas InceptionV1, InceptionV2, InceptionV3, InceptionV4, ResNetV1 y ResNetV2.

Tabla 6

| | *Conjunto de validación* | | | | |
|---|---|---|---|---|---|
| Modelo | Accuracy | Precision (Flare – Non Flare) | Recall (Flare – Non Flare) | F1-Score (Flare – Non Flare) | ROC |
| Stella | 0.98 | 0.97-0.98 | 0.91-0.99 | 0.94-0.99 | 0.99 |
| Inceptionv1 | 0.98 | 0.98-0.99 | 0.94-1.00 | 0.96-0.99 | 0.99 |
| Inceptionv2 | 0.97 | 0.99-0.97 | 0.88-1.00 | 0.93-0.98 | 0.99 |
| Inceptionv3 | 0.94 | 0.97-0.93 | 0.71-0.99 | 0.82-0.96 | 0.97 |
| Inceptionv4 | 0.99 | 0.99-0.99 | 0.94-1.00 | 0.96-0.99 | 1 |
| ResNetv1 | 0.99 | 0.98-0.99 | 0.96-1.00 | 0.97-0.99 | 0.99 |
| ResNetv2 | 0.99 | 0.99-0.99 | 0.96-1.00 | 0.97-0.99 | 0.99 |

En la Tabla 6 del conjunto de validación, se puede apreciar que cuatro de los modelos evaluados en este experimento han mejorado los resultados de referencia (Stella). Uno de los problemas más graves en el desempeño de Stella era la presencia de falsos positivos. se puede observar que los modelos InceptionV1, InceptionV4, ResNetV1 y ResNetV2 han mejorado las métricas de precisión, sensibilidad y F1, lo cual indica que estos cuatro modelos clasifican de manera más precisa los verdaderos positivos en comparación con Stella.

Tabla 7

| | *Conjunto de validación* | | | | *Conjunto de test* | | | |
|---|---|---|---|---|---|---|---|---|
| Modelo | TP | TN | FP | FN | TP | TN | FP | FN |
| Stella | 801 | 3550 | 28 | 80 | 780 | 3597 | 17 | 65 |
| Inceptionv1 | 828 | 3564 | 14 | 53 | 797 | 3604 | 10 | 48 |
| Inceptionv2 | 773 | 3573 | 5 | 108 | 748 | 3610 | 4 | 97 |
| Inceptionv3 | 625 | 3558 | 20 | 256 | 624 | 3598 | 16 | 221 |
| Inceptionv4 | 830 | 3566 | 12 | 51 | 803 | 3605 | 9 | 42 |
| ResNetv1 | 844 | 3562 | 16 | 37 | 816 | 3602 | 12 | 29 |
| ResNetv2 | 843 | 3567 | 11 | 38 | 805 | 3607 | 7 | 40 |

En la Tabla 7 se observa que todos los modelos propuestos disminuyen los falsos positivos, mientras que los modelos Inceptionv1, Inceptionv4, ResNetv1 y ResNetv2 también logran disminuir los falsos negativos.

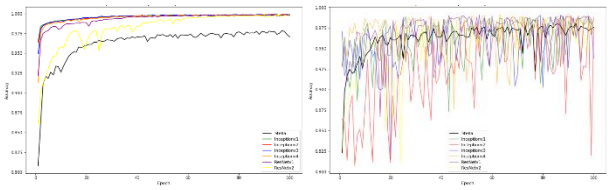


Figura 18: Comparativa del accuracy durante el entrenamiento en el conjunto de entrenamiento (izquierda), Comparativa del accuracy durante el entrenamiento en el conjunto de validación (derecha).

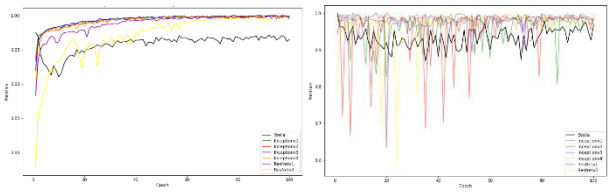


Figura 19: Comparativa de Precision durante el entrenamiento en el conjunto de entrenamiento (izquierda), Comparativa de Precision durante el entrenamiento en el conjunto de validación (derecha).

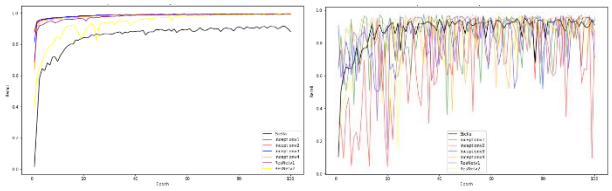


Figura 20: Comparativa de recall durante el entrenamiento en el conjunto de entrenamiento (izquierda), Comparativa del recall durante el entrenamiento en el conjunto de validación (derecha).

Sin embargo. Partiendo de las figuras 18, 19 y 20 podemos observar que todos los modelos se están sobre ajustando al conjunto de entrenamiento, incluso algunos más que Stella.

Tabla 8

| *Modelo* | *accuracy* | *Val accuracy* | *precision* | *Val precision* | *recall* | *Val recall* |
|---|---|---|---|---|---|---|
| Stella | 0.96±0.02 | 0.96±0.02 | 0.96±0.01 | 0.93±0.02 | 0.85±0.12 | 0.87±0.11 |
| Inceptionv1 | 0.996±0.004 | 0.96±0.03 | 0.99±0.01 | 0.96±0.04 | 0.99±0.01 | 0.84±0.14 |
| Inceptionv2 | 0.996±0.005 | 0.93±0.05 | 0.99±0.01 | 0.96±0.07 | 0.98±0.01 | 0.67±0.26 |
| Inceptionv3 | 0.996±0.006 | 0.96±0.03 | 0.99±0.01 | 0.98±0.01 | 0.98±0.02 | 0.83±0.13 |
| Inceptionv4 | 0.995±0.009 | 0.97±0.02 | 0.99±0.01 | 0.98±0.01 | 0.98±0.04 | 0.88±0.11 |
| ResNetv1 | 0.993±0.009 | 0.97±0.03 | 0.99±0.02 | 0.97±0.03 | 0.98±0.03 | 0.85±0.14 |
| ResNetv2 | 0.98±0.02 | 0.96±0.04 | 0.97±0.04 | 0.96±0.06 | 0.93±0.09 | 0.83±0.17 |

En la tabla 8, se puede observar que hay una mayor variabilidad en las métricas de validación que en las de entrenamiento, lo que sugiere que los modelos pueden estar sobre ajustando al conjunto de entrenamiento.

Tabla 9

| *Modelo* | *Total de parámetros* | *Parámetros entrenables* | *Parámetros no entrenables* |
|---|---|---|---|
| Stella | 105,729 | 105,729 | 0 |
| Inceptionv1 | 3,451,121 | 3,436,561 | 14,560 |
| Inceptionv2 | 5,760,680 | 5,740,392 | 20,288 |
| Inceptionv3 | 12,569,041 | 12,540,977 | 28,064 |
| Inceptionv4 | 28,558,465 | 28,507,457 | 51,008 |
| ResNetv1 | 16,051,537 | 15,985,137 | 66,400 |
| ResNetv2 | 35,151,841 | 35,027,041 | 124,800 |

En la tabla 9 se describe la cantidad de parámetros entrenables y no entrenables de cada arquitectura.

## *Segundo Experimento*

Para el segundo experimento comparamos la arquitectura de la red neuronal de Stella con las arquitecturas DilatedConv, DilatedConv+attention, DilatedConv+multihead attention, TCN y TCN+attention.

Tabla 10

| | *Conjunto de validación* | | | | |
|---|---|---|---|---|---|
| Modelo | Accuracy | Precision (Flare – Non Flare) | Recall (Flare – Non Flare) | F1-Score (Flare – Non Flare) | ROC |
| Stella | 0.98 | 0.97-0.98 | 0.91-0.99 | 0.94-0.99 | 0.99 |
| DilatedConv | 0.99 | 0.97-0.99 | 0.97-0.99 | 0.97-0.99 | 0.99 |
| DilatedConv +attention | 0.99 | 0.98-0.99 | 0.96-1.00 | 0.97-0.99 | 0.99 |
| DilatedConv +multihead attention | 0.99 | 0.98-0.99 | 0.96-0.99 | 0.97-0.99 | 1.00 |
| TCN | 0.99 | 0.97-0.99 | 0.95-0.99 | 0.96-0.99 | 0.99 |
| TCN +attention | 0.99 | 0.97-0.99 | 0.95-0.99 | 0.96-0.99 | 0.99 |

En la Tabla 10 en el conjunto de validación podemos observar que los 5 modelos propuestos para este experimento mejoran las métricas que indican la clasificación de ambas clases. En particular mejoran en +1% el accuracy.

Tabla 11

| | *Conjunto de test* | | | | |
|---|---|---|---|---|---|
| Modelo | Accuracy | Precision (Flare – Non Flare) | Recall (Flare – Non Flare) | F1-Score (Flare – Non Flare) | ROC |
| Stella | 0.98 | 0.98-0.98 | 0.92-1.00 | 0.95-0.99 | 0.99 |
| DilatedConv | 0.99 | 0.97-0.99 | 0.96-0.99 | 0.96-0.99 | 0.99 |
| DilatedConv +attention | 0.99 | 0.98-0.99 | 0.97-1.00 | 0.97-0.99 | 0.99 |
| DilatedConv +multihead attention | 0.99 | 0.98-0.99 | 0.96-1.00 | 0.97-0.99 | 0.99 |
| TCN | 0.99 | 0.97-0.99 | 0.96-0.99 | 0.97-0.99 | 0.99 |
| TCN +attention | 0.99 | 0.97-0.99 | 0.96-0.99 | 0.96-0.99 | 0.99 |

En la Tabla 11 del conjunto de pruebas, se observa un comportamiento similar al encontrado en el conjunto de validación. Los cinco modelos evaluados muestran una mejoría en la identificación de verdaderos positivos en comparación con el modelo de referencia (Stella).

Tabla 12

| | *Conjunto de validación* | | | | *Conjunto de test* | | | |
|---|---|---|---|---|---|---|---|---|
| Modelo | TP | TN | FP | FN | TP | TN | FP | FN |
| Stella | 801 | 3550 | 28 | 80 | 780 | 3597 | 17 | 65 |
| DilatedConv | 851 | 3551 | 27 | 30 | 813 | 3587 | 27 | 32 |
| DilatedConv +attention | 845 | 3563 | 15 | 36 | 817 | 3597 | 17 | 28 |
| DilatedConv +multihead attention | 844 | 3557 | 21 | 37 | 810 | 3601 | 13 | 35 |
| TCN | 841 | 3552 | 26 | 40 | 814 | 3588 | 26 | 31 |
| TCN +attention | 840 | 3553 | 25 | 41 | 811 | 3587 | 27 | 34 |

En la Tabla 12 podemos observar que todos los modelos mejoran la clasificación de las instancias en el conjunto de validación y en el conjunto de test mejora en la mayoría de clasificaciones.

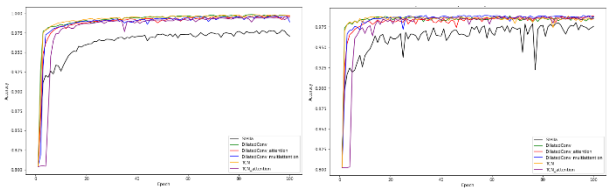


Figura 21: Comparativa del accuracy durante el entrenamiento en el conjunto de train (izquierda), Comparativa del accuracy durante el entrenamiento en el conjunto de validación (derecha).

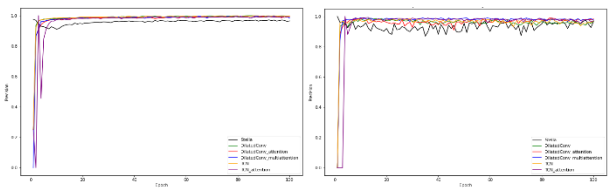


Figura 22: Comparativa de Precision durante el entrenamiento en el conjunto de train (izquierda), Comparativa de Precision durante el entrenamiento en el conjunto de validación (derecha).

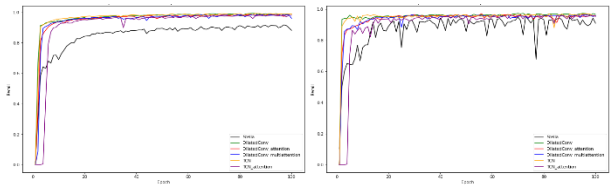


Figura 23: Comparativa de recall durante el entrenamiento en el conjunto de train (izquierda), Comparativa del recall durante el entrenamiento en el conjunto de validación (derecha).

En las Figuras 21, 22 y 23 podemos observar que los 5 modelos propuestos se ajustan muy bien a los datos, reduciendo la variabilidad de las métricas tanto al conjunto de entrenamiento como al conjunto de validación y no hay indicios de sobre ajuste. Incluso, sus gráficas son más suaves que las de Stella.

Tabla 13

| *Modelo* | *accuracy* | *Val accuracy* | *precision* | *Val precision* | *recall* | *Val recall* |
|---|---|---|---|---|---|---|
| Stella | 0.96±0.02 | 0.96±0.02 | 0.96±0.01 | 0.93±0.02 | 0.85±0.12 | 0.87±0.11 |
| DilatedConv | 0.99±0.02 | 0.98±0.02 | 0.98±0.07 | 0.95±0.10 | 0.96±0.10 | 0.94±0.10 |
| DilatedConv +attention | 0.99±0.02 | 0.98±0.02 | 0.98±0.07 | 0.96±0.10 | 0.95±0.12 | 0.94±0.10 |
| DilatedConv +multihead attention | 0.99±0.02 | 0.98±0.02 | 0.97±0.10 | 0.97±0.10 | 0.94±0.13 | 0.93±0.10 |
| TCN | 0.99±0.02 | 0.98±0.02 | 0.98±0.07 | 0.96±0.10 | 0.96±0.11 | 0.95±0.10 |
| TCN +attention | 0.98±0.04 | 0.98±0.04 | 0.96±0.13 | 0.95±0.17 | 0.92±0.20 | 0.90±0.19 |

La hipótesis de que los 5 modelos propuestos se ajustan muy bien a los datos y reducen la variabilidad de las métricas se ve respaldada por la tabla 13.

Tabla 14

| *Modelo* | *Total de parámetros* | *Parámetros entrenables* | *Parámetros no entrenables* |
|---|---|---|---|
| Stella | 105,729 | 105,729 | 0 |
| DilatedConv | 34,465 | 34,465 | 0 |
| DilatedConv +attention | 34,465 | 34,465 | 0 |
| DilatedConv +multihead attention | 35,545 | 35,545 | 0 |
| TCN | 34,465 | 34,465 | 0 |
| TCN +attention | 34,465 | 34,465 | 0 |

En la Tabla1 4 se describe la cantidad de parámetros entrenables y no entrenables de cada arquitectura.

## *Ensamble de los modelos del segundo experimento*

Para este ensamble utilizamos los modelos DilatedConv, DilatedConv+attention, DilatedConv+multihead attention, TCN, TCN+attention ya entrenados en el experimento 2.

Con el propósito de establecer la clasificación de flare o no flare, se realiza un proceso de votación suave en el cual cada

modelo individual genera una distribución de probabilidad para cada clase. Posteriormente, se calcula un promedio ponderado de las distribuciones de probabilidad emitidas por cada modelo. Para determinar la predicción final, se establece un umbral de clasificación en 0.5, en donde cualquier valor por encima del umbral se clasifica como flare y cualquier valor por debajo se clasifica como no flare.

Tabla 15

| *Conjunto* | *Accuracy* | *Precision (Flare-no flare)* | *Recall (Flare-no flare)* | *F1-score (Flare-no flare)* | *ROC* |
|---|---|---|---|---|---|
| Validación | 0.99 | 0.99-0.99 | 0.96-1.00 | 0.97-0.99 | 0.98 |
| Test | 0.99 | 0.99-0.99 | 0.97-1.00 | 0.98-1.00 | 0.98 |

En la tabla 15 observamos la mejoría que obtienen las métricas del ensamble con respecto a los modelos individuales del experimento 2.

Tabla 16

| | *Conjunto de validación* | | | | *Conjunto de test* | | | |
|---|---|---|---|---|---|---|---|---|
| Modelo | TP | TN | FP | FN | TP | TN | FP | FN |
| Stella | 801 | 3550 | 28 | 80 | 780 | 3597 | 17 | 65 |
| Ensamble | 847 | 3567 | 11 | 34 | 816 | 3608 | 6 | 29 |

En la tabla 16 observamos una mejora considerable en la clasificación de nuestro ensamble con respecto a Stella.

## VI. DISCUSIÓN

En este apartado discutiremos el significado de los resultados presentados en el Capítulo 5. Para ello lo dividiremos en 3 secciones:

- Primer experimento
- Segundo experimento
- Ensamble de los 5 modelos propuestos en el experimento 2

### *Primer experimento*

En esta sección, se llevará a cabo un análisis de los modelos Inceptionv1, Inceptionv4, ResNetv1 y ResNetv4 del experimento 1, ya que estos modelos presentan una mejora significativa en la clasificación de los verdaderos positivos en comparación con el modelo Stella.

Tabla 15

| *Modelo* | *Precision* | *% $\frac{modelo - Stella}{Stella} \cdot 100$* | *Recall* | *% $\frac{modelo - Stella}{Stella} \cdot 100$* | *F1-Score* | *% $\frac{modelo - Stella}{Stella} \cdot 100$* |
|---|---|---|---|---|---|---|
| Stella | 97 | | 91 | | 94 | |
| InceptionV1 | 98 | +1 | 94 | +3.2 | 96 | +2.1 |
| InceptionV4 | 99 | +2 | 94 | +3.2 | 96 | +2.1 |
| ResNetV1 | 98 | +1 | 96 | +5.5 | 97 | +3.2 |
| ResNetV4 | 99 | +2 | 96 | +5.5 | 97 | +3.2 |
| Promedio | | +1.5 | | +4.35 | | +2.65 |

En la Tabla 21 se muestra el porcentaje de mejora en el conjunto de validación de cada modelo en las métricas de precisión, Recall y puntuación F1. Es importante destacar que se ha observado una mejora significativa en el Recall, lo que indica que estos modelos han mejorado su capacidad para identificar verdaderos positivos, teniendo en cuenta la cantidad de falsos negativos, es decir, los eventos de flares que inicialmente habían sido clasificados incorrectamente como no flares.

Tabla 16

| *Modelo* | *Precision* | *% $\frac{modelo - Stella}{Stella} \cdot 100$* | *Recall* | *% $\frac{modelo - Stella}{Stella} \cdot 100$* | *F1-Score* | *% $\frac{modelo - Stella}{Stella} \cdot 100$* |
|---|---|---|---|---|---|---|
| Stella | 98 | | 92 | | 95 | |
| InceptionV1 | 99 | +1 | 94 | +2.2 | 96 | +1.1 |
| InceptionV4 | 99 | +1 | 95 | +3.3 | 97 | +2.1 |
| ResNetV1 | 99 | +1 | 97 | +5.4 | 98 | +3.2 |
| ResNetV4 | 99 | +1 | 95 | +3.3 | 97 | +2.1 |
| Promedio | | +1 | | +3.55 | | +2.13 |

En la Tabla 16, se puede observar un patrón similar en el desempeño de los modelos en el conjunto de test en comparación con el conjunto de validación de la Tabla 15.

Tabla 17

| *Modelo* | *Parámetros entrenables* | *Incremento porcentual de parámetros entrenables* |
|---|---|---|
| Stella | 105,729 | |
| InceptionV1 | 3,436,561 | +3,150.3 |
| InceptionV4 | 28,507,457 | +27,862.8 |
| ResNetV1 | 15,985,137 | +15,019 |
| ResNetV4 | 35,027,041 | +33,029.1 |
| Promedio | | +16385.8 |

Debido a la naturaleza compleja de estos modelos, existe un alto riesgo de sufrir sobre ajuste durante el proceso de entrenamiento, lo cual se evidencia en las Figuras 18, 19 y 20, así como en el resumen presentado en la Tabla 8. Los resultados obtenidos por estos modelos no son lo suficientemente significativos como para cumplir con los objetivos establecidos en el presente trabajo. Es decir, los modelos tienden a sobre ajustarse al conjunto de entrenamiento y aumentan drásticamente la cantidad de parámetros.

### *Segundo Experimento*

En esta sección se analizará detalladamente el rendimiento de los modelos DilatedConv, DilatedConv+attention, DilatedConv+multihead attention, TCN, TCN+attention.

Tabla 18

| *Modelo* | *Precision (flare-no flare)* | *% $\frac{modelo - Stella}{Stella} \cdot 100$* | *Recall (flare-no flare)* | *% $\frac{modelo - Stella}{Stella} \cdot 100$* | *F1-Score (flare-no flare)* | *% $\frac{modelo - Stella}{Stella} \cdot 100$* |
|---|---|---|---|---|---|---|
| Stella | 97-98 | | 91-99 | | 94-99 | |
| DilatedConv | 97-99 | 0, +1 | 97-99 | +6.6, 0 | 97-99 | +3.2, 0 |
| DilatedConv +attention | 98-99 | +1, +1 | 96-100 | +5.5, +1 | 97-99 | +3.2, 0 |
| | 98-99 | +1, +1 | 96-99 | +5.5, 0 | 97-99 | +3.2, 0 |
| DilatedConv +multihead attention | 97-99 | 0, +1 | 95-99 | +4.4, 0 | 96-99 | +2.1, 0 |
| | 97-99 | 0, +1 | 95-99 | +4.4, 0 | 96-99 | +2.1, 0 |
| Promedio | | +0.4, +1 | | +5.28, +0.2 | | +2.76, 0 |

En la Tabla 18 se puede observar que los cinco modelos evaluados en el segundo experimento en el conjunto de entrenamiento mejoraron tanto la tasa de verdaderos positivos

como la tasa de verdaderos negativos, lo cual se refleja en las métricas Recall y F1-Score. En promedio, estos modelos lograron una mejora del 5.28% en la tasa de verdaderos positivos, considerando los eventos de flares que inicialmente habían sido clasificados incorrectamente como no flares, y una mejora del 2.76% en el F1-Score.

Tabla 19

| *Modelo* | *Precision* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *Recall* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *F1-Score* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ |
|---|---|---|---|---|---|---|
| Stella | 98-98 | | 92-100 | | 95-99 | |
| DilatedConv | 97-99 | -1, +1 | 96-99 | +4.3, -1 | 96-99 | +1.1, 0 |
| DilatedConv +attention | 98-99 | 0, +1 | 97-100 | +5.4, 0 | 97-99 | +2.1, 0 |
| DilatedConv +multihead attention | 98-99 | 0, +1 | 96-100 | +4.3, 0 | 97-99 | +2.1, 0 |
| | 97-99 | -1, +1 | 96-99 | +4.3, -1 | 97-99 | +2.1, 0 |
| | 97-99 | -1, +1 | 96-99 | +4.3, -1 | 96-99 | +1.1, 0 |
| Promedio | | -0.6, +1 | | +4.52, -0.6 | | +1.7,0 |

En la Tabla 19 se puede apreciar un equilibrio más favorable entre las métricas de precisión y recall. Esto se refleja en un aumento del puntaje F1, lo cual indica una mejora en la capacidad de los modelos para identificar tanto verdaderos positivos como verdaderos negativos.

Tabla 20

| *Modelo* | *TP* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *TN* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *FP* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *FN* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ |
|---|---|---|---|---|---|---|---|---|
| Stella | 801 | | 3550 | | 28 | | 80 | |
| DilatedConv | 851 | +6.2 | 3551 | +0.02 | 27 | -3.6 | 30 | -62.5 |
| DilatedConv +attention | 845 | +5.5 | 3563 | +0.37 | 15 | -46.4 | 36 | -55 |
| DilatedConv +multihead attention | 844 | +5.3 | 3557 | +0.2 | 21 | -25 | 37 | -53.8 |
| TCN | 841 | +5 | 3552 | +0.06 | 26 | -7.1 | 40 | -50 |
| TCN +attention | 840 | +5 | 3553 | +0.08 | 25 | -10.7 | 41 | -49 |
| Promedio | | +5.4 | | +0.15 | | -18.6 | | -54.1 |

En la Tabla 20 se puede observar que los cinco modelos evaluados en el conjunto de validación lograron mejorar la clasificación de eventos, aumentando la cantidad de eventos que son clasificados correctamente como verdaderos positivos y verdaderos negativos, mientras que disminuyen los eventos clasificados incorrectamente como falsos positivos y falsos negativos.

Tabla 21

| *Modelo* | *TP* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *TN* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *FP* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *FN* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ |
|---|---|---|---|---|---|---|---|---|
| Stella | 780 | | 3597 | | 17 | | 65 | |
| DilatedConv | 813 | +4.2 | 3587 | -0.28 | 27 | +58.8 | 32 | -50.8 |
| DilatedConv + attention | 817 | +4.7 | 3597 | 0 | 17 | 0 | 28 | -56.9 |
| DilatedConv + multihead attention | 810 | +4.4 | 3601 | +0.11 | 13 | | 35 | -46.2 |
| TCN | 814 | +4.4 | 3588 | -0.25 | 26 | +52.9 | 31 | -52.3 |
| TCN + attention | 811 | +4.4 | 3587 | -0.28 | 27 | +58.8 | 34 | -47.7 |
| Promedio | | +4.42 | | -0.14 | | +34.1 | | -50.8 |

En la Tabla 21 se puede observar que, en el conjunto de prueba, los modelos DilatedConv, TCN y TCN+Attention lograron mejorar la clasificación de instancias como verdaderos positivos y falsos negativos, lo cual significa que fueron capaces de identificar más eventos de flares correctamente. Sin embargo, estos modelos también sacrificaron la clasificación de verdaderos negativos y falsos positivos, lo que significa que se clasificaron erróneamente algunos eventos que no eran flares como flares.

Tabla 22

| *Modelo* | *accuracy* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *precision* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *recall* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ |
|---|---|---|---|---|---|---|
| Stella | 96 | | 96 | | 85 | |
| DilatedConv | 99 | +3.1 | 98 | +2.1 | 96 | +12.9 |
| DilatedConv +attention | 99 | +3.1 | 98 | +2.1 | 95 | +11.8 |
| DilatedConv +multihead attention | 99 | +3.1 | 97 | +1 | 94 | +10.6 |
| TCN | 99 | +3.1 | 98 | +2.1 | 96 | +11.8 |
| TCN +attention | 98 | +2.1 | 96 | 0 | 92 | +8.2 |
| Promedio | | +2.9 | | +1.46 | | +11.06 |

Comparativa de las métricas en el conjunto de train durante el entrenamiento.

Tabla 23

| *Modelo* | *accuracy* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *precision* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ | *recall* | *%* $\frac{modelo - Stella}{Stella} \cdot 100$ |
|---|---|---|---|---|---|---|
| Stella | 96 | | 93 | | 87 | |
| DilatedConv | 98 | +2.1 | 95 | +2.2 | 94 | +8 |
| DilatedConv +attention | 98 | +2.1 | 96 | +3.2 | 94 | +8 |
| DilatedConv +multihead attention | 98 | +2.1 | 97 | +4.3 | 93 | +6.9 |
| TCN | 98 | +2.1 | 96 | +3.2 | 95 | +9.2 |
| TCN +attention | 98 | +2.1 | 95 | +2.2 | 90 | +3.4 |
| Promedio | | +2.1 | | +3.02 | | +7.1 |

Comparativa de las métricas en el conjunto de validación durante el entrenamiento.

Se puede observar en la Tabla 22 y 23 que a medida que avanza el entrenamiento, los 5 modelos del segundo experimento presentan mejoras en el accuracy, Precision y Recall. Esto indica que los modelos están aprendiendo y ajustándose cada vez mejor a los datos de entrenamiento, mejorando su capacidad para clasificar correctamente las instancias en verdaderos positivos y verdaderos negativos, y disminuyendo la cantidad de falsos positivos y falsos negativos.

Como último punto en este apartado analizamos la Tabla 14. En la Tabla 14 observamos que las 5 arquitecturas en este experimento se reducen en un gran porcentaje.

Tabla 24

| *Modelo* | *Parámetros entrenables* | *Incremento porcentual de parámetros entrenables* |
|---|---|---|
| Stella | 105,729 | |
| DilatedConv | 34,465 | -67.4 |
| DilatedConv +attention | 34,465 | -67.4 |
| DilatedConv +multihead attention | 35,545 | -66.4 |
| TCN | 34,465 | -67.4 |
| TCN +attention | 34,465 | -67.4 |
| Promedio | | -67.2 |

Debido a los resultados expuestos en esta sección, se puede concluir que cada uno de los modelos propuestos es una buena alternativa con respecto a Stella, ya que todos muestran una mejora en el accuracy, precision y recall. Aumentando la tasa de clasificación de verdaderos positivos y sobre todo disminuyendo la cantidad de parámetros entrenables.

### *Ensamble de los 5 modelos del segundo experimento*

Tabla 25

| *Modelo* | *Accuracy* | *%* | *Precision (Flare-no flare)* | *%* | *Recall (Flare-no flare)* | *%* | *F1-score (Flare-no flare)* | *%* |
|---|---|---|---|---|---|---|---|---|
| Stella | 98 | | 97-98 | | 91-99 | | 94-99 | |
| Ensamble | 99 | +1 | 99-99 | +2.1, +1 | 96-100 | +5.5, +1 | 97-99 | +3.2,0 |

Comparativa de métricas en el conjunto de validación del ensamble de los 5 modelos del experimento 2 con respecto a Stella.

Tabla 26

| *Modelo* | *Accuracy* | *%* | *Precision (Flare-no flare)* | *%* | *Recall (Flare-no flare)* | *%* | *F1-score (Flare-no flare)* | *%* |
|---|---|---|---|---|---|---|---|---|
| Stella | 98 | | 98-98 | | 92-100 | | 95-99 | |
| Ensamble | 99 | +1 | 99-99 | +1, +1 | 97-100 | +5.4, 0 | 98-100 | +3.2, +1 |

Comparativa de métricas en el conjunto de test del ensamble de los 5 modelos del experimento 2 con respecto a Stella.

Tabla 27

| *Modelo* | *TP* | *%* | *TN* | *%* | *FP* | *%* | *FN* | *%* |
|---|---|---|---|---|---|---|---|---|
| Stella | 801 | | 3550 | | 28 | | 80 | |
| Ensamble | 847 | +5.7 | 3567 | +0.5 | 11 | -60.7 | 34 | -57.5 |

Comparación matriz de confusión en el conjunto de validación.

Tabla 28

| *Modelo* | *TP* | *%* | *TN* | *%* | *FP* | *%* | *FN* | *%* |
|---|---|---|---|---|---|---|---|---|
| Stella | 780 | | 3597 | | 17 | | 65 | |
| Ensamble | 816 | +4.6 | 3608 | +0.3 | 6 | -64.7 | 29 | -55.4 |

Comparación matriz de confusión en el conjunto de test.

Se puede apreciar en la Tabla 27 y 28 que la utilización de un ensamble de los cinco modelos del segundo experimento resulta en una disminución significativa de la tasa de error de clasificación, así como en un aumento de la tasa de clasificación correcta de verdaderos positivos y verdaderos negativos en comparación con el modelo Stella.

Lo destacable de este ensamble es que, como se puede apreciar en las gráficas suaves presentadas en las Figuras 21, 22 y 23 para el conjunto de entrenamiento y validación durante el proceso de entrenamiento, es posible obtener resultados similares con una menor cantidad de épocas de entrenamiento.

## VII. CONCLUSIONES

A continuación, se reúnen las conclusiones alcanzadas en el presente trabajo.

En el apartado Objetivos y Metodología establecimos los objetivos de este trabajo y sus criterios de éxito. El primero consistía en identificar y explorar las principales herramientas utilizadas para clasificar fulguraciones estelares en curvas de luz y el tipo de técnicas que emplean.

Identificamos dos tipos de técnicas principales:

- Basadas en técnicas en transformaciones para detectar valores atípicos.
- Basadas en técnicas de Inteligencia artificial.

Particularmente centramos nuestro punto de atención en las herramientas basadas en técnicas de inteligencia artificial y nos permitió examinar la estructura de:

- FLATW'RM que se basa en el algoritmo RANSAC para la detección de valores atípicos.
- FLATW'RM2 que se basa en redes neuronales recurrentes de tipo LSTM.
- Stella que se basa en redes neuronales convolucionales.

El segundo objetivo consistía en analizar y profundizar en el entendimiento del algoritmo Stella. Explorando todos sus módulos y funcionalidades y por supuesto experimentar en su módulo de Neural Network que nos permitió entrenar nuevos modelos con diferentes arquitecturas de redes neuronales. Este trabajo se basa en los resultados obtenidos en este módulo.

Como tercer objetivo nos planteamos modificar el algoritmo Stella con otras arquitecturas de redes neuronales. Para cumplir con este objetivo, entrenamos 6 modelos distintos de arquitecturas conocidas adaptadas al procesamiento de señales unidimensionales. enumerados a continuación:

- GoogLeNet o InceptionV1
- InceptionV2
- InceptionV3
- InceptionV4
- ResNetV1
- ResNetV2

Y además propusimos 5 nuevas arquitecturas que llamamos:

- DilatedConv
- DilatedConv + attention
- DilatedConv + multihead attention
- TCN
- TCN + attention

En donde exploramos una diversidad de mecanismos que nos pudiese ayudar a abordar este problema.

Nuestro último objetivo fue evaluar estos algoritmos con diversas métricas como el accuracy, precision, recall, F1-score, ROC-AUC y por supuesto su coste computacional basándonos en la cantidad de parámetros. También utilizamos como métrica el análisis visual y las métricas del entrenamiento.

Con el análisis visual de las métricas del entrenamiento, descubrimos que los modelos entrenados con las arquitecturas Inception y ResNet estaban sobre ajustados. Esto puede deberse a distintos factores: complejidad de la arquitectura, no había suficientes datos de entrenamiento, desbalance entre las clases, ajuste de hiper parámetros, etc. Particularmente no exploramos la búsqueda de hiper parámetros adecuado e intentar reducir el sobreajuste porque cada entrenamiento tomaba mucho tiempo y no contábamos con el equipo adecuado para agilizar el proceso. Por lo que nos era inviable seguir con esa ruta.

Por lo que exploramos nuevas vías, tomando como objetivo reducir la complejidad de la arquitectura. Mediante la

investigación sistemática que nos permitiera resolver el problema de clasificación de fulguraciones estelares en curvas de luz encontramos 3 técnicas que nos podían ayudar.

- Capas de convoluciones dilatadas
- Capas de convoluciones causales
- Capas de atención

Esta investigación nos permitió encontrar casos de éxito como por ejemplo WaveNet, que combina las capas convoluciones dilatadas con las capas convolucionales causales en capas convolucionales causales dilatadas y por supuesto, los transformers con sus mecanismos de atención.

Las 5 arquitecturas propuestas en el experimento 2 que combinan estas tres técnicas mejoraron los resultados de Stella que fue nuestro punto de referencia. Mediante las capas convolucionales nos ayudaba a ampliar nuestro campo receptivo sin añadir complejidad al modelo, las capas convolucionales causales nos ayudaron a aprovechar el componente temporal presente en las curvas de luz y las capas de atención nos ayudaron a que el modelo aprendiera qué característica nos ayudaba a clasificar mejor.

Precisamente estas 5 arquitecturas nos ayudaron a conseguir cumplir con el objetivo general de este trabajo de investigación "Optimizar el algoritmo Stella para clasificar fulguraciones estelares en curvas de luz proporcionadas por la misión TESS utilizando nuevas arquitecturas de redes neuronales obteniendo una mejora en la clasificación de verdaderos positivos con un menor coste computacional".

Estas son las métricas que soportan esta afirmación:

- Los modelos finales en el conjunto de validación: precision: +0.4% en promedio, recall: +5.28 en promedio, f1-score: +2.76 en promedio (Tabla 18).
- Los modelos finales en el conjunto de test: precision: -0.6% en promedio, recall: +4.52% en promedio, f1-score: +1.7% en promedio (Tabla 19).
- Métricas del conjunto de train durante el entrenamiento: accuracy: +2.9% en promedio, precision: +1.46% en promedio, recall: +11.06% en promedio (Tabla 22).
- Métricas del conjunto de validación durante el entrenamiento: accuracy: +2.1% en promedio, precision: +3.02% en promedio, recall: +7.1% en promedio (Tabla 23).
- En promedio logramos un 5.4% de mejora en la clasificación de verdaderos positivos según la matriz de confusión en el conjunto de validación (Tabla 20) y 4.42% en el conjunto de test (Tabla 21) con los modelos obtenidos.
- En promedio una reducción de parámetros de 67.2% (Tabla 24).

Por último, el ensamble de estos 5 modelos del segundo experimento obtuvo mejores resultados que todos los modelos individuales (incluido Stella), reduciendo en un 64.7% la clasificación de los falsos positivos y un 55.4 de reducción en la clasificación de los falsos negativos.

La clasificación de fulguraciones estelares es de gran relevancia en el campo de astrofísica, por lo que este trabajo puede servir de base a varias líneas de trabajo futuras.

- Análisis detallado de las instancias clasificadas incorrectamente por estos modelos: Nos permitirá identificar en los casos particulares en donde nuestros modelos fallan. De esa forma poder tomar acción en la búsqueda e implementación de métodos que eviten que nuestros modelos tengan esos fallos.
- Utilizar nuestros modelos para clasificar fulguraciones estelares en curvas de luz de los demás sectores que proporciona la misión TESS. Aplicar filtros para disminuir falsos positivos de la forma en que Feinstein et al. (2022) plantea en su artículo.
- Utilizar las arquitecturas propuestas en el experimento 2 para la clasificar fulguraciones observadas en otras longitudes de onda (UV) y otras cadencias temporales.

Las arquitecturas CNN son bastante flexibles, por lo que puede utilizarse para la clasificación de señales unidimensionales en otros ámbitos. Por ejemplo:

- Electrocardiogramas y encefalogramas en medicina
- Señales bursátiles en economía.
- Clasificación de sonidos.
- Etc.

## REFERENCIAS

[1] Bahdanau, D., Cho, K., & Bengio, Y. (2016). Neural Machine Translation by Jointly Learning to Align and Translate (arXiv:1409.0473). https://doi.org/10.48550/arXiv.1409.0473

[2] Borucki, W. J., Koch, D. G., Lissauer, J. J., Basri, G. B., Caldwell, J. F., Cochran, W. D., Dunham, E. W., Geary, J. C., Latham, D. W., Gilliland, R. L., Caldwell, D. A., Jenkins, J. M., & Kondo, Y. (2003). The Kepler mission: A wide-field-of-view photometer designed to determine the frequency of Earth-size planets around solar-like stars. 4854, 129-140. https://doi.org/10.1117/12.460266

[3] Davenport, J. R. A. (2016). The Kepler Catalog of Stellar Flares. The Astrophysical Journal, 829(1), 23. https://doi.org/10.3847/0004-637X/829/1/23

[4] Davenport, J. R. A., Hawley, S. L., Hebb, L., Wisniewski, J. P., Kowalski, A. F., Johnson, E. C., Malatesta, M., Peraza, J., Keil, M., Silverberg, S. M., Jansen, T. C., Scheffler, M. S., Berdis, J. R., Larsen, D. M., & Hilton, E. J. (2014). Kepler Flares II: The Temporal Morphology of White-Light Flares on GJ 1243. The Astrophysical Journal, 797(2), 122. https://doi.org/10.1088/0004-637X/797/2/122

[5] Feinstein, A. D., Montet, B. T., & Ansdell, M. (2020). \texttt{stella}: Convolutional Neural Networks for Flare Identification in \textit{TESS}. Journal of Open Source Software, 5(52), 2347. https://doi.org/10.21105/joss.02347

[6] Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6), 381-395. https://doi.org/10.1145/358669.358692

[7] Google Collaboratory. (s. f.). Recuperado 31 de enero de 2023, de https://colab.research.google.com/

[8] Günther, M. N., Zhan, Z., Seager, S., Rimmer, P. B., Ranjan, S., Stassun, K. G., Oelkers, R. J., Daylan, T., Newton, E., Kristiansen, M. H., Olah, K., Gillen, E., Rappaport, S., Ricker, G. R., Vanderspek, R. K., Latham, D. W., Winn, J. N., Jenkins, J. M., Glidden, A., … Ting, E. B. (2020). Stellar Flares from the First TESS Data Release:

Exploring a New Sample of M Dwarfs. The Astronomical Journal, 159(2), 60. https://doi.org/10.3847/1538-3881/ab5d3a

[9] Hao, X., Su, X., Wang, Z., Zhang, H., & Batushiren, A. (2019). UNetGAN: A Robust Speech Enhancement Approach in Time Domain for Extremely Low Signal-to-Noise Ratio Condition. 1786-1790. https://doi.org/10.21437/Interspeech.2019-1567

[10] Ilin, E., Schmidt, S. J., Poppenhäger, K., Davenport, J. R. A., Kristiansen, M. H., & Omohundro, M. (2022). Ekaterinailin/AltaiPony [Jupyter Notebook]. https://github.com/ekaterinailin/AltaiPony/blob/319cf3b5a685ae3fbdc3befb51031c7fc76cd637/notebooks/01_Getting_Started.ipynb

[11] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., … Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. Nature, 596(7873), Art. 7873. https://doi.org/10.1038/s41586-021-03819-2

[12] Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation (arXiv:1508.04025). https://doi.org/10.48550/arXiv.1508.04025

[13] Oord, A. van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio (arXiv:1609.03499; Versión 2). arXiv. http://arxiv.org/abs/1609.03499

[14] Ricker, G. R., Winn, J. N., Vanderspek, R., Latham, D. W., Bakos, G. Á., Bean, J. L., Berta-Thompson, Z. K., Brown, T. M., Buchhave, L., Butler, N. R., Butler, R. P., Chaplin, W. J., Charbonneau, D., Christensen-Dalsgaard, J., Clampin, M., Deming, D., Doty, J., De Lee, N., Dressing, C., … Villasenor, J. (2015). Transiting Exoplanet Survey Satellite (TESS). Journal of Astronomical Telescopes, Instruments, and Systems, 1, 014003. https://doi.org/10.1117/1.JATIS.1.1.014003

[15] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning (arXiv:1602.07261). https://doi.org/10.48550/arXiv.1602.07261

[16] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going Deeper with Convolutions (arXiv:1409.4842). arXiv. https://doi.org/10.48550/arXiv.1409.4842

[17] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision (arXiv:1512.00567). arXiv. https://doi.org/10.48550/arXiv.1512.00567

[18] Teimourian, H., & Dimililer, K. (2019). La inteligencia artificial en la detección de radiogalaxias. Dilemas contemporáneos: Educación, Política y Valores. https://doi.org/10.46377/dilemas.v30i1.1260

[19] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need (arXiv:1706.03762). https://doi.org/10.48550/arXiv.1706.03762

[20] Vida, K., Bódi, A., Szklenár, T., & Seli, B. (2021). Finding flares in Kepler and TESS data with recurrent deep neural networks. Astronomy & Astrophysics, 652, A107. https://doi.org/10.1051/0004-6361/202141068

[21] Vida, K., & Roettenbacher, R. M. (2018). Finding flares in Kepler data using machine-learning tools. Astronomy and Astrophysics, 616, A163. https://doi.org/10.1051/0004-6361/201833194

[22] Yu, F., & Koltun, V. (2016). Multi-Scale Context Aggregation by Dilated Convolutions (arXiv:1511.07122). https://doi.org/10.48550/arXiv.1511.07122