
Facultad de Ciencias

# Clasificación de imágenes astronómicas de rayos X mediante machine learning

*Proyecto Fin de Grado*

Autor:

Juan Antonio Ruiz Saro

Febrero 2021

**Resumen**

El aumento constante de datos de fuentes astronómicas obtenidos por numerosos observatorios crea la necesidad de la implementación de métodos de automatización para los procesos que actualmente requieren de análisis manual por parte de personas especializadas, como el control de calidad visual de las observaciones en la construcción de catálogos astronómicos de fuentes detectadas.

En este trabajo se presenta una posible solución a un ejemplo con esta problemática: el diseño de un algoritmo de *machine learning* y particularmente, de una red neuronal convolucional, que aprenda a clasificar imágenes de fuentes astronómicas de rayos X según la calidad de las mismas. La red entrenará con un conjunto de imágenes, ya clasificado y etiquetado por un equipo de expertos, que será tratado y analizado para que el algoritmo sea capaz de extraer sus características con éxito. Se probaran varias arquitecturas así como técnicas de regularización para dar con el modelo que mejor se ajuste a la solución del problema.

**Palabras clave:** screening, machine learning, red neuronal convolucional, técnicas de regularización

# Índice

**1. Introducción** **3**

1.1. Objetivos . . . . . 3

**2. Metodología** **3**

2.1. XMM-Newton . . . . . 4

2.1.1. Detectores . . . . . 4

2.1.2. Observando con el XMM-Newton . . . . . 5

2.1.3. Catálogo de fuentes de rayos-X . . . . . 7

2.2. Estadística paramétrica . . . . . 8

2.3. Inteligencia Artificial . . . . . 9

2.4. Aprendizaje automático . . . . . 10

2.5. Redes neuronales . . . . . 11

2.6. Aprendizaje profundo y redes neuronales convolucionales . . . . . 12

2.6.1. Evaluación de un modelo de aprendizaje automático . . . . . 14

2.6.2. Sobreajuste . . . . . 15

**3. Análisis del problema** **16**

3.1. Datos del problema . . . . . 16

3.2. Pruebas paramétricas . . . . . 17

3.3. Tratamiento de imágenes . . . . . 17
3.3.1. Rebinning . . . . . 18
3.3.2. Normalización . . . . . 18
3.3.3. Definir sets de entrada . . . . . 18
3.4. Modelos de redes neuronales convolucionales . . . . . 19
3.4.1. Tamaño de filtro . . . . . 20
3.5. Formas de hacer frente al Overfitting . . . . . 20

**4. Resultados** **21**
4.1. Resultados de la estadística paramétrica . . . . . 21
4.2. Resultados de pruebas con redes neuronales convolucionales . . . . . 21
4.3. Pruebas con tamaños de filtros . . . . . 21
4.4. Pruebas con normalizaciones . . . . . 22
4.5. Pruebas de regularización . . . . . 24

**5. Conclusiones** **28**

**A. Configurando modelo** **33**

# 1. Introducción

Hoy en día, la cantidad de observatorios astronómicos, tanto en la superficie de la Tierra como en el espacio, aumenta constantemente y con ello la cantidad de datos que estos recolectan. Tradicionalmente, el análisis de estos datos se hacía de forma semi-manual por astrónomos pero con los avances en el campo de la informática, algunos de estos métodos de análisis se están viendo sustituidos por técnicas de automatización.

Habitualmente estas observaciones se incorporan a catálogos de fuentes después del análisis de las mismas. Con el fin de garantizar la calidad de estos catálogos, parte del cribado es realizado visualmente por expertos con gran conocimiento sobre cada observatorio, que se encargan de que el catálogo no contenga fuentes espúreas o cuyas características no estén bien definidas.

Debido al desbordamiento en la cantidad de datos a escudriñar por las personas especializadas, surge la necesidad de investigar algoritmos automáticos que ayuden a aliviar la tarea que hasta ahora se hace de forma manual.

En los últimos años han proliferado dentro del campo de la inteligencia artificial unos algoritmos disruptivos de aprendizaje automático. Esta tecnología ya se usa en campos en los que se requiere del manejo de grandes conjuntos de datos como en la conducción autónoma, en la clasificación de secuencias de ADN o en estudios sociológicos.

La característica de estos algoritmos que los ha vuelto tan útiles en tantos ámbitos diferentes es la de aprender a realizar una tarea en base a la experiencia previa.

Todo esto hace que se plantee la posibilidad de incorporar esta tecnología al proceso semi-manual de control de calidad en la clasificación de fuentes astronómicas.

## 1.1. Objetivos

Uno de los observatorios astronómicos que genera regularmente catálogos de fuentes de rayos X detectadas es el *XMM-Newton*. Estos catálogos se ven sometidos a un control de calidad manual llevado a cabo por expertos conocido como *screening*, del que se extrae una etiqueta que se asigna a cada imagen indicando las porción de imagen con regiones de mala calidad (ver sección 2.1.3.)

Entre los métodos de aprendizaje automático que han dado resultado en el pasado para la clasificación de imágenes astronómicas, se encuentran las redes neuronales convolucionales, que ya han sido utilizados en la clasificación de estrellas, galaxias y otros objetos celestes [1].

Por lo tanto, nuestro objetivo es entrenar una red neuronal convolucional para que aprenda a clasificar imágenes que contienes fuentes de rayos X para que en los futuros controles de calidad de los catálogos, disminuya la carga de trabajo debida a la etapa de *screening*.

- El modelo entrenado deberá ser capaz de predecir si una imagen es de buena o mala calidad. Basándonos en el etiquetado realizado por los expertos.

# 2. Metodología

En esta sección se describen las propiedades del observatorio que obtiene las imágenes con fuentes de rayos X, así como los conceptos necesarios para comprender los algoritmos de clasificación de imágenes empleados.

## 2.1. XMM-Newton

El observatorio XMM-Newton es una de las cuatro misiones definidas en el programa Horizon 2000. Fue lanzado a órbita el 10 de Diciembre de 1999, portando dos tipos de telescopios: tres telescopios de rayos-X, con diferentes detectores de rayos-X, y un telescopio óptico/UV [2]. El uso observatorios espaciales de rayos X es esencial para detectar fuentes astronómicas de rayos X, debido a que los rayos X son totalmente absorbidos por la átmosfera, penetrando hasta aproximadamente 80km sobre la superficie terrestre.

Desde su puesta en órbita, este observatorio científico ha sido uno de los más productivos de la ESA, ofreciendo un archivo de todas las observaciones realizadas con los datos escudriñados para ser analizados. [3]

### 2.1.1. Detectores

Dos de los telescopios de rayos X del XMM-Newton están equipados con cámaras CCD compuestas por chips MOS (Semiconductores de óxido de metal) y el tercer telescopio porta una camara CCD diferente conocida como EPIC pn [4].

Las matrices de chips MOS constan de 7 chips individuales idénticos con iluminación frontal. Estos CCD individuales no son coplanarios, sino que están desplazados entre sí, siguiendo la ligera curvatura de la superficie focal de los telescopios. 1

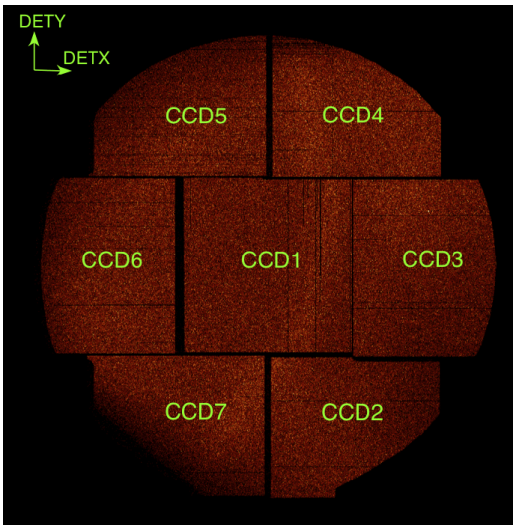



Figura 1: Disposición de los detectores para las cámaras MOS. Imagen tomada de [5]

Por otro lado, la camara pn esta formada por una única oblea de silicio con 12 chips CCD integrados. Ver figura.2

Las imágenes finales obtenidas son la suma de los tres detectores y por lo tanto, en ellas se pueden apreciar las formas de los detectores.

Los chips de las cámaras CCD consisten en una gran cuadrícula de píxeles individuales sensibles a los fotones. Las lecturas de los fotones al interactuar con los detectores se producen de forma secuencial por filas y columnas, dando lugar a un tiempo lectura muerta. Esta limitación genera problemas, por ejemplo, a la hora de observar fuentes muy brillantes que saturan los detectores al recibir gran cantidad de fotones. Para reducir este efecto temporal se acude a modos de adquisición de datos en los que solo se leen algunas partes de los detectores [7]:

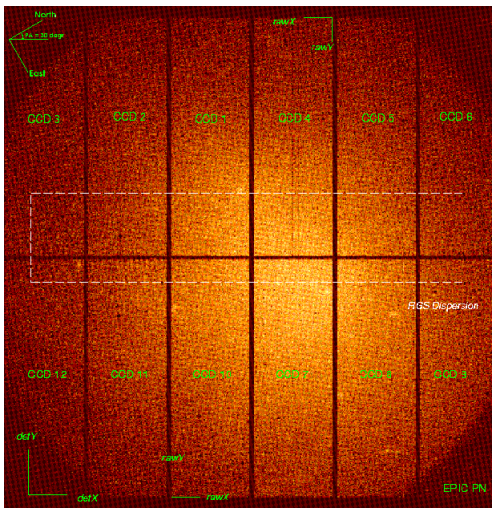



Figura 2: Disposición de los chips CCD de la cámara pn. Imagen tomada de [6]

- Ventana completa y ventana completa extendida (solo pn): En este modo, todos los píxeles de todos los CCDs están activos y por lo tanto se cubre todo el campo de visión.

- Ventana parcial (Ventana grande o pequeña)

  a) MOS

  En un modo de ventana parcial, el CCD central de ambas cámaras MOS se puede operar en un modo diferente de adquisición de datos científicos, leyendo solo una parte del chip CCD.

  b) pn

  En el modo de ventana grande, solo se lee la mitad del área en los 12 CCD, mientras que en el modo de ventana pequeña solo se usa una parte del CCD número 4 para recopilar datos. A pesar de que en este modo hay regiones de los CCD sin lecturas, siguen estando expuestas, de modo que las fuentes brillantes en estas áreas *oscuras* podrían afectar a la observación.

- Temporal a) MOS + pn

  En el modo temporal, la información espacial solo se mantiene en una dimensión. Para la cámara pn, solo colecta datos el CCD 4, mientras que

  b) pn

  La cámara pn tiene un modo temporal especial, conocido como modo ráfaga o *burst*, que ofrece alta resolución temporal pero ciclos operativos cortos.

### 2.1.2. Observando con el XMM-Newton

El XMM-Newton fue lanzado el 10 de diciembre de 1999 por un lanzador Ariane 5 en una órbita muy elíptica, con un apogeo de unos 115.000 km y un perigeo de aproximado de 6000 km sobre la superficie terrestre. XMM-Newton opera con tres estaciones terrestres, ubicadas en Yatharagga, Kourou y Santiago de Chile. Ocasionalmente también se opera desde Madrid. [8]

Debido a varias perturbaciones, la órbita del XMM-Newton cambia con el tiempo.

Limitaciones en las observaciones:

-Cinturones de radiación

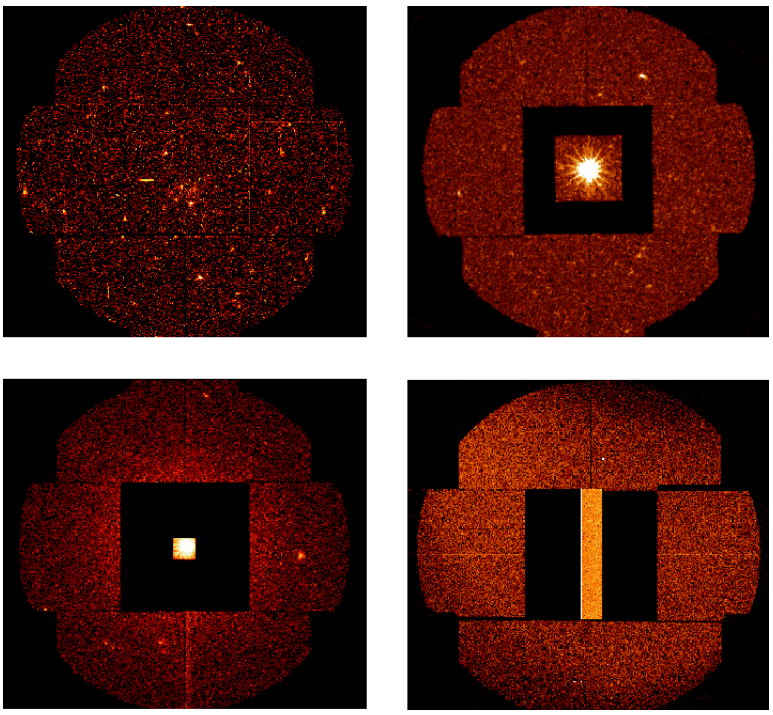


Figura 3: Modos operacionales de las cámaras MOS-CCD. Arriba a la izquierda: Modo de ventana completa. Arriba a la derecha: Modo de ventana parcial grande. Abajo a la izquierda: Modo de ventana parcial pequeña. Abajo a la derecha: Modo temporal. Fuente: [7]

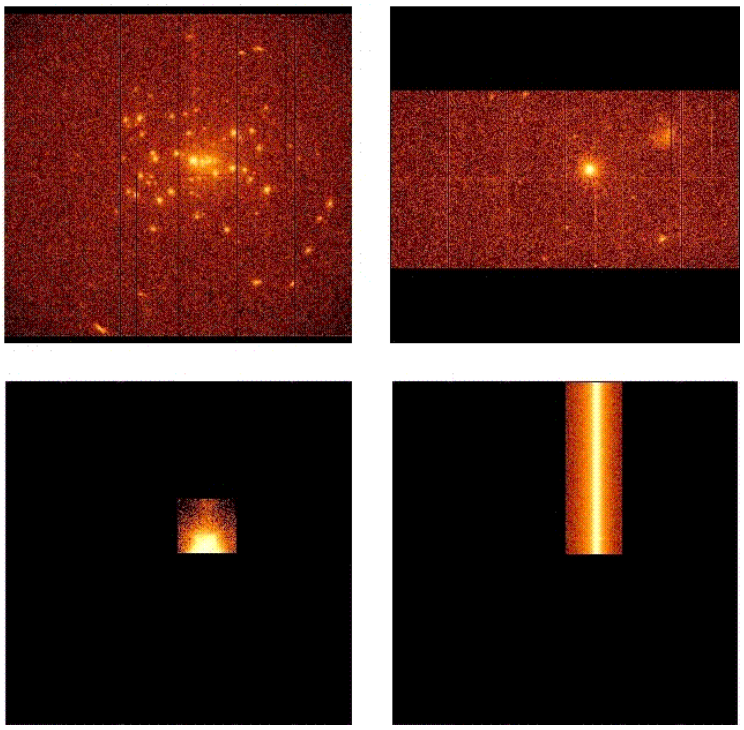


Figura 4: Modos operacionales de las cámaras pn-CCD. Arriba a la izquierda: Modo de ventana completa y ventana completa extendida. Arriba a la derecha: Modo de ventana parcial grande. Abajo a la izquierda: Modo de ventana parcial pequeña. Abajo a la derecha: Modo Burst. .Fuente: [7]

El fondo de radiación varia alrededor de la órbita del XMM-Newton en función de la ubicación del observatorio respecto a la magnetosfera terrestre. Para realizar observaciones útiles, la elevación mínima es de 46000km. A esta altura, los instrumentos pueden comenzar a realizar observaciones, pero solo si la radiación de fondo es lo suficientemente baja para no dañar los detectores. Esta limitación

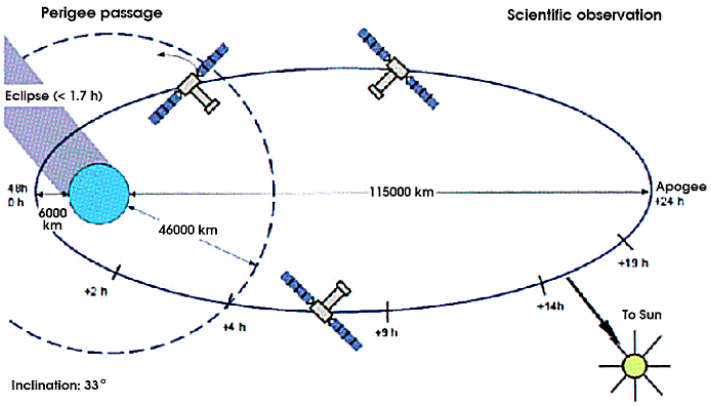



Figura 5: Esquema de la órbita elíptica que sigue el XMM-Newton

operativa implica que los instrumentos de rayos X solo tienen una pequeña ventana temporal de observación segura: de un minuto hasta aproximadamente 3 horas después de que se alcanza la elevación adecuada. Así mismo, los instrumentos de rayos-X tienen que ser cerrados entre un minuto y 3 horas antes de que la elevación del observatorio disminuya de los 46000km al final de cada órbita. [9]

Otro factor que limita la visibilidad de las fuentes de rayos-X, es la presencia de las fuentes del sistema solar. Para evitar el ruido proveniente de estas fuentes, el observatorio tiene que mantener un cierto ángulo de evitación respecto a dichas fuentes: [10]

El ángulo de evitación solar debe mantenerse dentro del rango $70^{o}$-$110^{o}$ en todo momento. Este rango es indispensable para suplir a la nave con energía suficiente además de mantener la estabilidad térmica.

Para las extremidades terrestres y la Luna, los ángulos de evitación mínimos son de $42.5^{o}$ y $22^{o}$ respectivamente. Durante los eclipses, el ángulo de evitación mínimo con la Luna es $35^{o}$.

### 2.1.3. Catálogo de fuentes de rayos-X

Las dos cámaras EPIC tienen un campo de visión de aproximadamente medio grado arco de diámetro y, por tanto, muy frecuentemente observan un área del cielo mucho mayor que la ocupada por la fuente central, que es normalmente la solicitada por el observador.

El XMM-Newton Survey Science Centre es un consorcio de 10 institutos europeos designados por la Agencia Espacial Europea (ESA). El consorcio tiene cuatro tareas dentro del proyecto XMM-Newton:
1) Proporcionar software para reducir y analizar los datos provenientes de XMM-Newton.
2) Procesar todos los datos para detectar las fuentes presentes en las imágenes, tanto las que aparecen en la zona central como las fortuitas -*serendipitous*- en el campo de visión.
3) Construir catálogos a partir de las fuentes de rayos X.
4) Realizar investigaciones para comprender mejor estas fuentes.

Pese a que el proceso de detección de fuentes es muy robusto, se pueden dar detecciones espurias debidas a fuentes muy brillantes, arcos de reflexión causados por fuentes brillantes fuera del campo de visión, grandes emisiones difusas o ruido anómalo en la región del detector [11].

Debido a este inconveniente, un equipo de expertos se encarga de inspeccionar visualmente cada una de las observaciones con el fin de validar la metodología empleada en la obtención de las observaciones, así como etiquetar las detecciones problemáticas. Las regiones afectas son marcadas siguiendo los criterios de los expertos y la fracción del campo de visión marcada es caracterizada por

el parámetro clase observacional (OBS_CLASS).

El catálogo 4XMM-DR9 contiene 810795 detecciones de rayos X de 550124 fuentes astronómicas seleccionadas de 11204 observaciones realizadas por el XMM-Newton entre el 1 de Febrero del 2000 al 26 de Febrero del 2019, ver tabla.1. [12]

| OBS_CLASS | Regiones enmascaradas | 4XMM-DR9 |
|---|---|---|
| 0 | área mala = 0 % | 4386 |
| 1 | 0 % < área mala < 0.1 % | 3484 |
| 2 | 0.1 % < área mala < 1 % | 1571 |
| 3 | 1 % < área mala < 10 % | 1198 |
| 4 | 10 % < área mala < 100 % | 544 |
| 5 | área mala = 100 % | 21 |

Tabla 1: OBS_CLASS, es la clase observacional dada en la primera columna, el porcentaje de regiones problemáticas en la imagen se encuentra en la segunda columna y el número de imágenes del catálogo 4XMM-DR9 para cada clase está en la tercera columna.

## 2.2. Estadística paramétrica

Existen coeficientes en estadística paramétrica que dan información sobre la forma que tienen las distribuciones de datos. Estimar algunos puede darnos una visión general de como se agrupan nuestros datos y por tanto, obtener una clasificación a priori. Valores paramétricos como la desviación estándar, el coeficiente de asimetría o la curtosis.

1) La desviación estándar, $\sigma$, es una medida de la dispersión de los datos con respecto a la media. [13]

$$\sigma = \sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2/(N-1)} \tag{1}$$

donde $\bar{x}$ es la media de los datos.

Si este indicador es bajo, el conjunto de datos tiende a agruparse cerca de su media, mientras que un valor alto indica que los datos ocupan un rango de valores más amplio.

2) El coeficiente de asimetría o *skewness* es una medida de la asimetría de la distribución de probabilidad de una variable aleatoria sobre su media. Se puede expresar como el coeficiente de asimetría de Fisher-Person, $g_1$[14]:

$$g_1 = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^3/N}{\sigma^3} \tag{2}$$

donde $\bar{x}$ es la media, $\sigma$ es la desviación estándar y $N$ es el número de datos.

El valor del skewness puede ser positivo, negativo o no definido.

- Si toma valores menores que -1 o mayores que 1, la distribución es muy sesgada
- Si su valor se encuentra entre -1 y -0.5 o entre 0.5 y 1, la distribución es moderadamente sesgada.
- Si el coeficiente se encuentra entre -0.5 y 0.5, la distribución es aproximadamente sesgada.

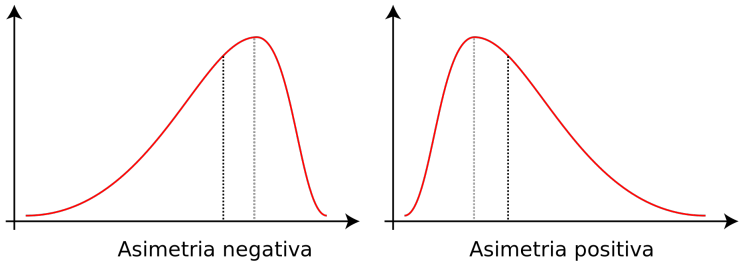



Figura 6: Curvas con coeficiente de asimetría negativo (izquierda) y positivo (derecha).La linea de puntos negros representa la media y la de puntos grises la moda. Imagen de [15]

3) La curtosis es una medida de forma que indica cuan escarpada o achatada está una distribución con respecto a una gaussiana. Se puede definir mediante el coeficiente de la curtosis de Fisher, $g_2$ [14]:

$$g_2 = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^4/N}{\sigma^4} - 3 \tag{3}$$

donde $\bar{x}$ es la media, $\sigma$ es la desviación estandar, y $N$ es el número de datos. Se suele añadir el -3 debido a que la curtosis de la distribución normal es 3.

- Si el coeficiente es 0, la distribución es normal o gausiana (mesocúrtica).
- Si el coeficiente toma valores positivos, la densidad de datos es mayor en torno a la media. Dando lugar a una figura más "puntiaguda"(leptocúrtica).
- Si el coeficiente toma valores negativos, la distribución de puntos se concentra menos en torno a la media, dando lugar a una forma más aplanada (platicúrtica).

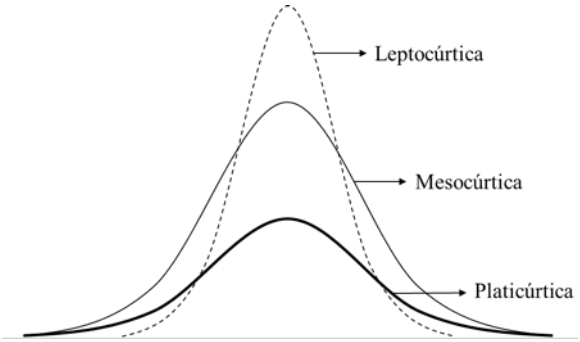



Figura 7: Representación de curvas mesecúrtica ($g_2 = 0$), leptocúrtica ($g_2 > 0$) y platicúrtica ($g_2 < 0$). Imagen de [16]

## 2.3. Inteligencia Artificial

Los trabajos precursores de lo que conocemos como Inteligencia Artificial moderna, aparecen en la década de los 40 del siglo XX, aunque no sería hasta 1950, con la publicación de Alan Turing *Computing Machinery and Intelligence*, cuando estos estudios consiguen repercusión. En al artículo de Alan Turing se teoriza sobre la posibilidad de que una máquina sea capaz de imitar el comportamiento

de la mente humana así como el conocido Test de Turing, diseñado para verificar si una máquina ha alcanzado la inteligencia suficiente como para ser indistinguible con la de un humano. [17]

La primera aproximación sobre estas ideas que triunfó, fue la de crear inteligencias artificiales basadas en como los humanos comprenden el mundo a través de representaciones simbólicas. Entonces, si un cerebro es análogo a un ordenador, cada situación que nos encontramos puede expresarse como un programa que sigue, paso a paso, una serie de reglas basadas en la lógica. Dando esto por válido, esas mismas reglas sobre la organización del mundo, pueden ser descubiertas y codificadas en algoritmos que un ordenador puede interpretar. A este tipo de algoritmos de inteligencia artificial se les conoce como inteligencia artificial simbólica.

La inteligencia artificial dio lugar a desarrollos interesantes como el sistema SIR (Semantic Information Retrieval), que era capaz de aprender relaciones entre objetos asemejándose a la inteligencia real en este tipo de razonamientos lógicos.[18]

El apogeo de las IA simbólicos fue alcanzado en la década de los 80s con los llamados "sistemas expertos"que intentan usar sistemas basados en reglas para resolver problemas del mundo real. Sin embargo, aunque los sistemas expertos son sólidos, tienen limitaciones como su elevado coste, requieren de actualizaciones constantes y pueden ser menos precisos cuantas más reglas incorporen. Además, por como están concebidos, no mejoran con el tiempo ni se adaptan a nuevas situaciones debido a que se basan en reglas estrictas.

## 2.4. Aprendizaje automático

Hoy en día el paradigma está mayormente dominado por los algoritmos de aprendizaje automático o aprendizaje de las máquinas (*machine learning, ML).* Estos algoritmos tratan de abordar el problema de la inteligencia artificial de forma opuesta a los algoritmos simbólicos, ya que pretenden descubrir las reglas lógicas que explican el pensamiento humano a partir de datos reales del comportamiento deseado.

Los algoritmos de *machine learning* se dividen en tres categorías principales: aprendizaje supervisado, no supervisado y semisupervisado. [19]

1) En el aprendizaje supervisado se tienen variables de entrada (X) con sus correspondientes variables de salida o etiquetas (Y) y es el algoritmo el que se encarga de descubrir la función que liga estas variables. Desde un punto de vista probabilístico, el sistema supervisado trata de estimar la probabilidad bayesiana $p(y|X)$, aprendiendo a predecir el valor de y dado X

$$Y = f(X) \tag{4}$$

El objetivo es aproximar la función lo suficiente para que al introducir nuevos datos, el algoritmo sea capaz de predecir las variables de salida para esos datos.

Los problemas de aprendizaje supervisado se pueden agrupar según el tipo de dato de salida. Un problema de clasificación se emplea cuando la salida es una categoría y un problema de regresión cuando el dato de salida es un número real.

2) En el aprendizaje no supervisado solo se tienen los datos de entrada (X) y no sus correspondientes variables de salida. Desde el punto de vista estadístico, tratan de hallar directamente la distribución de probabilidad $p(X)$ y las propiedades relevantes de dicha distribución.

El objetivo de este tipo de entrenamientos es el de modelar la estructura de los datos para aprender sobre los propios datos. Se utilizan en problemas de agrupación.

3) Aprendizaje semisupervisado

En los problemas de aprendizaje supervisado, parte del set de datos de entrada (X) está etiquetada (Y) y otra parte no. Por lo tanto se emplean técnicas de aprendizaje supervisado y no supervisado.

## 2.5. Redes neuronales

Las redes neuronales son uno de los algoritmos de *machine learning* más importantes. Se basan en conjuntos de nodos o neuronas artificiales, que pretenden imitar el comportamiento de los axones de las neuronas en un cerebro.

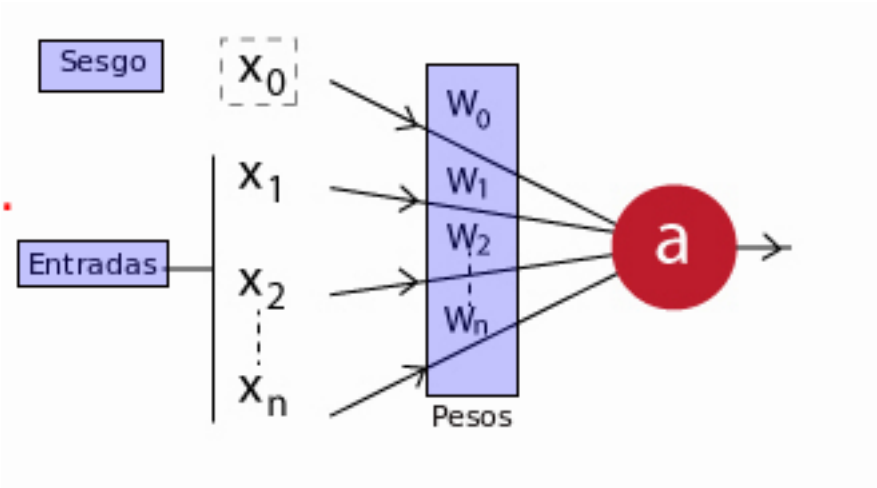



Figura 8: Diagrama de una neurona.

- $x_1, x_2, ..., x_n$: Datos de entrada en la neurona. Pueden ser los datos de salida de otra neurona de la red.

- $x_0$: La neurona sesgo o *bias*, es una neurona que se añade en cada capa de una red neuronal y alberga el valor 1 generalmente.

- $w_0, w_1, w_2, ..., w_n$: Son los pesos relativos de cada entrada. Pueden tener valores positivos o negativos, dependiendo de su una neurona activa o inhibe otra.

- $a$: Es el producto de salida de la neurona, que se estima mediante la siguiente ecuación:

$$a = f\left(\sum_{i=0}^{n} w_i \cdot x_i\right) \tag{5}$$

Donde $f$ es la *función de activación* de la neurona.

Las neuronas se organizan en capas y la combinación de estas capas da lugar a arquitecturas más complejas. Normalmente la arquitectura de una red neuronal se divide en tres partes: la capa de entrada, con neuronas que representan los campos de entrada; una o más capas ocultas; y una capa de salida con una o más unidades que representan el resultado computado por la red. En la primera capa se introducen los datos de entrada, y los valores se van propagando desde cada neurona hasta las neuronas de la capa siguiente. Finalmente la capa de salida devuelve un resultado.

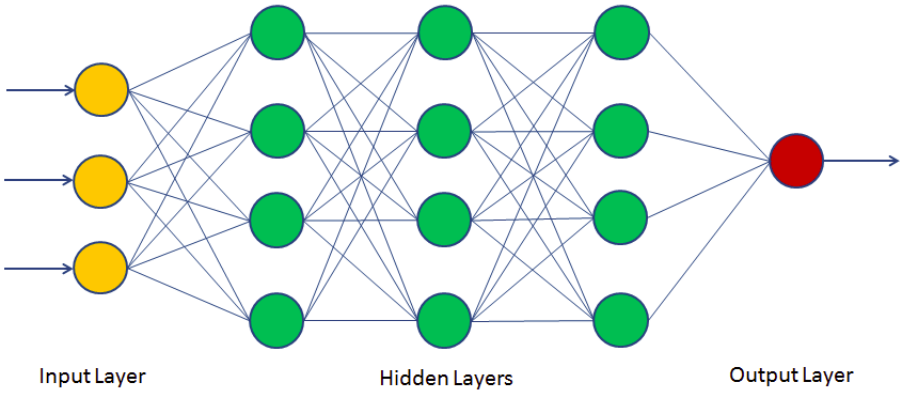



Figura 9: Estructura típica de una red neuronal. Incluye la capa de entrada, algunas capas ocultas y la capa de salida.

En algoritmos supervisados, la red aprende a partir de un proceso de retroalimentación llamado propagación hacia atrás de errores o *Backpropagation*, que consiste en comparar la señal de salida generada por la red con la deseada y estima un error para cada una de las salidas. Estos errores se propagan hacia atrás desde la capa de salida iterativamente hasta que todas las neuronas de la red hayan recibido una señal que indique su contribución relativa al error total. Así el Backpropagation actualiza los pesos de cada neurona, con lo que va reduciendo la diferencia entre el resultado esperado y el obtenido.

## 2.6. Aprendizaje profundo y redes neuronales convolucionales

El aprendizaje profundo o *deep learning* es un subcampo del aprendizaje automático, que se caracteriza por el aprendizaje en capas sucesivas de representación de datos cada vez más complejos. Por ello también se denomina aprendizaje de representación jerárquica. El número de capas que posee el modelo representa la profundidad del mismo e implica el grado de abstracción del aprendizaje.

Las redes neuronales convolucionales, o CNN, son un tipo particular de estas representaciones por capas.

El factor diferenciador de este tipo de redes neuronales son sus capas de convolución. La operación de convolución, en el contexto del modelo, consiste en la multiplicación de una matriz de entrada y una matriz bidimensional de pesos, llamada filtro o kernel.

Normalmente, estos filtros son mas pequeños que la matriz de entrada, por lo que le permite realizar esta operación sistemáticamente a medida que recorre la imagen. El filtro aprende a reconocer alguna característica de las imágenes de entrada sin importar la posición de dicha característica en la matriz de píxeles, lo que implica invarianza traslacional.

Las redes convolucionales aprovechan la invarianza traslacional (cambios en la posiciones de un elemento no alteran el resultado final) para usar los mismos parámetros en el filtro convolucional y reducir las imágenes con el pooling.

El output de multiplicar el kernel por la matriz de píxeles una vez es un único valor. Como el filtro es aplicado varias veces sobre la imagen, el resultado es otra matriz bidimensional conocida como mapa de características.

Habitualmente, las capas de convolución contienen varios filtros diferentes, de esta forma, las redes neuronales pueden aprender a identificar múltiples características.

Las capas convolucionales no son solo aplicables a la matriz de entrada, también pueden aplicarse a la salida de otras capas. Si por ejemplo consideramos que los filtros que operan sobre los píxeles directamente, extraen características de bajo orden como líneas, los filtros que reciben como entrada los mapas de características puede que sean capaces de extraer combinaciones de características de menor nivel.

Este proceso continua hasta que las capas más profundas son capaces de extraer formas complejas como rostros, animales o señales de tráfico.

Después de cada convolución se aplica una función de activación, cuyo objetivo es mapear la salida de las neuronas en valores entre 0 y 1 o -1 y 1 [20]. Existen diversas funciones de activación, siendo la función ReLU (Rectified Linear Unit) la más extendida para redes neuronales convolucionales en los últimos años.

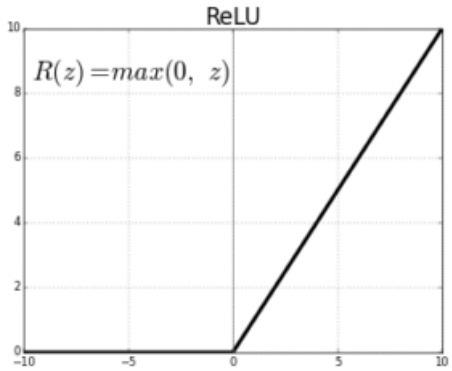



Figura 10: Función ReLU. Imagen de [21]

Algunos de los beneficios que explican su popularidad son los siguientes:

a) Simplicidad computacional: Es una función simple de implementar, requiriendo unicamente una función max(). A dificerencia de otras funciones de activación, como la tangencial o la sigmoide, que requieren del cálculo de exponentes.

b) Verdaderos ceros: A diferencia de las funciones tangencial y sigmoide, que únicamente son capaces de aproximarse a una salida con valor cero, la función ReLU si puede representar verdaderos ceros.

c) Comportamiento lineal: Rectified linear units [. . . ] are based on the principle that models are easier to optimize if their behavior is closer to linear.

Otra capa habitual en la arquitectura de redes neuronales convolucionales es la de Pooling. Se incluye entre capas de convolución con el objetivo de reducir progresivamente el tamaño de las matrices, reduciendo así el número de parámetros y la carga computacional de la red. El uso habitual consiste en filtros de tamaño 2x2, aplicados a toda la muestra con un corrimiento de 2 de alto y ancho, seleccionando el valor máximo de entre los 4 números sobre los que aplica cada paso ver Figura 14. Este procedimiento particular se conoce como MaxPooling, aunque existen otros métodos de Pooling que efectuan otras funciones, como la media (Average Pooling).

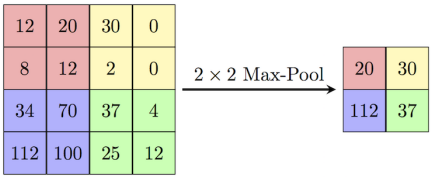



Figura 11: Ejemplo de una operación de Max Pooling. Imagen de [22]

Después de las capas convolucionales y de pooling, se añade una capa de aplanamiento o *flatten layer*. Esta capa pasa los valores del mapa de características a un vector unidimensional.

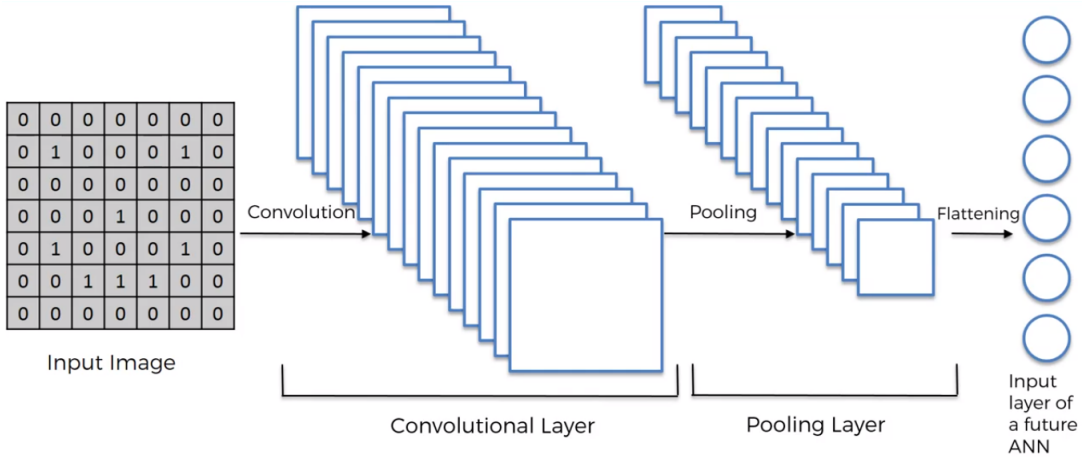



Figura 12: Representación esquemática de la reducción de parámetros de la matriz de entrada pasando por las capas convolucionales y de pooling. La matriz de características es transformada en un vector por la capa de aplanamiento. Imagen de [23]

Finalmente se añaden capas densas, que se corresponden con redes neuronales clásicas, donde cada neurona de la capa recibe la señal de todas las neuronas de la capa previa. Se utilizan para realizar la clasificación de final.

### 2.6.1. Evaluación de un modelo de aprendizaje automático

Existen múltiples métricas que se deben monitorizar durante el entrenamiento de una red neuronal. Estos valores nos dan una idea sobre los parámetros de nuestra red y como deberían modificarse para que el aprendizaje sea más efectivo.

Habitualmente estas cantidades se representan en unidades de épocas, que miden el número de veces que cada muestra ha sido vista por la red durante el entrenamiento. Es preferible monitorizar las épocas en vez de las iteraciones, ya que el número de iteraciones depende del número de muestras que son propagadas por la red al mismo tiempo (*batch size)*.

- Función de perdida

Como se explicó anteriormente, las redes neuronales ajustan su aprendizaje mediante la reducción de la diferencia entre los valores de entrada y de salida. Este error es una función que queremos minimizar o maximizar conocida como función objetivo o criterión. Cuando queremos minimizar esta función, también se puede llamar función de coste, función de perdida o función de error y es una de las métricas a tener en cuenta a la hora de evaluar el rendimiento de un modelo.

- Precisión

Es el ratio del número de predicciones correctas frente al número total de muestras de entrada.

- ROC y AUC

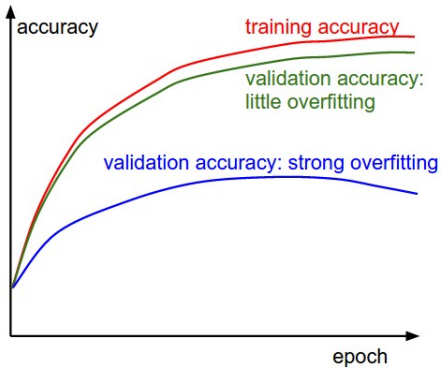



Figura 13: Representación esquemática de la precisión frente a las épocas. Se muestran curvas de precisión para sets de datos con los que entrena la red (*training*) y datos con los que valida su funcionamiento (*validation*). La separación entre estas curvas indica el grado de overfitting. En el caso de la curva de validación azul, el sobreajuste es elevado, ya que su tasa de acierto es muy inferior a la de entrenamiento y por lo tanto se deben tomar medidas para reducirlo. En el caso de la curva verde, la separación no es tan grande, por lo que el sobreajuste no es muy elevado, es decir, el modelo es capaz de generalizar ofreciendo una precisión similar a nuevas muestras. Imagen de

La curva ROC (*Receiver Operating Characteristic* o Característica Operativa del Receptor) representa el rendimiento de un modelo en todos los umbrales de clasificación. [24]

Esta curva representa la tasa de verdaderos positivos, $TPR$, frente a la tasa de falsos positivos, $FPR$

Tasa de verdaderos positivos:

$$TPR = \frac{VP}{VP + FN} \tag{6}$$

donde $VP$ son los verdaderos positivos y $FN$ los falsos negativos.

Tasa de falsos positivos:

$$TPR = \frac{FP}{FP + VN} \tag{7}$$

donde $FP$ son los falsos positivos y $VN$ son los verdaderos negativos.

Si se reduce el umbral de clasificación, se clasifican más elementos como positivos, por lo tanto aumentarán tanto los falsos positivos como los verdaderos positivos. Entonces, para estimar todos los puntos de la curva ROC, se podría evaluar el modelo varias veces modificando este umbral. Sin embargo esto resulta ineficiente y por lo tanto se recurre a la AUC (*Area Under the ROC o área bajo la curva*).

Este indicador integra el área bajo la ROC, proporcionando un valor agregado en todos los umbrales posibles.

### 2.6.2. Sobreajuste

Un modelo entrenado tiene sobreajuste cuando es capaz de clasificar o predecir con gran exactitud sobre los datos que fueron incluidos en el set de entrenamiento pero no tiene resultados

tan buenos con datos que no fueron utilizados en su entrenamiento. Es decir, el modelo no es bueno generalizando.

Una forma de reducir el sobreajuste es añadiendo más datos al set de entrenamiento. De esta forma, el modelo obtendrá mayor diversidad de muestras con las que poder generalizar mejor.

Sin embargo, en ocasiones no es posible obtener más muestras con las que entrenar nuestro modelo. Por tanto, es necesario emplear técnicas de regularización como generación de datos artificiales o *Data Augmentation*, abandono o *dropout* y normalización de lotes o *batch normalization*.

La generación de datos artificiales consiste en crear más muestras mediante la pequeña modificación de las imágenes del set de entrenamiento. Por ejemplo, rotando la imagen, invirtiéndola vertical u horizontalmental o desplazándola.

El dropout es una forma efectiva y simple de prevenir el sobreajuste en redes neuronales. Mientras el algoritmo esta entrenando, el dropout es implementado manteniendo a la neurona activada o desactivada con una probabilidad. Esto hace que cada capa de neuronas con dropout funcione como otra capa diferente con otro número de nodos y conexiones con otras neuronas. Es decir, cada entrenamiento se realiza desde un punto de vista diferente. *p.* [25]

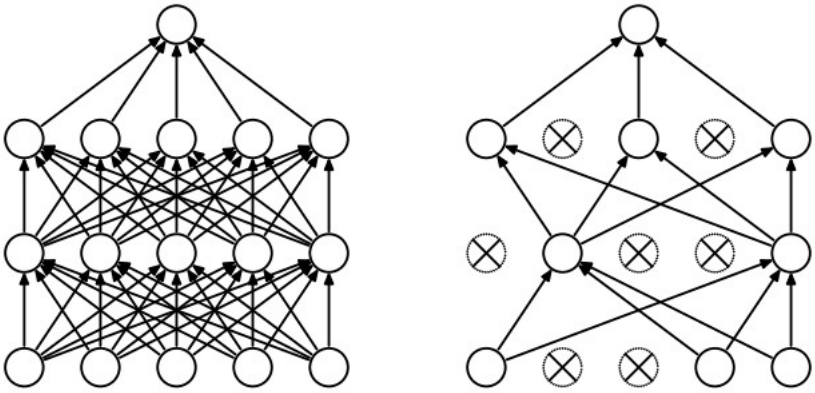



Figura 14: Modelo de red neuronal con dropout. A la izquierda: Una red neuronal con 2 capas ocultas. A la derecha: La misma estructura, pero con algunos nodos desactivados. Las neuronas abandonadas no interaccionan con las neuronas de la capa posterior. Imagen de [25]

Esta técnica ayuda a evitar el sobreajuste en cuanto a que evita situaciones en las que la red se adapta a fallos de capas anteriores.

# 3. Análisis del problema

En este apartado se planteará el problema a resolver, junto con los datos que de los que partimos y las técnicas empleadas para obtener el resultado deseado.

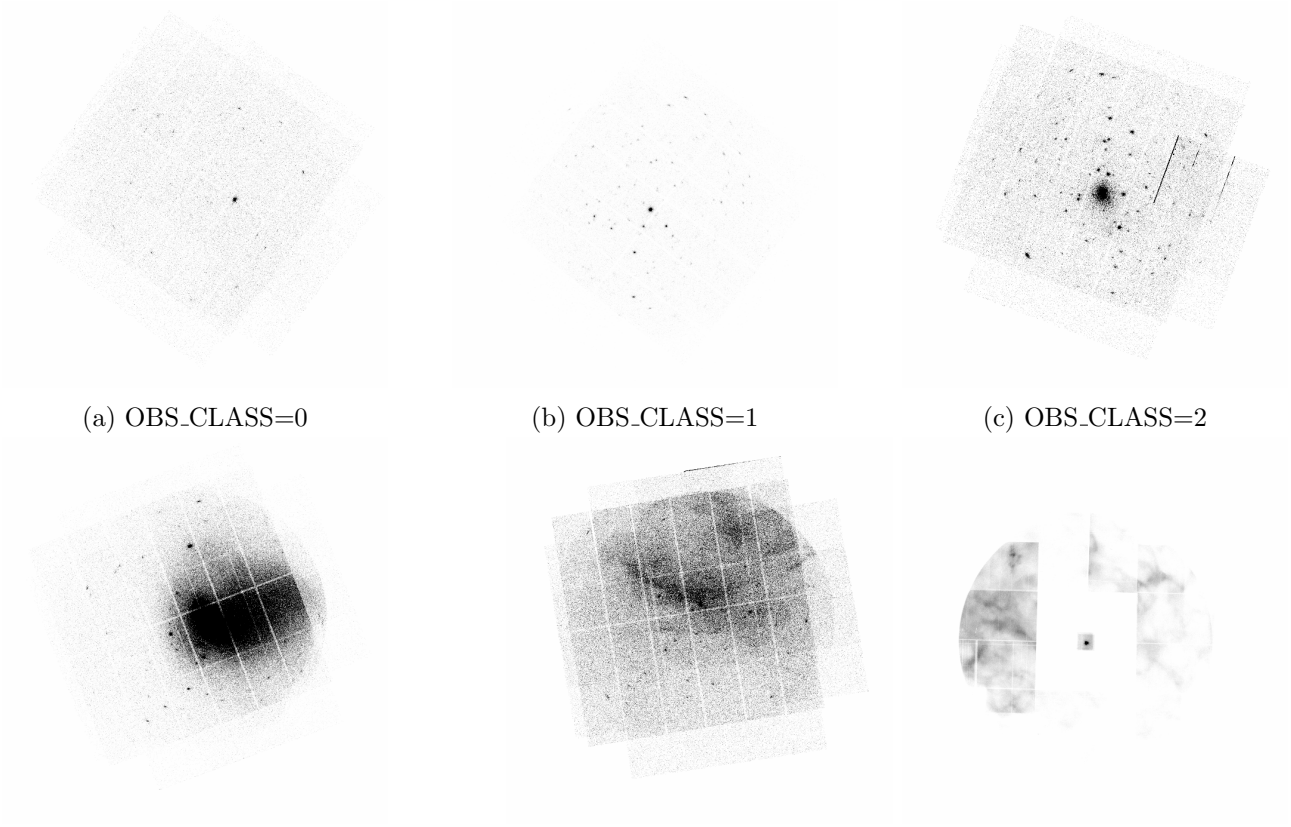
## 3.1. Datos del problema

El catálogo de rayos X empleado en la realización del proyecto es el 4XMM-DR9, que contiene detecciones de fuentes en 11204 observaciones realizadas por el XMM-Newton.

Las imágenes se encuentran en formato FITS ,epresentadas por matrices de píxeles cuyos valores indican las cuentas totales en cada posición. Las cuentas están en la banda entre 0.2 y 12 keV.

La dimensión de cada imagen es de 648x648 píxeles. Algunos ejemplos de estás imagenes se pueden ver en la figura.15.

Además se dispone de una tabla con información sobre cada imagen: Un identificador o etiqueta, la clase observacional, los tiempos de exposición de cada cámara y otros datos adicionales no empleados en el desarrollo del trabajo.



(a) OBS_CLASS=0 (b) OBS_CLASS=1 (c) OBS_CLASS=2

(d) OBS_CLASS=3 (e) OBS_CLASS=4 (f) OBS_CLASS=5

Figura 15: Ejemplos de imagenes con diferente clase observacional

Todas las imágenes se normalizan a su correspondiente tiempo de exposición para conseguir imágenes que representen cuentas por segundo, que son más comparables con otras observaciones con distintos tiempos de exposición.

## 3.2. Pruebas paramétricas

En primera aproximación al problema, se estiman coeficientes paramétricos para cada imagen (desviación estándar, skewness y curtosis) con la motivación de obtener una distribución de las imágenes que nos permita agruparlas en categorías o identificar algunas que se salgan de la distribución para analizarlas posteriormente.

Para calcular estos parámetros se empleó el paquete scipy.stats. Que utiliza las definiciones expuestas en la sección 2.2.

## 3.3. Tratamiento de imágenes

Antes de emplear métodos de aprendizaje automático sobre los datos del problema, es conveniente tratar los mismos. Si nosotros no somos capaces de extraer características visuales de una

imagen, es más probable que la red neuronal tenga mayores dificultades para extraer las reglas lógicas que permitan clasificar las observaciones.

### 3.3.1. Rebinning

Debido a la gran cantidad de imágenes que se tienen que procesar, es conveniente reducir el tamaño de las mismas para realizar pruebas sin tener problemas de capacidad computacional y tiempo.

Se probaron reducciones de escala en factores 2×2 y 4×4 para subsanar estas limitaciones. Utilizar el set de datos sin reescalado no es viable, debido al gran uso de RAM necesario para poder cargar todas las imágenes.

Para realizar el rebinning se empleo el método de interpolación linear de la función *resize* del paquete *cv2*. Dado que las reducciones propuestas son divisores enteros de la dimensión de cada imagen, esta operación equivaldría a realizar la media de pequeñas matrices 2x2 o 4x4 sobre toda la imagen.

### 3.3.2. Normalización

A simple vista, es complicado clasificar imágenes no normalizadas, lo que nos da la intuición de que los algoritmos podrían reconocer mejor las áreas buenas de las imágenes si aplicamos operaciones de normalización.

Por tanto, se utilizarán distintos tipos de normalización utilizados comúnmente en análisis de imágenes astronómicas como el seno hiperbólico, el logaritmo o la raíz cuadrada, así como intervalos minmax y zscale.

Se empleó el paquete de python astropy, ya que contiene funcionalidades y herramientas comunes en el tratamiento de imágenes en formato FITS. Entre los métodos que posee, se encuentran los distintos tipos de normalizaciones e intervalos característicos en el tratamiento de imágenes astronómicas.

### 3.3.3. Definir sets de entrada

En los algoritmos de clasificación por aprendizaje automático es importante que la cantidad de muestras por categoría este balanceada, ya que si una categoría es predominante, el modelo entrenado tenderá a aprender a diferenciar las características de la clase con más muestras. Debido a que las clases observacionales 2, 3, 4 y 5 contienen menos imágenes y son las de peor calidad (contiene más de un 0.1 % del área total clasificada como mala), se agrupan en una única categoría y el resto, las consideradas de mejor calidad, en otra categoría (OBS_CLASS=0,1). También se distinguen el set de entrada $X$, que contiene los arrays que representan los píxeles de cada imagen, y el set de salida $Y$ que contiene categoría de cada imagen (buenas o malas).

Por otro lado, es habitual dividir el conjunto de datos de entrada en sets de entrenamiento, validación y test. La motivación detrás de esto, es la de entrenar la red con una porción del total de imágenes (set de entrenamiento) y validar el funcionamiento de modelo entrenado con una porción del set que la red no tuvo en cuenta a la hora de entrenar.

(a) Sin normalizar

(b) logarítmica

(c) seno hiperbólico

(d) raíz cuadrada

Figura 16: Representación de la misma imagen con diferentes normalizaciones y en escala minmax. El mapa de colores está invertido. En la imagen sin normalizar a penas se perciben las fuentes y defectos. En cambio, en la imagen c se visualizan las zonas de interés con más claridad. En las imagenes b y d se aprecian con claridad los bordes de los detectores.

### 3.4. Modelos de redes neuronales convolucionales

Se emplearán redes convolucionales neuronales ya que son las más indicadas para la clasificación de imágenes, debido a su gran capacidad para extraer patrones.

Construimos el modelo a aprendizaje profundo desde cero, realizando pruebas con diferentes arquitecturas para probar que capas o parámetros funcionan mejor y así añadirlos a la red haciéndola cada vez más compleja.

Se empezó diseñando una configuración de capas típica de redes neuronales convolucionales, con capas de convolución seguidas de capas de *MaxPooling*, la capa de *flattening* y las capas densas, siendo la última de estas la de salida. Ver figura 27. Todas las capas usan la función de activación *ReLu* exceptuando la de salida, que emplea la función de activación sigmoide para que el rango de valores vaya de 0 a 1. De esta forma no solo se obtendrá información sobre si la predicción es buena o mala, sino cuanto de buena o mala es esa predicción.

| OBS_CLASS | Entrenamiento | Validación | Test |
|---|---|---|---|
| 0 | 3367 | 862 | 157 |
| 1 | 2684 | 690 | 110 |
| 2 | 1220 | 293 | 58 |
| 3 | 927 | 214 | 57 |
| 4 | 425 | 101 | 18 |
| 5 | 20 | 1 | 0 |
| Total | 8643 | 2161 | 400 |

Tabla 2: Distribución de las imágenes en sets de entrenamiento, validación y test.

Una red neuronal convolucional con tres capas de convolución es capaz de obtener características complejas, pero a su vez, si el set de datos empleado no requiere de tal grado de complejidad, es probable que esta arquitectura produzca sobreajuste en los modelos entrenados. Es por esto que también se probará otra arquitectura más sencilla con dos capas convolucionales.

La implementación de la arquitectura planteada se realizó con la interfaz de programación de aplicaciones *keras*. Esta API es una de las más reconocidas en el entorno científico (CERN, NASA, NIH) debido a su diseño orientado hacia la sencillez de uso. Está construido sobre *TensorFlow*, que se encarga de realizar las operaciones de bajo nivel.

Después de definir las capas de nuestra red convolucional, se compila el modelo indicando los parámetros de optimización y perdida. En nuestro caso se empleó el optimizador *Adam* y la función de perdida *Binary Crossentropy* ya que es la más apropiada para problemas de clasificación con dos posibles salidas.

En el Apéndice.A se muestra con código un ejemplo de como se configura la CNN y como se entrena el modelo.

### 3.4.1. Tamaño de filtro

Al observar las imágenes, se intuyen zonas que podrían aportar información a los algoritmos. Como estas zonas tienen diferentes tamaños, pensamos que probar filtros de diferentes dimensiones podría ayudar a extraer toda la información de cada zona mediante una operación de convolución. Por ello se probarán kernels de tamaños 3x3, 5x5, 7x7, 9x9... En función de sí el ordenador es capaz de realizar estas operaciones en un tiempo razonable.

## 3.5. Formas de hacer frente al Overfitting

Con el objetivo de prevenir el sobreajuste, se probaron técnicas de regularización.

Dropout: Se añade una capa de dropout después de la primera capa densa con ratios de abandono de 0.2 y 0.4, por debajo de 0.6 como se recomienda en [25].

Batch normalization: Se prueba está técnica de estandarización del set de entrada con el objetivo de acelerar el aprendizaje, además de intentar reducir el overfitting.

Data augmentation: Debido a desbalance entre los sets de imágenes buenas y malas, se prueba a generar muestras artificiales mediante operaciones sobre la imagen que no alteren la física de la misma. Estas operaciones son rotaciones en un rango entre $0^{\underline{o}}$ y $180^{\underline{o}}$, volteos verticales y volteos horizontales.

# 4. Resultados

En esta sección se mostraran los resultados obtenidos siguiendo los procedimientos anteriormente descritos, así como la valoración de los mismos.

## 4.1. Resultados de la estadística paramétrica

A partir del cálculo de los parámetros $\sigma$, $g_1$ y $g_2$ para cada imagen, se realizó una representación gráfica en 3 dimensiones. En la siguiente figura se muestra dicho gráfico desde dos perspectivas.

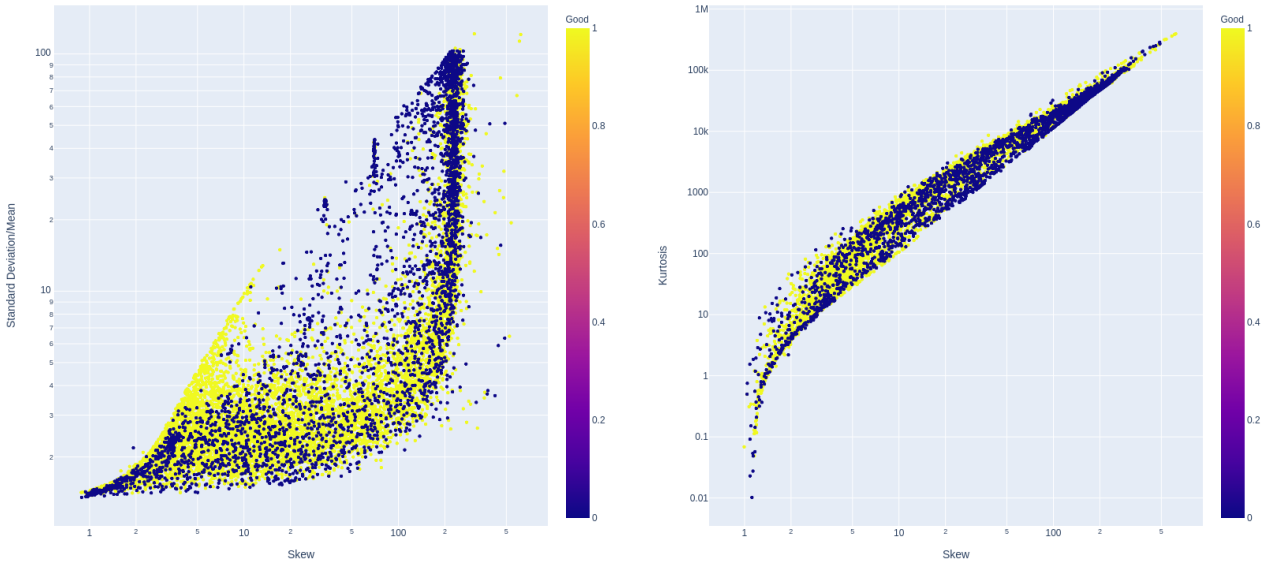



Figura 17: Representación de los coeficientes de forma (desviación estándar, skewness y curtosis) de cada imagen en escala logarítmica. Los puntos azules representan las imágenes *buenas* (OBS_CLASS=0,1) y los puntos amarillos las imágenes *malas* (OBS_CLASS=2,3,4,5).

De acuerdo con la nube de puntos de la Figura.17, no se puede extraer una separación de categorías clara.

## 4.2. Resultados de pruebas con redes neuronales convolucionales

Puesto que los factores de forma de las curvas de las imágenes no sirven a priori para obtener una distribución por clases. Está justificado el uso de técnicas más complejas, como el aprendizaje automático y en particular las redes neuronales convolucionales.

## 4.3. Pruebas con tamaños de filtros

Los resultados obtenidos para las pruebas con diferentes tamaños de filtros de convolución se encuentran en la tabla. 3. Las métricas de precisión y perdida son ligeramente mejores con los kernels (5, 5), (7, 7) y (9, 9). Sin embargo, esta mejora conlleva tiempos de entrenamiento mucho más largos: de 95, 140 y 341 segundos por época aproximadamente. Por lo tanto, decidí escoger los filtros (3, 3) para las demás pruebas, ya que dan resultados similares en menos tiempo.

| kernel | Training accuracy | Validation accuracy | Training loss | Validation loss | t/e (s) | AUC |
|---|---|---|---|---|---|---|
| (3, 3) | 0.7757 | 0.7825 | 0.4886 | 0.4679 | 60 | 0.774 |
| (5, 5) | 0.7759 | 0.7922 | 0.4823 | 0.4600 | 95 | 0.793 |
| (7, 7) | 0.7814 | 0.7894 | 0.4687 | 0.4670 | 140 | 0.784 |
| (9, 9) | 0.7804 | 0.7913 | 0.4743 | 0.4559 | 341 | 0.807 |

Tabla 3: Valores de las métricas accuracy y loss para la época con menor loss de validación. Se recogen estos valores para los diferentes tamaños de kernel probados en la arquitectura con tres capas convolucionales. En la penúltima columna se muestra el tiempo, $t$, que tardó dicha época, $e$, en completarse y en la última columna se encuentran las AUC de las curvas ROC para cada modelo. Las imágenes empleadas tienen un rescalado 4x4 y solamente están normalizadas al tiempo de exposición.

En la figura.18 se observa como el la red neuronal aprende a clasificar mejor las imágenes de entrenamiento con cada época. Sin embargo, la precisión del conjunto de validación se estanca rápidamente, es decir, la red no está generalizando todavía.

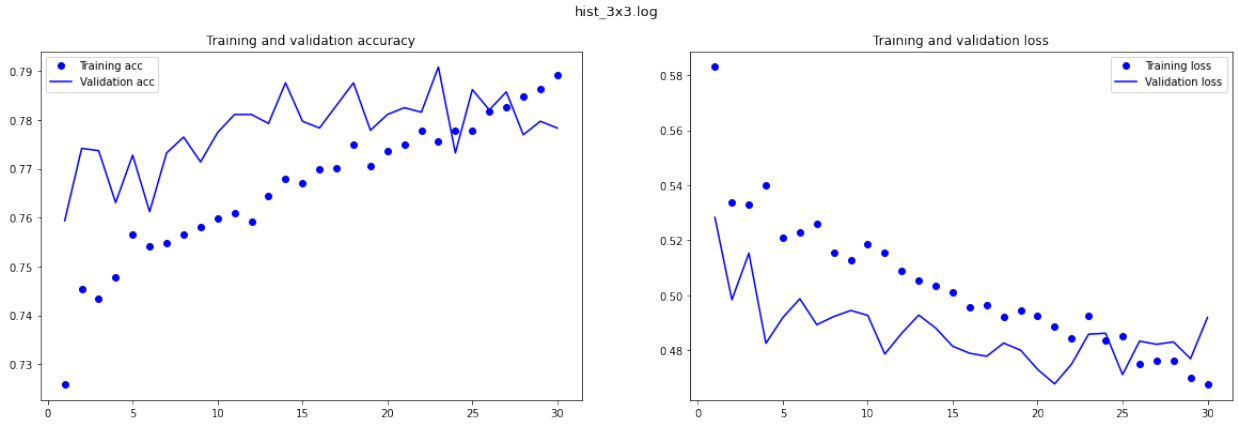



Figura 18: Desarrollo de las métricas accuracy y loss frente a las épocas durante el entrenamiento de la red con tres capas convolucionales. La curva de puntos representa el entrenamiento y la curva continua la validación. A la izquierda: Precisión en entrenamiento y validación. A la derecha: Perdida en entrenamiento y validación.

En la figura.19 se puede observar como la curva ROC de cada modelo tienen un área bajo la curva similar y por ende, el uso de unos filtros u otros no conlleva grandes mejoras en la precisión.

## 4.4. Pruebas con normalizaciones

En esta sección se muestran los valores de las métricas obtenidas utilizando las imágenes rescaladas 4x4 y con varios tipos de normalizaciones e intervalos.

Según los resultados de la tabla.4, los modelos entrenados con normalizaciones logarítmica o raíz cuadrada dan resultados ligeramente superiores, destacando esta última con un intervalo MinMax que tiene un valor de la AUC superior. Las pruebas posteriores se hicieron tomando la normalización sqrt con intervalo MinMax.

En la figura.20 se observa como las curvas entrenamiento y validación comienzan a separarse notablemente a partir de la décima época. Esto se debe a que el modelo sigue mejorando su precisión con los datos de entrenamiento, pero no refleja esa mejora a la hora de predecir datos de validación. Por lo tanto la red tiene sobreajuste y se deben realizar otras pruebas para reducir este efecto.

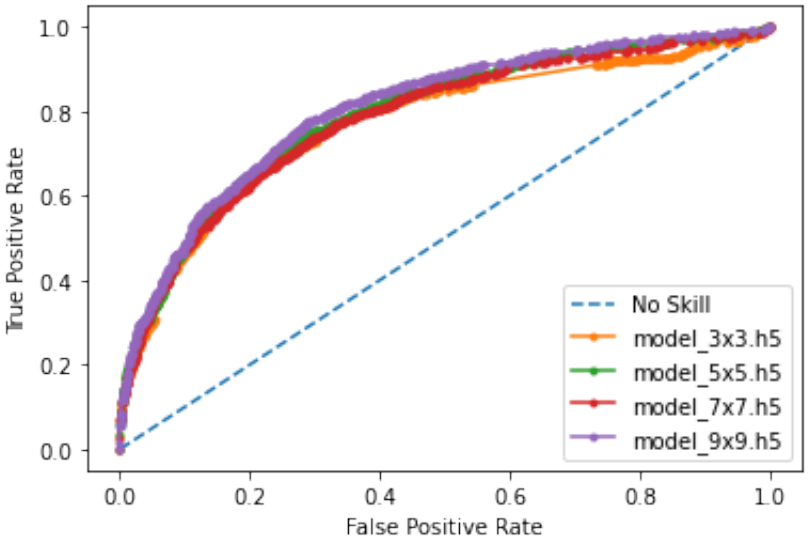



Figura 19: ROC obtenidas con las predicciones de cada modelo con diferentes filtros sobre el set de validación. Los valores del AUC se encuentran en la tabla 3

| Intervalo/Norm. | Training accuracy | Validation accuracy | Training loss | Validation loss | AUC |
|---|---|---|---|---|---|
| MinMax Log | 0.8149 | 0.8177 | 0.3946 | 0.4016 | 0.837 |
| zScale Log | 0.8179 | 0.8112 | 0.3906 | 0.3992 | 0.829 |
| MinMax Sqrt | 0.8170 | 0.8103 | 0.3941 | 0.4009 | 0.853 |
| zScale Sqrt | 0.7952 | 0.8130 | 0.4347 | 0.4094 | 0.829 |
| MinMax Power | 0.7398 | 0.7533 | 0.5212 | 0.5008 | 0.732 |
| zScale Power | 0.7924 | 0.8066 | 0.4400 | 0.4102 | 0.826 |
| MinMax Sinh | 0.7819 | 0.7802 | 0.4404 | 0.4320 | 0.809 |

Tabla 4: Resultados obtenidos con el modelo de tres capas convolucionales para diferentes normalizaciones e intervalos de las muestras de entrada. Se muestran las métricas accuracy y loss para el entrenamiento y validación del modelo, además de la AUC de cada curva ROC.

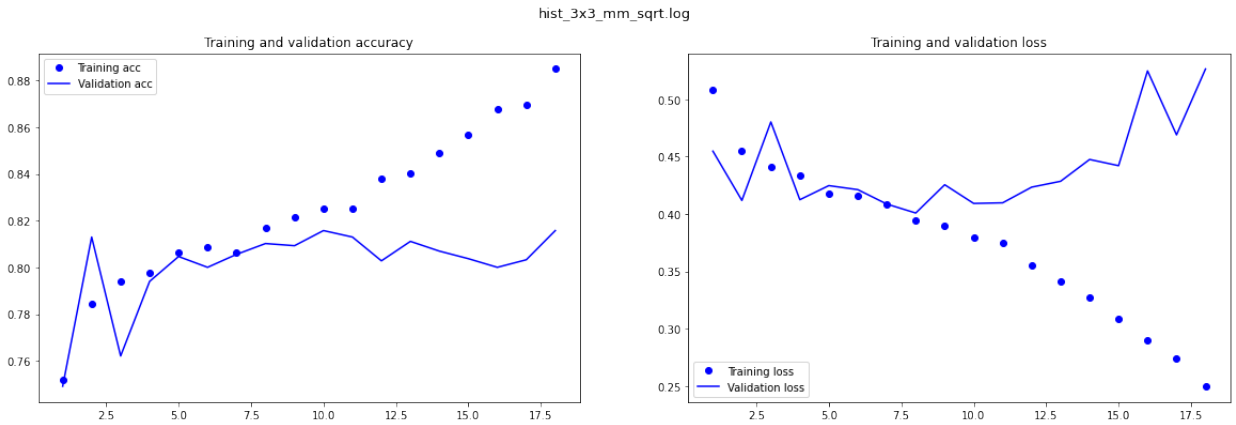



Figura 20: Representación gráfica de métricas de evaluación del modelo frente a las épocas, que se entrenó con muestras normalizadas a la raíz y con intervalo MinMax. A la izquierda se muestran las curvas de la precisión para el entrenamiento (curva de punteada) y la validación (curva continua). A la derecha se muestra la perdida para el entrenamiento y la validación del modelo. Se puede observar que a partir de la época 10 aproximadamente, las curvas de entrenamiento y validación divergen.

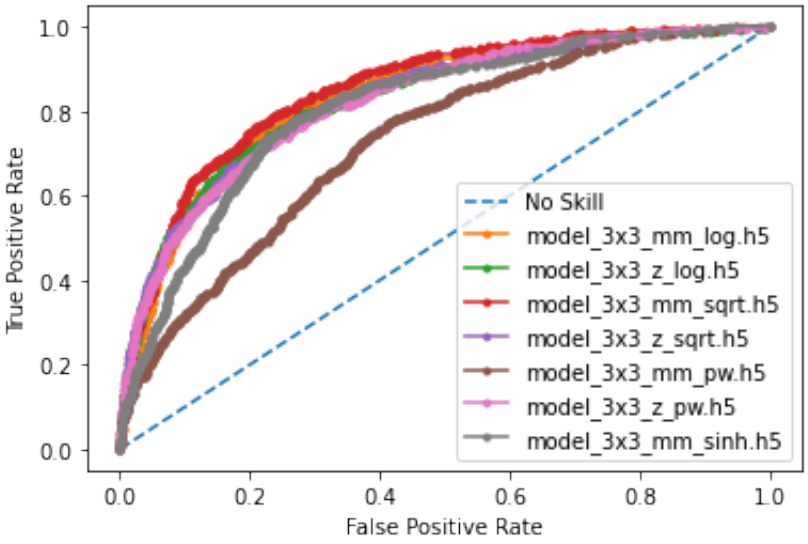



Figura 21

## 4.5. Pruebas de regularización

Como se comentó en los resultados de las pruebas anteriores, la precisión y la perdida que dan el conjunto de imágenes de validación al pasar por la red, no se ajusta a las métricas que obtiene durante el entrenamiento. Por lo tanto, se utilizaron algunas técnicas de regularización habituales, cuyos resultados se muestran en la tabla.5.

| Regularización | Training accuracy | Validation accuracy | Training loss | Validation loss | AUC |
|---|---|---|---|---|---|
| Dropout=0.2 | 0.8042 | 0.8135 | 0.4187 | 0.4004 | 0.848 |
| Dropout=0.4 | 0.8158 | 0.8130 | 0.3960 | 0.4029 | 0.846 |
| Dropout=0.5 | 0.8192 | 0.8126 | 0.3976 | 0.4088 | 0.848 |
| Batch Norm | 0.8489 | 0.7867 | 0.3295 | 0.4549 | 0.818 |
| $0^o$-$180^o$ | 0.8056 | 0.7978 | 0.4198 | 0.4217 | 0.866 |
| vflip | 0.8081 | 0.8140 | 0.4135 | 0.3925 | 0.863 |
| hflip | 0.8153 | 0.8195 | 0.3981 | 0.3950 | 0.865 |

Tabla 5: Resultados obtenidos al emplear técnicas para disminuir el sobreajuste. Se utilizaron imágenes normalizadas a la raiz cuadrada en intervalo MinMax y se muestran las métricas obtenidas en la época con menor perdida en validación. En las tres primeras filas están los resultados para diferentes valores de dropout. En la cuarta la prueba con *batch normalization*. Y en las tres últimas, se introdujo sets de datos artificiales creados con data augmentation (Rotaciones $0^o$-$180^o$ y volteos verticales *vflip* y horizontales *hflip*

De los modelos con dropout testados, el modelo con dropout=0.5 da resultados ligeramente superiores, por lo que se añadirá a las siguientes pruebas. Los modelos entrenados con muestras artificiales ofrecen mayor generalización a la hora de clasificar nuevas imáginas, así que también se tendrán en cuenta en las próximas pruebas. En cambio, la técnica *batch normalization*, aunque acelera el aprendizaje de la red, generó mucho sobreajuste como se puede observar en la figura.

El overfitting también se puede disminuir simplificando la arquitectura de la red neuronal. A parte de la arquitectura empleada en las pruebas anteriores, con tres capas de convolución, se probó otra con dos capas de convolución y con menos parámetros en las capas densas. Los resultados de este testeo se encuentran en la tabla.6. Esta vez, ambos modelos tienen las operaciones anteriormente descritas salvo el *batch normalization*.

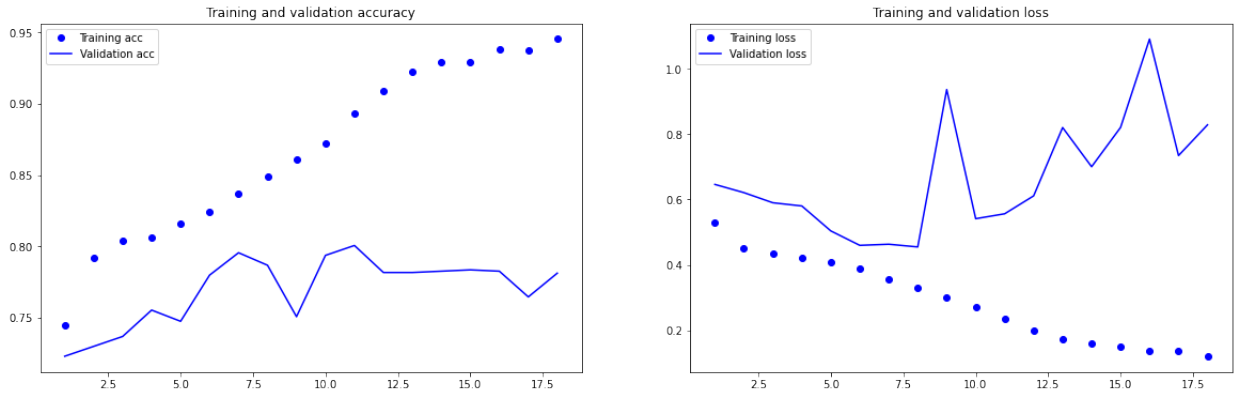



Figura 22: Histórico de la precisión y la perdida de los conjuntos de imágenes de entrenamiento (puntos) y validación (líneas). La arquitectura incluye *batch normalization*

| Arquitectura | Training accuracy | Validation accuracy | Training loss | Validation loss | AUC |
|---|---|---|---|---|---|
| 2Conv | 0.7967 | 0.8066 | 0.4358 | 0.4179 | 0.864 |
| 3Conv | 0.7904 | 0.8052 | 0.4534 | 0.4155 | 0.864 |

Tabla 6: Resultados obtenidos con dropout y data augmentation para dos arquitecturas diferentes: una más compleja con tres capas de convolución (3Conv) y otra con dos capas de convolución (2Conv).

De las dos arquitecturas descritas, la simplificada ofrece resultados ligeramente mejores. Como se puede ver en la figura.23, ahora la curva de validación se ajusta mejor a los resultados en entrenamiento. Sin embargo, hace un efecto de zigzag que indica que todavía tiene overfiting.

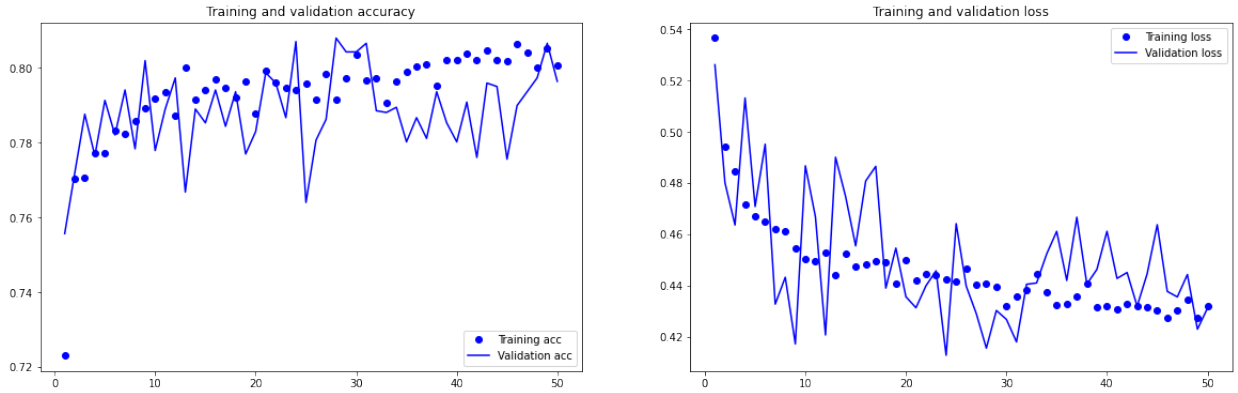



Figura 23: Histórico de la precisión y la perdida de los conjuntos de imágenes de entrenamiento (puntos) y validación (líneas) del modelo simplificado con algunas operaciones de regularización.

Finalmente se probó la arquitectura simplificada con imágenes dimensionadas a la mitad para comprobar que estas no pierden tanta información valiosa para la red al usar escalados más pequeños. Los resultados obtenidos no mejoran lo que ya se había conseguido, ver tabla.7

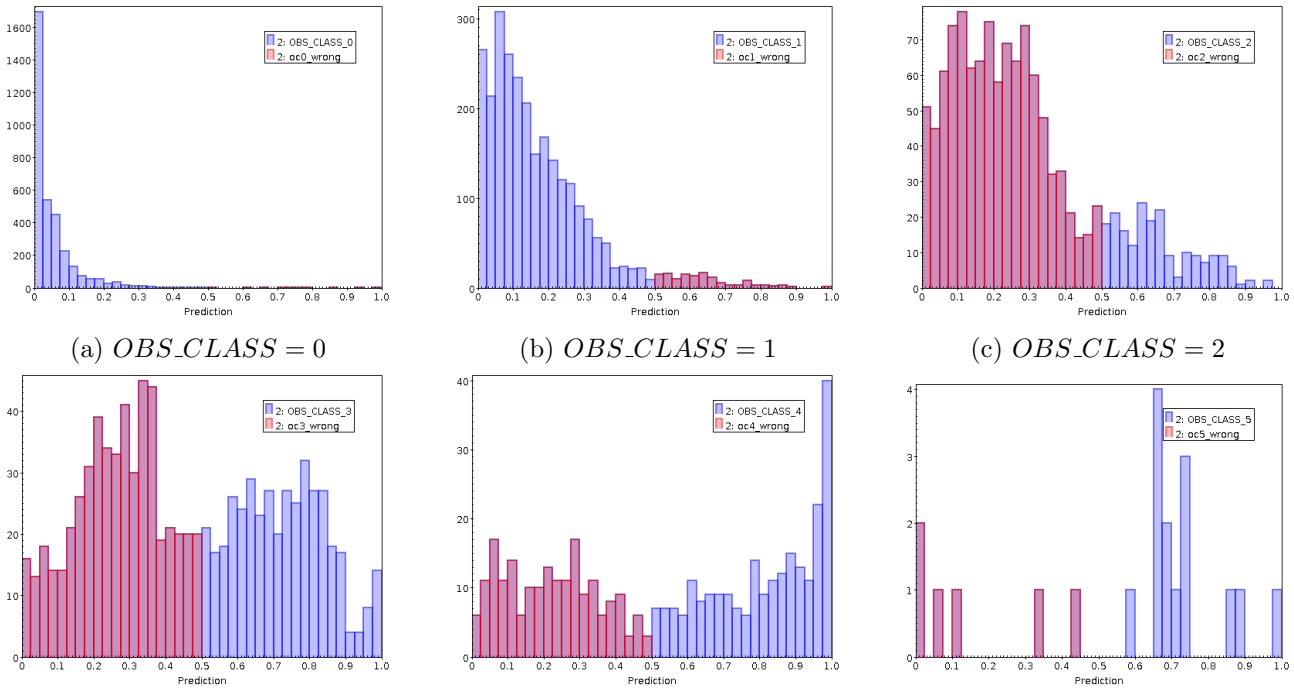
| Training accuracy | Validation accuracy | Training loss | Validation loss | AUC |
|---|---|---|---|---|
| 0.7942 | 0.8070 | 0.4425 | 0.4127 | 0.864 |

Tabla 7: Métricas obtenidas al usar imágenes menos rescaladas en el modelo simplificado y con técnicas de regularización

A pesar de las distintas técnicas empleadas para reducir el sobreajuste e intentar mejorar el desempeño del clasificador, este no consigue ofrecer precisiones más allá de 0.80 para el set de

validación. Por tanto, se indagó en la distribución de predicciones que hace el modelo, con el objetivo de ver en que imágenes falla más el algoritmo.

Para ello se representan las predicciones en histogramas, ver figura.24. El umbral que define si el algoritmo predice que una imagen es buena o mala se ha considerado en 0.5. Es decir, si la predicción de una muestra se encuentra en valores próximos al 0, es porque predice que la imagen es buena con bastante seguridad, y si los valores se encuentra cerca de 1, el modelo pensará que la imagen es mala. Por ello conviene revisar algunos ejemplos de falsos positivos y falsos negativos para ver en que está confundiéndose.




(a) $OBS\_CLASS = 0$ (b) $OBS\_CLASS = 1$ (c) $OBS\_CLASS = 2$

(d) $OBS\_CLASS = 3$ (e) $OBS\_CLASS = 4$ (f) $OBS\_CLASS = 5$

Figura 24: Predicciones obtenidas al aplicar el modelo 'definitivo' al set de training. Se muestran los histogramas para cada OBS_CLASS. Se utilizó 0.5 como el umbral para diferenciar si la predicción es buena o mala. En azul las predicciones acertadas y en rojo las erróneas.

Se observa claramente como la red es capaz de hacer buenas predicciones para las clases consideradas como buenas (OBS_CLASS=0,1) ya que solo realiza 15 predicciones erróneas para la clase 0 (0.45 %) y 131 para la clase 1 (4.88 %). En cambio, el modelo tiene dificultades para clasificar las imágenes consideradas como malas (OBS_CLASS). En el caso de las imágenes de clase observacional 2, 1021 imagenes fueron mal predichas (83.69), como si el modelo tuviese dificultadas para discernir las imágenes de esta clase con las de las clases 0 y 1. Por otra parte, las imágenes de las clases 3 y 4 fueron mal predichas en la mitad de los casos aproximadamente (55.45 % y 45.64 % respectivamente). En cuanto a la clase 5, aunque se dispone de pocas muestras, 6 de las 20 imágenes fueron confundidas con imágenes de buena calidad, cuando las imágenes de esta clase son las que más defectos contienen. Ver figura. 24 y tabla. 8

A continuación se analizaran algunas de las peores predicciones para intentar comprender estos falsos positivos y falsos negativos. Ver figuras 25 y 26.

Por ejemplo, las imágenes a de la figura.25 y d e i de la figura.26 son similares visualmente: se perciben los chips MOS, sin tomar lecturas en el central y con gran luminosidad en el centro. Estas imágenes fueron predichas como malas, pero la figura.25.a en concreto pertenece a la clase OBS_CLASS=1, por lo que se trata de un falso negativo.

Otra imagen visualmente similar a estas es la figura.25.e, con OBS_CLASS=1. En este caso

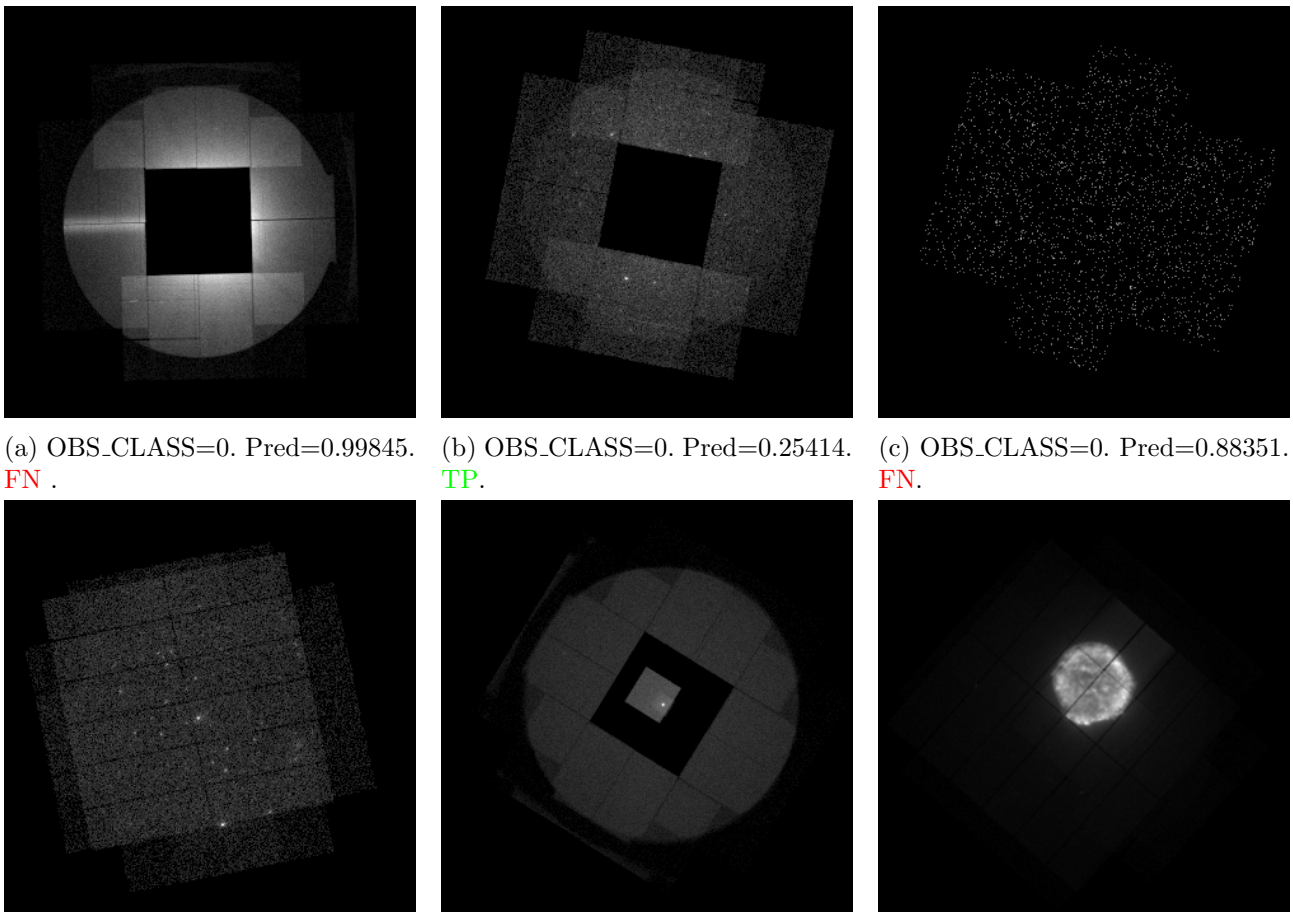
| OBS_CLASS | Imágenes | Predicciones erróneas | Ratio % |
|---|---|---|---|
| 0 | 3367 | 15 | 0.45 |
| 1 | 2684 | 131 | 4.88 |
| 2 | 1220 | 1021 | 83.69 |
| 3 | 927 | 519 | 55.98 |
| 4 | 425 | 192 | 45.64 |
| 5 | 20 | 6 | 30.00 |

Tabla 8: Número de predicciones erróneas por cada OBS_CLASS, así como el porcentaje que supone del total de imágenes de cada categoría. Se utilizó el set de entrenamiento.

se perciben los chips MOS y el pn en modo ventana parcial con una fuente en el centro. Nuevamente, se produce un falso negativo, habiendo sido considerada como una imagen de mala calidad.

Las imágenes d de la figura.25 y a, c, e y g de la figura.26 también comparten características visuales similares, ya que en todas se perciben los chips pn y varias fuentes puntuales. Todas estas imagenes han sido consideradas como buenas por el modelo. Para la figura.25.d., esto resulta en un verdadero postivo, pero para el resto resulta en falsos positivos. Por una parte, la figura.26.a pertenece a la OBS_CLASS=2, con lo que el error en la predicción no es tan grande, pero en el caso de las imágenes c y e de la figura.26, se aprecian regiones malas que la red no tuvo en cuenta. Por ejemplo, en la primera se observan arcos de reflexion en la esquina superior izquierda, además de una zona extensa con emisión difusa y en la segunda se aprecian en gran parte de la imagen arcos de reflexión.

Cabe mencionar que estas imágenes se muestran como las recibe la red. Se puede ir una por una modificando parámetros visuales de forma que haya una diferenciación más clara.



(a) OBS_CLASS=0. Pred=0.99845. FN .

(b) OBS_CLASS=0. Pred=0.25414. TP.

(c) OBS_CLASS=0. Pred=0.88351. FN.

(d) OBS_CLASS=1, Pred=0.00068. TP.

(e) OBS_CLASS=1. Pred=0.82074. FN

(f) OBS_CLASS=1. Predicción=0.97591. FN.

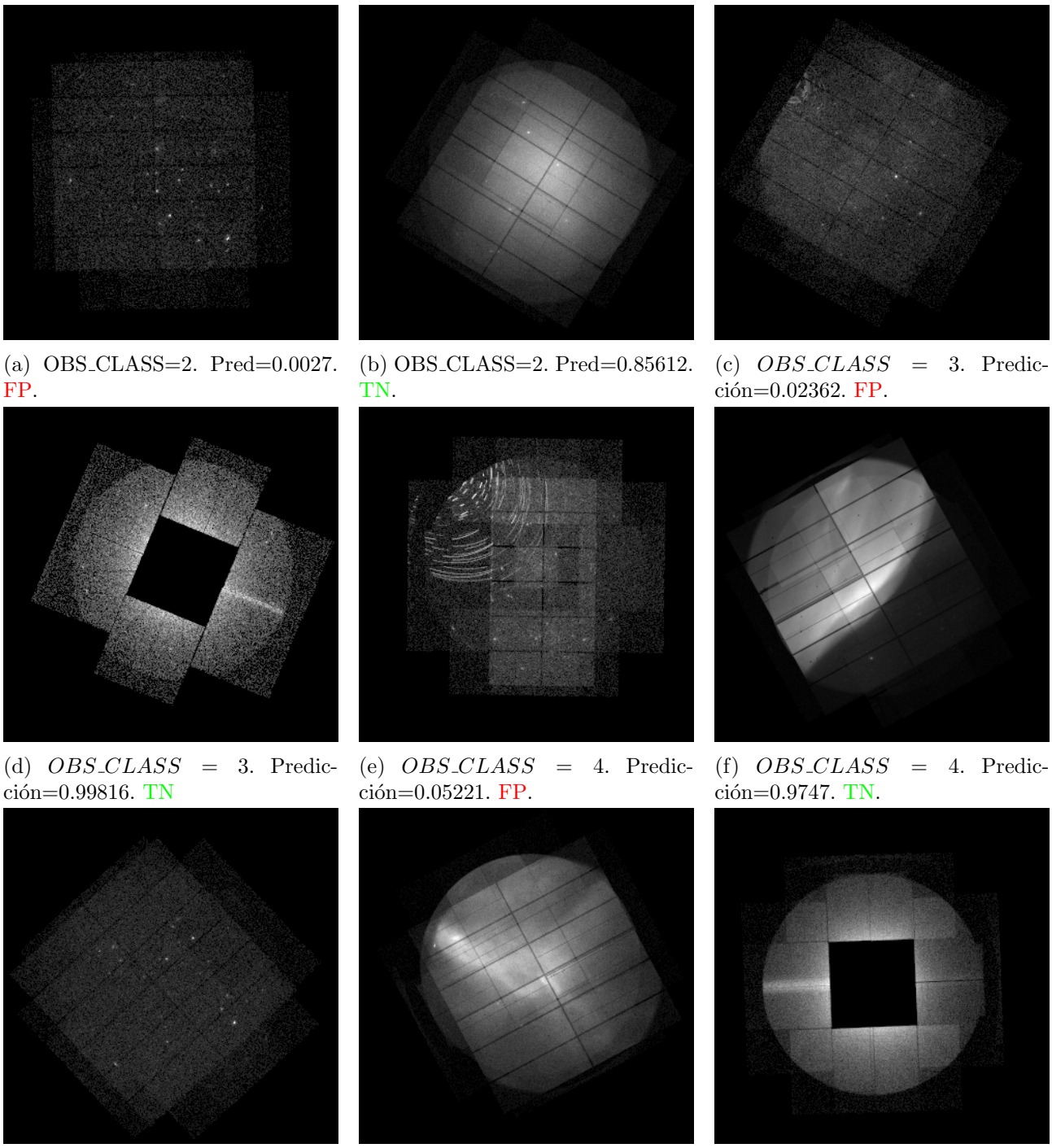
Figura 25: Algunas imágenes *buenas* con rebinning 2x2, normalización sqrt e intervalo MinMax con las que entrenó el último modelo. Cada imagen tiene su clase observacional asociada (OBS_CLASS, la predicción (Pred) y si es un verdadero positivo (TP) o un falso negativo (FN).

# 5. Conclusiones

En esta última sección de la memoria se dan las conclusiones obtenidas en la realización del trabajo, así como los conocimientos adquiridos. También se exponen posibles mejoras del modelo empleado para la clasificación automáticas de las observaciones.

La finalidad de este proyecto es implementar técnicas de clasificación de imágenes para el caso de fuentes astronómicas de rayos X, y en particular técnicas de aprendizaje automático, de forma que la tarea de realizar esta agrupación de forma visual por parte de expertos conlleve menos tiempo. El modelo de *machine learning* empleado ha resultado exitoso en la clasificación de imágenes de buenas, proporcionan predicciones correctas en la mayoría de los casos. En el caso de las imágenes consideradas como malas, el modelo tiene dificultas para clasificarlas correctamente como se expuso con algunos ejemplos en la parte de resultados. Una de las razones de esta discrepancia es la complicada tarea para reconocer visualmente, a la vista de una persona no experta, si ciertas imágenes se podrían considerar de una categoría u otra.

Cómo se mencionó en la sección sobre inteligencia artificial, si los modelos de aprendizaje automático buscan aprender los patrones, que a nosotros como humanos nos permiten llegar a clasificaciones lógicas, y nosotros mismos somos incapaces de encontrar esas características, es probable que la red neuronal tampoco llegue a dar una buena clasificación de algunas imágenes.



(a) OBS_CLASS=2. Pred=0.0027. FP.

(b) OBS_CLASS=2. Pred=0.85612. TN.

(c) $OBS\_CLASS = 3$. Predicción=0.02362. FP.

(d) $OBS\_CLASS = 3$. Predicción=0.99816. TN

(e) $OBS\_CLASS = 4$. Predicción=0.05221. FP.

(f) $OBS\_CLASS = 4$. Predicción=0.9747. TN.

(g) $OBS\_CLASS = 5$, Predicción=0.00298. FP

(h) $OBS\_CLASS = 5$. Predicción=0.88351. TN

(i) $OBS\_CLASS = 5$. Predicción=0.99435. TN

Figura 26: Algunas imágenes *buenas* con rebinning 2x2, normalización sqrt e intervalo MinMax con las que entreno el modelo final. Cada imagen tiene su clase observacional asociada y la predición que dio el modelo para esta, tomando 0 como una muy buena predicción y 1 como muy mala.

Se podría ayudar a mejorar la clasificación de las observaciones mediante redes neuronales convencionales si en vez de entrenar el modelo con las imágenes etiquetadas por categorías, se las enseñase a razonar como los expertos: dándole observaciones con las regiones problemáticas marcadas.

También se podría realizar otra categorización del conjunto de datos de entrada, como añadir la clase observacional 2 al set de imágenes buenos, ya que el modelo predice muchas de las imágenes de esta clase como buenas. O directamente, sería interesante utilizar redes neuronales convolucionales para desarrollar un algoritmo que haga una clasificación por clase observacional.

La realización del presente proyecto me ha proporcionado nuevos y profundos conocimientos en el campo del aprendizaje automático, así como al desarrollo de habilidades en el entorno de programación *Python* y las diferentes librerías que posee orientadas a la creación de modelos de *machine learning* como *Keras* y de análisis científico en general como *Scipy*, *Pandas* [26] o *Astropy* [27]. A manejar un conjunto de datos grande, con las limitaciones que ello conlleva en cuanto a gestión de recursos informáticos. También se adquirieron conocimientos en software especializado en análisis de imágenes astronómicas como *SAOimageDS9* [28] y de manipulación tablas como *TOPCAT* [29].

# Referencias


[1] A. Martinazzo, M. Espadoto, and N. S. T. Hirata, "Self-supervised learning for astronomical image classification," 2020.

[2] "Xmm-newton - a concise overview." https://xmm-tools.cosmos.esa.int/external/xmm_user_support/documentation/uhb/overview.html. Accessed: 2021-02-19.

[3] "Xmm-newton sciencex archive (xsa)." https://www.cosmos.esa.int/web/xmm-newton/xsa.

[4] "European photon imaging camera (epic)." https://xmm-tools.cosmos.esa.int/external/xmm_user_support/documentation/uhb/epic.html. Accessed: 2021-02-19.

[5] "Epic mos chip geometry." https://xmm-tools.cosmos.esa.int/external/xmm_user_support/documentation/uhb/moschipgeom.html. Accessed: 2021-02-19.

[6] "Epic pn chip geometry." https://xmm-tools.cosmos.esa.int/external/xmm_user_support/documentation/uhb/pnchipgeom.html. Accessed: 2021-02-19.

[7] "Science modes of the epic cameras." https://xmm-tools.cosmos.esa.int/external/xmm_user_support/documentation/uhb/epicmode.html. Accessed: 2021-02-19.

[8] "Xmm-newton orbit." https://xmm-tools.cosmos.esa.int/external/xmm_user_support/documentation/uhb/orbit.html. Accessed: 2021-02-19.

[9] "Radiation belts." https://xmm-tools.cosmos.esa.int/external/xmm_user_support/documentation/uhb/radbelts.html. Accessed: 2021-02-19.

[10] "Celestial constraints." https://xmm-tools.cosmos.esa.int/external/xmm_user_support/documentation/uhb/celest.html. Accessed: 2021-02-19.

[11] N. A. Webb, M. Coriat, I. Traulsen, J. Ballet, C. Motch, F. J. Carrera, F. Koliopanos, J. Authier, I. de la Calle, M. T. Ceballos, E. Colomo, D. Chuard, M. Freyberg, T. Garcia, M. Kolehmainen, G. Lamer, D. Lin, P. Maggi, L. Michel, C. G. Page, M. J. Page, J. V. Perea-Calderon, F. X. Pineau, P. Rodriguez, S. R. Rosen, M. Santos Lleo, R. D. Saxton, A. Schwope, L.ás@, M. G. Watson, and A. Zakardjian, "The XMM-Newton serendipitous survey. IX. The fourth XMM-Newton serendipitous source catalogue," , vol. 641, p. A136, Sept. 2020.

[12] "The xmm-newton serendipitous source catalogue: 4xmm-dr9." http://xmmssc.irap.omp.eu/Catalogue/4XMM-DR9/4XMM-DR9$_{C}atalogue_{U}ser_{G}uide.htmlDiscOverview. Accessed$ : 2021 − 02 − 25.

[13] "Measures of scale." https://www.itl.nist.gov/div898/handbook/eda/section3/eda356.htm. Accessed: 2021-02-19.

[14] "Skewness." https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.skew.html. Accessed: 2021-02-19.

[15] "Asimetria (estadística)." https://ca.wikipedia.org/wiki/Asimetria_(estad\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{\OT1\i\global\mathchardef\accent@spacefactor\spacefactor}\let\begingroup\endgroup\relax\let\ignorespaces\relax\accent19\OT1\i\egroup\spacefactor\accent@spacefactor\futurelet\@let@token\protect\penalty\@M\hskip\z@skipstica). Accessed: 2021-02-19.

[16] "Diferencias en la estimación del coeficiente de curtosis en diferentes softwares estadísticos.." https://revistas.tec.ac.cr/index.php/eagronegocios/article/view/4456. Accessed: 2021-02-21.

[17] A. M. TURING, "I.—COMPUTING MACHINERY AND INTELLIGENCE," *Mind*, vol. LIX, pp. 433–460, 10 1950.

[18] B. Raphael, "Sir: A computer program for semantic information retrieval," tech. rep., USA, 1964.

[19] "Supervised and unsupervised machine learning algorithms." https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/. Accessed: 2021-02-19.

[20] "Commonly used activation functions." https://cs231n.github.io/neural-networks-1/. Accessed: 2021-02-19.

[21] "Redes neuronales." https://ml4a.github.io/ml4a/es/neural_networks/. Accessed: 2021-02-19.

[22] "Max-pooling / pooling." https://computersciencewiki.org/index.php/Max-pooling_/_Pooling. Accessed: 2021-02-19.

[23] "Flattening." https://medium.com/@PK_KwanG/cnn-step-2-flattening-50ee0af42e3e. Accessed: 2021-02-21.

[24] "Clasificación: Roc y auc." https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=es$_4$19.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[26] "Pandas." https://pandas.pydata.org/pandas-docs/stable/index.html.

[27] "The astropy project." https://www.astropy.org/.

[28] "Saoimageds9: An image display and visualization tool for astronomical data." https://sites.google.com/cfa.harvard.edu/saoimageds9.

[29] "Topcat: Tool for operations on catalogues and tables." http://www.star.bris.ac.uk/ mbt/topcat/.

# A. Configurando modelo

Para diseñar los modelos empleados, se creó una función en *python* que pide varios parámetros:

```
def create_3conv_model(dataset_dir, x_train, y_train, x_val, y_val, filter_dimension,
                       dropout, dropout_val, epochs, batch_size, patience,
                       data_augmentation, datagen, batch_normalization, verbose, log_path,
                       log_name, model_path, model_name):

    import pickle, os, sys
    import numpy as np
    import pandas as pd
    from keras import layers, models, optimizers
    from keras.preprocessing.image import ImageDataGenerator
    from keras.models import load_model
    from keras.callbacks import CSVLogger, EarlyStopping

    # Loads datasets
    x_train = pickle.load(open(os.path.join(dataset_dir, x_train), 'rb'))
    y_train = pickle.load(open(os.path.join(dataset_dir, y_train), 'rb'))
    x_val = pickle.load(open(os.path.join(dataset_dir, x_val), 'rb'))
    y_val = pickle.load(open(os.path.join(dataset_dir, y_val), 'rb'))

    #Creates savepaths
    model_save_path = os.path.join(model_path, model_name)
    log_save_path = os.path.join(log_path, log_name)

    # Validates if the file already exists
    if os.path.exists(model_save_path):

        sys.exit('El archivo '+model_name+' ya existe en la carpeta '+model_path)

    if os.path.exists(log_save_path):

        sys.exit('El archivo '+log_name+' ya existe en la carpeta '+log_path)

    model = models.Sequential()
    # Capas convolucionales
    model.add(layers.Conv2D(16, filter_dimension, activation='relu',
                            input_shape=(x_train.shape[1:]), padding='SAME'))
    model.add(layers.MaxPooling2D((2, 2)))

    if batch_normalization==True:

        model.add(BatchNormalization())

    model.add(layers.Conv2D(32, filter_dimension, activation='relu', padding = 'same'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, filter_dimension, activation='relu', padding = 'same'))
    model.add(layers.MaxPooling2D((2, 2)))
    # Clasificador
    model.add(layers.Flatten())
```

```
    model.add(layers.Dense(128, activation='relu'))

    if dropout==True:
        model.add(layers.Dropout(dropout_val))

    model.add(layers.Dense(16, activation='relu'))

    model.add(layers.Dense(1, activation='sigmoid'))

    #Compile the model
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

    #Set callbacks
    es = EarlyStopping(monitor='val_loss', mode='min', patience=patience)
    csv_logger = CSVLogger(log_save_path, separator=',', append=False)

    if data_augmentation==True:

        datagen.fit(x_train)
        #Fits the model on batches with real-time data augmentation:
        hist = model.fit(datagen.flow(x_train, y_train[:,0], batch_size=batch_size),
                         validation_data=(x_val, y_val[:,0]), epochs=epochs, verbose=verbose,
                         callbacks=[csv_logger, es])

    else:
        #Train/fit the model
        hist = model.fit(x_train, y_train[:,0], batch_size=batch_size, epochs=epochs,
                          verbose=verbose, validation_data=(x_val, y_val[:,0]),
                          callbacks=[csv_logger, es])

    #Saves model
    model.save(model_save_path)

    return print('Model '+model_name+' trained and saved')
```

Parámetros:

- dataset_dir, x_train, y_train, x_val, y_val: Piden el nombre de los sets de datos de entrenamiento y validación, sus categorías y el directorio donde se encuentran.

- filter_dimension: El tamaño del filtro que queremos asignar.

- dropout, dropout_val: Indicar mediante un booleano si se quiere utilizar *dropout* y el valor del mismo

- epochs, batch_size, patience: Número de épocas, tamaño de *batch* y la paciencia.

- data_augmentation, datagen: Nos pide un booleano para saber si queremos utilizar *data augmentation* y como queremos modificar nuestras imágenes introduciendo un ImageDataGenerator() al que podemos incluir rotaciones, volteos...

- batch_normalization: Booleano para indicar si se quiere añadir *batch normalization*.

- verbose: sirve para indicar cuanta información queremos ir viendo mientras el modelo entrena. 0 no muestra nada, 1 muestra una barra de progresión por época y 2 solo menciona la época en la que está entrenando.

- log_path, log_name: Directorio donde se guardará el histórico y el nombre que queramos darle.

- model_path, model_name: Directorio donde se guardará el modelo y el nombre que queramos darle.

Ejemplo de utilización:

```
create_3conv_model(dataset_dir='/home/juan/ED/Datasets/', x_train='x_train_mm_sqrt.pickle',
            y_train='y_train_bin.pickle', x_val='x_val_mm_sqrt.pickle', y_val='y_val_bin.pickle',
            filter_dimension=(3,3), dropout=True, dropout_val=0.5, epochs=50, batch_size=64,
            patience=50, data_augmentation=True,
            datagen=ImageDataGenerator(rotation_range=180, vertical_flip=True, horizontal_flip=True),
            batch_normalization=False, verbose=1,
            log_path='/home/juan/HistN/', log_name='hist_3x3_aug_all_50e_mm_sqrt.log',
            model_path='/home/juan/ModelsN/', model_name='model_3x3_aug_all_50e_mm_sqrt.h5')
```

La función model.summary() muestra la estructura de la red y en número de parámetros.

```
    Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 162, 162, 16)      160
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 81, 81, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 81, 81, 32)        4640
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 40, 40, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 40, 40, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 20, 20, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 25600)             0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               3276928
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 16)                2064
_________________________________________________________________
```

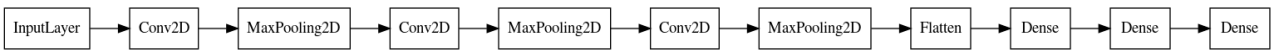
```
dense_3 (Dense)              (None, 1)                 17
=================================================================
Total params: 3,302,305
Trainable params: 3,302,305
Non-trainable params: 0
```

La CNN comienza a entrenar y en pantalla se va mostrando el progreso por época:

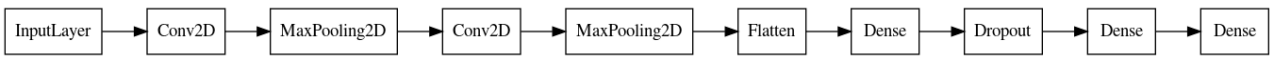
```
Epoch 1/50
136/136 [==============================] - 63s 466ms/step - loss: 0.5376 -
accuracy: 0.7231 - val_loss: 0.5262 - val_accuracy: 0.7557
Epoch 2/50
136/136 [==============================] - 66s 484ms/step - loss: 0.4964 -
accuracy: 0.7705 - val_loss: 0.4801 - val_accuracy: 0.7719
Epoch 3/50
136/136 [==============================] - 60s 439ms/step - loss: 0.4833 -
accuracy: 0.7707 - val_loss: 0.4636 - val_accuracy: 0.7876
Epoch 4/50
136/136 [==============================] - 60s 440ms/step - loss: 0.4695 -
accuracy: 0.7772 - val_loss: 0.5132 - val_accuracy: 0.7765
...
Epoch 48/50
136/136 [==============================] - 58s 426ms/step - loss: 0.4338 -
accuracy: 0.8002 - val_loss: 0.4443 - val_accuracy: 0.7973
Epoch 49/50
136/136 [==============================] - 58s 424ms/step - loss: 0.4247 -
accuracy: 0.8053 - val_loss: 0.4229 - val_accuracy: 0.8066
Epoch 50/50
136/136 [==============================] - 58s 425ms/step - loss: 0.4311 -
accuracy: 0.8008 - val_loss: 0.4311 - val_accuracy: 0.7964
Model model_3x3_aug_all_50e_mm_sqrt.h5 trained and saved
```

Una vez terminado el entrenamiento, se guardan el modelo entrenado y un archivo con el histórico del *accuracy* y la *loss* por cada época.

También es posible obtener una representación esquemática de la estructura del modelo mediante la función *plot_model* de *keras*.




(a) Modelo con tres capas convolucionales y de *MaxPooling*




(b) Modelo simplificado con dos capas convolucionales y de *MaxPooling*

Figura 27: Representación esquemática de la arquitectura de los modelos utilizados