Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

Detección de incendios en áreas forestales usando algoritmos de deep learning

Alumno: Irene Moreno Rubio

Tutor: Francisco Charte Ojeda

Dpto: Informática

UNIVERSIDAD DE JAÉN

D./Dª Francisco Charte Ojeda, tutor(es) del Trabajo Fin de Grado titulado: **Detección de incendios en áreas forestales usando algoritmos de deep learning**, que presenta Irene Moreno Rubio, autoriza(n) su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, julio de 2021

El estudiante

El tutor

CHARTE OJEDA FRANCISCO - 25996290B

Firmado digitalmente por CHARTE OJEDA FRANCISCO - 25996290B
Fecha: 2021.07.19 19:17:28 +02'00'

Irene Moreno Rubio

Francisco Charte Ojeda

Agradecimientos

A Francisco Charte Ojeda, por su labor como tutor de este trabajo. Por sus sabios consejos y la gran atención que me ha prestado durante la elaboración del mismo.

A Irene Padilla, Javier Martínez, Sergio Bonachera y Paúl Andrade, porque sin ellos no habría llegado a conseguir ninguno de los objetivos de estos cuatro años.

A Azahara y Nazaret, que aun sin saber cómo ayudarme con la elaboración de este trabajo, han estado para darme apoyo siempre que las he necesitado.

Pero sin duda, mi mayor agradecimiento va hacia mi familia. Ellos son mi mayor apoyo en todo lo que hago y lo seguirán siendo durante toda mi vida.

Tabla de contenidos

Lista de figuras

Lista de tablas

Capítulo 1

INTRODUCCIÓN

En una primera toma de contacto con el tema a tratar en este trabajo se hablará sobre la importancia de evitar la deforestación y los problemas irreversibles que causa.

Concretamente se profundizará en cómo afectan los incendios forestales a este problema y cuáles son los principales motivos de la aparición de los mismos. Tras explicar esto, se remarcará la importancia de la detección precoz y lo útil que podría resultar hallar un sistema automático para que ayude con esto.

1.1. Deforestación

La deforestación es el proceso por el cual un área forestal deja de serlo para convertirse en otro tipo de terreno utilizado para fines como la construcción de zonas urbanas.

Tal y como se explica en Chakravarty et al. [2012], la importancia de evitarla nace en los efectos que esta produce. Principalmente la pérdida de biodiversidad y el agravamiento del efecto invernadero.

El 30 % del terreno del planeta está cubierto por áreas forestales, esto supone alrededor de 3900 millones de hectáreas. Teniendo en cuenta que se estima que la cobertura original era de 6000 millones de hectáreas, se han reducido estas áreas en un 35 %, lo que a largo plazo puede causar problemas.

Como se muestra en la Figura 1.1, entre los años 1990 y 2015, en África, América del Sur e Indonesia, se encuentran las zonas que más pérdida de áreas forestales

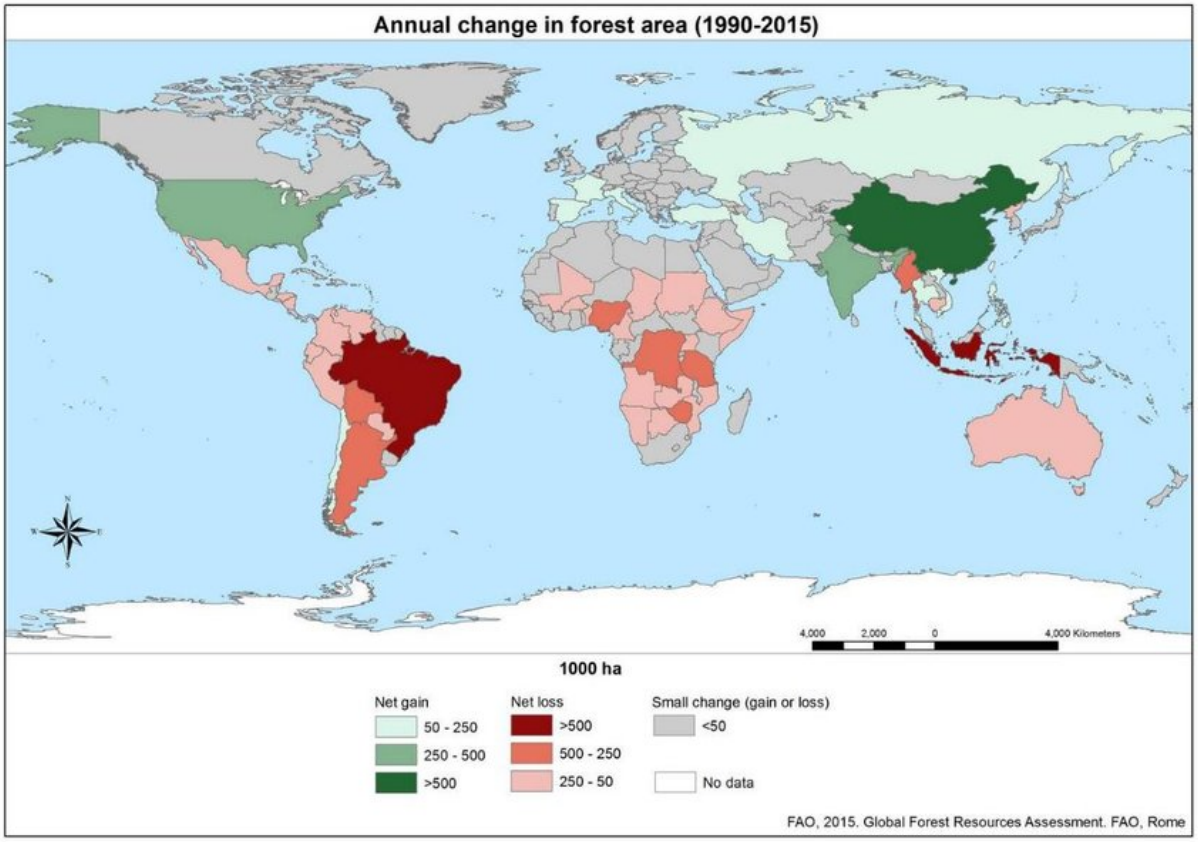

Figura 1.1: Cambio anual en las áreas forestales de todo el mundo. Di Lallo [2017]

sufrieron, superando las 500 000 hectáreas. Por otra parte, también se aprecian zonas en las que hay una ganancia como Estados Unidos, China e India, pero esto no significa que la situación global se equilibre ni mejore.

1.2. Incendios forestales

Los incendios forestales son una de las causas de la deforestación. Estos acaban con la vida de una gran cantidad de árboles cada año y extinguirlos a tiempo puede marcar la diferencia entre un incendio leve o una catástrofe.

Conviene analizar cuáles son las causas más comunes de los incendios forestales y también los efectos que causan a corto y largo plazo.

1.2.1. Causas de los incendios forestales

Un estudio realizado entre los años 1996 y 2010 en Portugal, Lourenço [2013], muestra cuáles fueron las causas de los incendios forestales producidos en este periodo de tiempo.

- **Actos humanos**: el uso irresponsable de pirotecnia o incluso algunos juegos de niños fueron los causantes del 19.4 % de los incendios. A este grupo también pertenecen el uso de hogueras y el vandalismo.
- **Uso negligente del fuego**: el 19 % de los incendios se originaron debido a la quema de basura o áreas de vegetación por agricultores.
- **Reinicio**: un fuego extinguido puede llegar a volver a iniciarse. Esta causa fue responsable del 6.2 % del total.
- **Accidentes**: el tendido eléctrico o incluso algún tipo de maquinaria instalada cerca de áreas forestales puede desatar un incendio si se produce algún roce entre estos y las ramas de los árboles. En el caso estudiado, esto provocó el 3 % de los incendios ocurridos.
- **Natural**: el impacto de un rayo con una rama o un tronco de árbol es un evento natural que inevitablemente puede causar un incendio. Este tipo de situaciones en particular originaron solamente el 0.8 % de los incendios.

Sin embargo, el 50.7 % de los incendios se originaron por causas desconocidas, lo que hace aún más complicado evitarlos, ya que no está claro a qué o quién se puede culpar por ello.

1.2.2. Efectos de los incendios forestales

Las consecuencias más graves de un incendio forestal, recaen sobre la tierra. En Verma and Jayakumar [2012] se especifica que el impacto provoca cambios físicos, químicos y biológicos en ella.

- **Cambios físicos**: además de afectar en la resistencia al agua, que resulta en una menor infiltración y una mayor erosión de la capa superior del suelo, causa cambios en el color, textura e incluso el pH de la tierra.

- **Cambios químicos**: sin duda son los más importantes, ya que afectan a la productividad. El decremento de la cantidad de nitrógeno hace que sea menos probable que vuelvan a crecer arbustos y otro tipo de plantas.

- **Cambios biológicos**: las altas temperaturas hacen que desaparezcan las especies que habitan en la tierra, tales como microorganismos e invertebrados.

Figura 1.2: Zonas de tierra afectadas por el incendio del Amazonas en 2019. El_Pais [2019]

La Figura 1.2 muestra el cambio físico que puede sufrir una zona cuando se produce un incendio. En concreto, el fuego que afectó a la selva amazónica el pasado año 2019.

Estos efectos concretos sobre la tierra son los que eventualmente causan la deforestación. Pero, definitivamente estos no son los únicos problemas que provoca el fuego, ya sea controlado o no.

1.3. Motivación

Una vez definidos los efectos tan graves que pueden originar los incendios forestales, es evidente que existen dos opciones: evitarlos o, en caso de ser inevitables, extinguirlos con la mayor rapidez.

La motivación de realizar este trabajo viene dada por la necesidad de encontrar un método fiable que pueda llegar a agilizar el proceso de detección de incendios y aviso a los servicios de extinción de los mismos.

La automatización no solamente puede resultar en un ahorro en recursos humanos, sino también en una mayor eficiencia, ya que se podrían procesar muchas imágenes en un tiempo reducido, haciendo así posible abarcar áreas mucho más extensas.

Es obvio que estos sistemas seguirían necesitando supervisión humana, ya que la no detección de un incendio activo puede ser desfavorable y originar el descontrol del fuego.

1.4. Automatización

Para llevar a cabo una automatización de todo el proceso de detección y aviso de un incendio forestal sería necesario construir sistemas de apoyo a la vigilancia aprovechando todo lo que la tecnología puede ofrecer en la actualidad.

Las cámaras que disparan automáticamente cada vez que ocurre cierto suceso o que pasa un intervalo de tiempo pueden ser parte de este tipo de sistemas. A esto se podrían unir las redes de comunicaciones necesarias para permitir el envío de esa información captada por las cámaras para después ser procesadas en un centro de control en el que se ejecuten algoritmos concretos como los que se van a estudiar en este trabajo. Incluso cabe la posibilidad de que sean las propias cámaras las que cuenten con una unidad de procesamiento que permita la detección de un potencial incendio y lance alertas.

En conclusión, los sistemas que se plantean se apoyan finalmente en modelos de visión por computador y aprendizaje automático, que es precisamente el campo sobre el que se va a profundizar durante la realización de este trabajo.

1.5. Capítulos de la memoria

Para que la lectura de esta memoria se haga más amena para todo aquel que desee leerla, es imprescindible especificar cuáles son los capítulos que la conforman y cuál es el contenido de cada uno de ellos de forma breve.

Capítulo 1. Introducción

Este mismo capítulo, en el que se habla de forma general del problema que se va a tratar de resolver en este trabajo y se especifican las razones que han llevado a que sea algo necesario.

Capítulo 2. Antecedentes

Estudio previo de todos los aspectos teóricos necesarios para abordar el tema del trabajo. Este capítulo también incluye cuáles son los métodos empleados hasta el momento para la solución al problema especificado y algunas posibles técnicas más.

Capítulo 3. Objetivos

Explica el objetivo principal definido desde el inicio de este trabajo así como los objetivos específicos que llevan a conseguirlo.

Capítulo 4. Materiales y Métodos

Como bien indica el nombre del capítulo, en él se detallan todos los materiales y métodos que son necesarios para la realización del trabajo. Esto se refiere a los datos, procedimientos, hardware y software empleados en todo momento.

Capítulo 5. Resultados

En este capítulo se muestra cuál ha sido la experimentación realizada utilizando los materiales y métodos elegidos y finalmente cuáles han sido los resultados ofrecidos por los sistemas obtenidos. Se presentan comparativas para así poder tomar una decisión en cuanto al modelo que presenta un mejor funcionamiento.

Capítulo 6. Conclusiones

Como cierre de esta memoria, se especificará si se han alcanzado los objetivos propuestos en un inicio y el grado de satisfacción con los resultados finalmente obtenidos.

Capítulo 2

ANTECEDENTES

Para poder trabajar con aprendizaje profundo es necesario comenzar estudiando el campo de la Ciencia de datos de forma general, profundizando poco a poco en las técnicas concretas que se van a utilizar en este trabajo. En este capítulo se pretende explicar todo lo necesario acerca de los fundamentos teóricos que han sido necesarios para buscar una solución al problema propuesto.

Por otra parte, se hablará sobre cómo se ha abordado la detección de incendios hasta el momento, y algunos procedimientos que se tuvieron en cuenta antes de realizar los experimentos, pero que finalmente no se utilizaron.

2.1. Técnicas de visión por computador

La visión por computador o *computer vision*, tal y como se introduce en Brownlee [2019], se trata del desarrollo de técnicas que permitan a los ordenadores "ver" y entender el contenido de imágenes digitales como videos y fotografías.

El sistema visual de los humanos es mucho más complejo que, por ejemplo, el habla. Los expertos en percepción han estudiado su funcionamiento durante décadas sin llegar a tener una idea exacta, por lo que esto hace mucho más difícil poder llegar a imitarlo computacionalmente. De esto se habla más detalladamente en Szeliski [2011].

En la actualidad, la visión computacional se aplica en una amplia variedad de problemas reales, entre ellos:

- **Reconocimiento de caracteres**: con el objetivo de leer cartas o incluso reconocer números de matrícula.
- **Construcción de modelos 3D**: desde imágenes aéreas.
- **Seguridad automovilística**: para la detección de obstáculos o viandantes.
- **Vigilancia**: monitorización de todo tipo de imágenes para aumentar la seguridad en distintas situaciones como el tráfico de autovías o posibles ahogos en piscinas.
- **Reconocimiento de huella y biometría**: para dar acceso automático a lugares o reconocer cadáveres.
- **Detección facial**: esto se puede ver en redes sociales como *Facebook*, en la que se reconoce inmediatamente a las personas que aparecen en la imagen para poder ser etiquetadas.

Existe una gran cantidad de técnicas de visión computacional. A continuación se van a abordar aquellas que se consideran relevantes para este trabajo.

2.1.1. Procesamiento de imágenes

El *image processing* se refiere a cualquier tipo de transformación que se realice sobre las imágenes para adaptarlas a las condiciones exigidas por técnicas que se vayan a aplicar *a posteriori*. Dentro de esta técnica se incluyen operaciones como corrección del color, reducción del ruido, cambio de brillo, etc.

Concretamente, dentro de esta técnica, la forma en la que se generan los píxeles de salida con respecto a los píxeles de entrada da lugar a una serie de subtécnicas, todas ellas explicadas en Szeliski [2011].

- ***Point operators***: en este tipo de operaciones el píxel resultado únicamente va a depender del píxel de entrada, sin tener en cuenta vecinos. En este grupo se pueden incluir el equilibrado del color, *matting* y *compositing* y la ecualización del histograma.
- **Filtro lineal**: cuando se aplica un filtro lineal, cada píxel de salida es resultado de una combinación lineal (suma con pesos) del píxel de entrada y sus vecinos. Dentro de este grupo se encuentran las convoluciones, que no son más que operaciones realizadas sobre los elementos que componen la matriz que representa

una imagen pasando un filtro de ciertas dimensiones sobre ellos. Esta operación tiene gran relevancia en este trabajo, por lo que será detallada con detenimiento más adelante.

- **Filtro no lineal**: en este caso, los píxeles salida son combinación del píxel original y sus vecinos, pero no de forma lineal. Estos filtros, entre el que se incluye el filtro de mediana o *median filter* y el bilateral, ayudan a reducir ciertos tipos de ruido como el "sal y pimienta".

Dentro de esta técnica, además, se encuentra el redimensionamiento de las imágenes, que pretende cambiar el número de píxeles a una cantidad menor o mayor. Para poder hacer esto de manera correcta, es esencial que se aplique un interpolador, es decir, que se defina qué función es la encargada de definir el valor del nuevo píxel que puede ser resultado de varios píxeles (reducción) o parte de uno concreto (aumento).

Interpolación de imágenes

A la hora de realizar un aumento o reducción de una imagen es necesario aplicar un tipo de interpolación. A continuación se muestran los interpoladores relevantes para la experimentación posterior.

- **Vecino más cercano**: este tipo de interpolación simplemente iguala el nuevo píxel a aquel que fuera más cercano en la imagen original.
- **Bilineal**: en Gabri [2018] se especifica que este interpolador utiliza una media ponderada de los cuatro vecinos más cercanos del píxel, para de esta forma suavizar la salida y evitar los cambios bruscos que puede generar el interpolador anterior.
- **Área**: el nuevo píxel será resultado de un muestreo realizado con relación de área con respecto al píxel original.
- **Bicúbico**: interpolador explicado en Preet [2021], considera 4×4 píxeles vecinos al píxel a calcular, es decir, un total de 16 píxeles, dándole un peso mayor a los cercanos y menor a los lejanos.
- **Lanczos**: tal y como se especifica en Conejero and PTeam [2018], este interpolador considera los $2n\times2n$ píxeles vecinos del original, siendo n el orden utilizado. En el caso de este trabajo, el interpolador Lanczos utilizado tiene un orden de 4, por lo que se tendrían en cuenta 8×8 píxeles, es decir, los 64 vecinos más cercanos.

Algunos de estos interpoladores no están recomendados para su uso en la reducción del tamaño de las imágenes, ya que producen lo contrario a lo que se especifica en el teorema de *Nyquist-Shannon*: la frecuencia de muestreo debe ser superior al doble del ancho de banda, esto se explica en Llamas [2020]. El muestreo realizado para disminuir las dimensiones de una imagen hace que la frecuencia de muestreo caiga por debajo de lo debido y podría ser un problema a la hora de reconstruir la señal original, ya que se pierde demasiada información.

2.1.2. Detección de características y emparejamiento

Siendo un componente esencial de muchas de las aplicaciones de la visión por computador, esta técnica se encarga de encontrar ciertos objetos o elementos concretos en las imágenes.

Principalmente, tal y como se explica en Szeliski [2011] se suelen buscar dos tipos de elementos: características concretas y bordes.

- **Puntos y parches**: las características se suelen utilizar para encontrar una misma localización en diferentes imágenes. Normalmente estos puntos de interés se caracterizan por la existencia de parches en los píxeles vecinos. A la hora de realizar el emparejamiento entre diferentes imágenes se llevan a cabo una serie de procedimientos.

  1. **Detección de características**: se buscan puntos de ambas imágenes que puedan ser fácilmente reconocibles en otras.
  2. **Descripción de características**: se crean descriptores estables para ser comparados con otros.
  3. **Emparejamiento de características**: se buscan los candidatos.

- **Bordes**: al contrario que en la detección de características, lo que se busca encontrar son simplemente los contornos de los objetos, lo que puede llevar a incluso al reconocimiento de qué tipo de objetos se encuentran en las imágenes.

2.1.3. Alineación basada en características

En la Sección 2.1.2 se habla sobre el emparejamiento de características en imágenes. Esta técnica, explicada en Szeliski [2011], va un paso más allá, por lo que se

parte de la base de que inicialmente se ha tenido que aplicar la técnica nombrada anteriormente. Una vez hecho esto, la alineación se encarga de comprobar si el conjunto de pares de características es consistente geométricamente, es decir, si se puede llegar a tener las características en el mismo lugar geométrico en ambas imágenes con simples transformaciones.

Esta técnica es utilizada para simplemente alinear imágenes con fines como la comparación o incluso para determinar la posición de la cámara con respecto a un objeto o escena, esto sería el llamado *pose estimation*.

2.2. Técnicas de aprendizaje automático

El aprendizaje automático o *machine learning* es el conjunto de técnicas que se encargan de encontrar patrones en cantidades muy grandes de datos para, posteriormente, hacer predicciones o apoyar la toma de decisiones. Todo esto se introduce en Murphy [2012].

Hoy en día se cuenta con gran cantidad de datos, pero lo importante de esto no es cuánta información existe, sino qué se puede obtener de ella. Este es el pilar fundamental de estas técnicas.

Como se especifica en Kubát [2017], a diferencia de otro tipo de técnicas en las que se cuenta con una serie de instrucciones que el modelo va a seguir, teniendo como objetivo el desarrollo de una tarea, el aprendizaje automático se encarga de crear algoritmos que aprenden a realizar la tarea por sí mismos, sin ningún tipo de reglas a seguir.

A continuación se detallan cuáles son los diferentes enfoques que puede tomar el aprendizaje automático y las aplicaciones básicas de este.

2.2.1. Enfoques

Dependiendo de la estrategia de aprendizaje que se tome, se encuentran principalmente tres enfoques: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo.

- **Aprendizaje supervisado**: en este enfoque, el entrenamiento del modelo se

realiza de forma que el conjunto de datos que se le facilita lleva asociada la salida deseada, por lo que el algoritmo aprende una función general para asignar las entradas a sus correspondientes salidas. Normalmente se suele utilizar en problemas de clasificación.

- **Aprendizaje no supervisado**: al contrario que en el enfoque supervisado, no se ofrecen las salidas que deberían asignarse a las entradas, por lo que el modelo debe encontrar patrones en las entradas para de esa forma poder asignar las salidas correspondientes. El análisis de grupos suele ser la tarea más realizada con este tipo de enfoque.

- **Aprendizaje por refuerzo**: la base de este enfoque es la experiencia. Los algoritmos realizan acciones que provocan un beneficio. Un ejemplo claro sería un algoritmo que aprende a jugar al ajedrez simplemente por el hecho de jugar con humanos. A la hora de realizar un movimiento el algoritmo tendrá en cuenta estrategias aprendidas de los jugadores previos.

2.2.2. Aplicaciones

Una vez explicados los enfoques que se pueden tomar, queda por detallar cuáles son las diferentes tareas que se pueden llevar a cabo por medio del aprendizaje automático. Entre las más importantes se encuentran las siguientes:

- **Clasificación**: se trata de la asignación de la entrada en diferentes clases o categorías. Un ejemplo de esto sería la diferenciación entre perros y gatos de un conjunto de datos que contiene imágenes de ambos.

- **Regresión**: en lugar de asignar la entrada a un grupo concreto, se le asigna una variable continua, como puede ser una puntuación del 1 al 10.

- **Análisis de grupos**: como su nombre indica, se analizan las características de los elementos de entrada y se clasifican en grupos de tal forma que los elementos de un mismo grupo son más parecidos entre ellos que si se comparasen con otros grupos.

2.3. Modelos supervisados para clasificación

Esta sección se centra en la parte del aprendizaje automático que será más relevante para la realización del trabajo: la tarea de clasificación con aprendizaje supervisado como enfoque.

Como ya se introdujo en la Sección 2.2.2, la tarea de clasificación consiste en la asignación de un elemento de entrada a una categoría concreta. Si se tiene en cuenta que el enfoque elegido es el aprendizaje supervisado, habría que facilitar al modelo los elementos de entrada junto a una etiqueta, que le hace saber la clase o categoría a la que pertenece, de forma que el algoritmo sea capaz de aprender una función con la capacidad de asociar cada patrón de entrada a la categoría de salida que de corresponde.

Si se tiene en cuenta el número de valores distintos que puede tomar la salida, como bien se explica en Murphy [2012], se puede diferenciar entre tres tipos de clasificación.

- **Clasificación binaria**: únicamente existen dos categorías. Un problema de clasificación binaria es el tratado en este trabajo ya que, las imágenes a clasificar pueden pertenecer a la clase “incendios” o a la clase “no incendio”, siendo estas excluyentes entre sí.
- **Clasificación multiclase**: en este caso el número de clases es mayor a dos, pudiendo un elemento pertenecer solamente a uno de ellos. Un ejemplo de este tipo de clasificación sería la categorización de plantas en diferentes especies.
- **Clasificación de etiquetas múltiples**: en cualquiera de los dos tipos nombrados anteriormente, puede ocurrir que los elementos no pertenezcan únicamente a un grupo, sino que simultáneamente les corresponda más de una clase. En este tipo concreto se encontraría por ejemplo la clasificación de personas en características físicas: una persona alta y fuerte pertenece a estas dos clases a la vez.

En la Figura 2.1 se muestra un ejemplo muy claro para poder diferenciar los dos últimos tipos de clasificación nombrados. Se tienen tres clases, iguales para ambos tipos, pero las muestras de la clasificación multiclase únicamente pertenecen a una de ellas, mientras que las mostradas en la parte de etiquetado múltiple pueden ser incluidas hasta en los tres grupos a la vez.

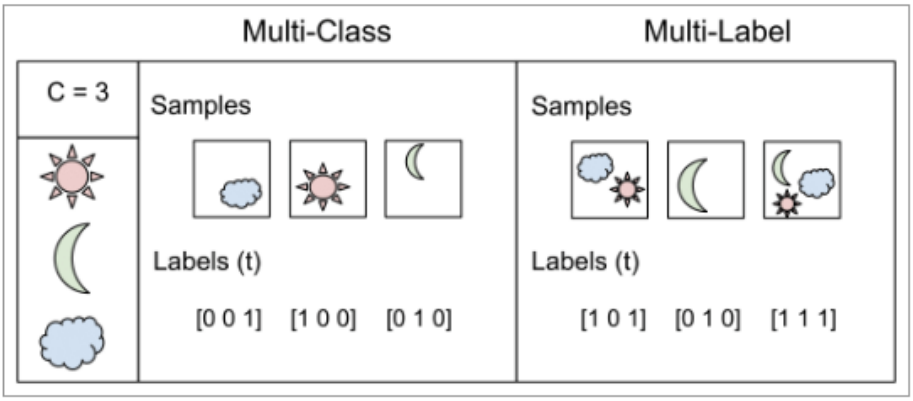

Figura 2.1: Diferencia entre clasificación multiclase y de etiquetas múltiples. Jadhav [2021]

Las técnicas pertenecientes a la clasificación con modelos supervisados forman una larga lista. A continuación se va a hablar brevemente de algunas de ellas teniendo como base la información recogida en Kotsiantis [2007].

2.3.1. Árboles de decisión

Esta técnica pertenece a los algoritmos basados en lógica. Los elementos de entrada van pasando por los nodos del árbol, teniendo que tomar una decisión cada vez hasta llegar a un nodo hoja. Cada elemento del árbol tiene una función, se pueden diferenciar: nodo raíz, nodos intermedios, nodos hoja y ramas.

- **Nodo raíz**: normalmente en este nodo se encuentra la característica que más claramente divide los elementos. Al igual que en los nodos intermedios, habrá que tomar una decisión.
- **Nodos intermedios**: representan una característica concreta que el elemento puede tener o no.
- **Ramas**: muestran las posibles decisiones que se pueden tomar sobre un nodo concreto. En un árbol en el que un nodo muestre la característica “color negro”, las ramas podrían ser “sí” o “no”. En definitiva, marcan el camino a seguir.
- **Nodos hoja**: sus valores están marcados por las diferentes clases o grupos que existen y serán el final del camino tras haber tomado una serie de decisiones y, por tanto, la categoría a la que pertenece el elemento.

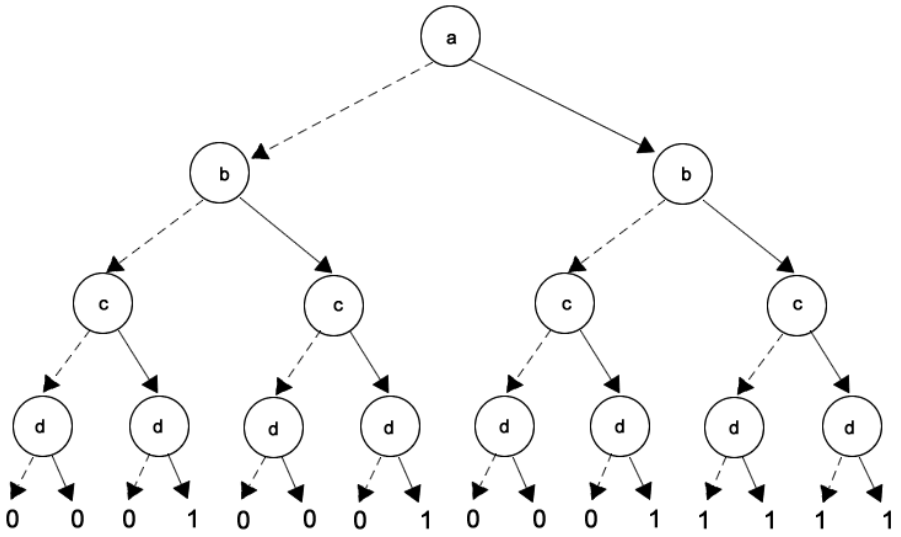

Figura 2.2: Ejemplo de un árbol de decisión. Santos et al. [2018]

En la Figura 2.2 se muestra un ejemplo de árbol de decisión para la clasificación binaria, donde cada uno de los nodos representa una característica. El objetivo es ir pasando por ellos tomando una decisión hasta llegar a la parte inferior, en la que cada hoja muestra la categoría a la que pertenecería el elemento.

2.3.2. Clasificadores de *Naive Bayes*

Dentro de las técnicas estadísticas, los clasificadores de *Naive Bayes* se basan en el teorema de *Bayes*, tal y como se explica en Berrar [2019]. El objetivo es calcular la probabilidad de que un elemento concreto de entrada pertenezca a una clase. La representación gráfica de este algoritmo la componen una serie de grafos con un solo nodo padre representado a la clase y tantos nodos hijo como atributos existan en ese grupo. Esta estructura se puede ver en la Figura 2.3.

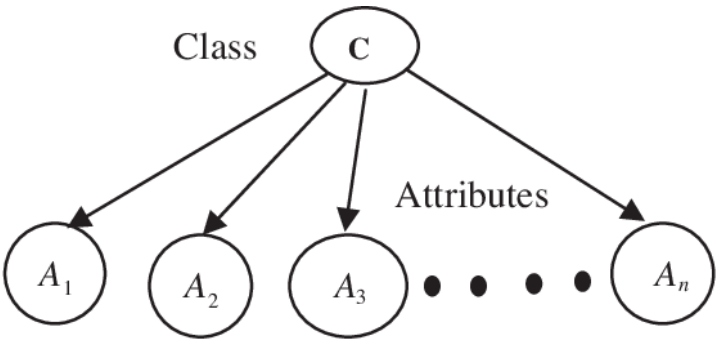

Figura 2.3: Estructura de una red *Naive Bayes*. Ali et al. [2012]

La Fórmula 2.1 es, por tanto, la utilizada para el cálculo de las probabilidades, siendo X_i el elemento a clasificar, y Y_j la categoría en la cual se calcula la probabilidad.

$$P(Y_j|X_i) = \frac{P(X_i|Y_j)P(Y_j)}{P(X_i)} \tag{2.1}$$

2.3.3. Redes bayesianas

Teniendo en cuenta el método estadístico del clasificador *Naive Bayes*, explicado en la Sección 2.3.2 se puede dar un paso más hacia las redes bayesianas. En este caso, la representación gráfica sigue siendo un grafo en el que los nodos son variables aleatorias únicas, representando la probabilidad condicionada entre las variables teniendo en cuenta el sentido de las aristas que las unen. La estructura básica sería la mostrada en la Figura 2.4.

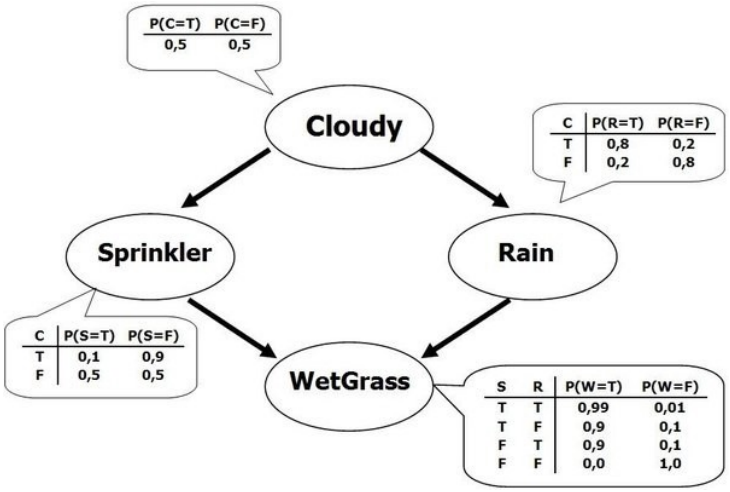

Figura 2.4: Estructura de una red bayesiana. Soni [2018]

2.3.4. Máquinas de soporte vectorial

Esta técnica utiliza el concepto de "margen" para separar los elementos de entrada en las clases correspondientes. El algoritmo sigue unos pasos concretos.

1. Proyección de los patrones de entrada en el espacio n-dimensional generado por las distintas características.
2. Búsqueda del hiperplano que separe los elementos dejando el mayor margen posible con los elementos más cercanos.

3. En caso de ser una clasificación multiclase, repetir el paso 2 tantas veces como sea necesario.

4. Finalmente, los elementos habrán sido separados en las clases y se visualizará perfectamente qué elementos pertenecen a los diferentes grupos.

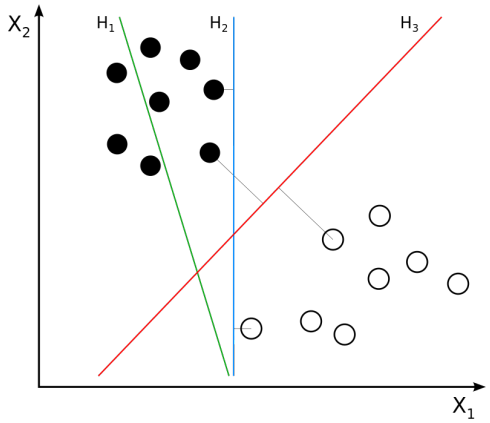

Figura 2.5: Posibles hiperplanos para realizar una clasificación binaria. Daicy [2020]

La figura 2.5 muestra un ejemplo del resultado final tras seguir el procedimiento explicado. Se aprecia que los hiperplanos H_1 y H_2 no consiguen dejar el máximo margen con los elementos más cercanos. El hiperplano H_3 es el óptimo, es decir, consigue diferenciar las dos clases definitivas.

2.3.5. *k* vecinos más cercanos

Este método, también llamado *kNN*, está basado en la vecindad. Como se explica en Guo et al. [2003], para clasificar un elemento se atenderá a la categoría a la que pertenecen sus *k* vecinos más cercanos. El parámetro *k* es definido por el desarrollador y normalmente, se considera la ejecución de este clasificador utilizando diferentes valores para el mismo. De esta forma será más fácil encontrar el parámetro con el que se consiguen unos resultados óptimos.

En la Figura 2.6 se puede ver un ejemplo claro de este clasificador. La estrella es el nuevo elemento a clasificar y los puntos amarillos y morados corresponden a los elementos cuya categoría es conocida. En el caso de escoger los tres vecinos más cercanos, se aprecia que dos de ellos pertenecen a la clase B, por lo que el nuevo elemento también sería clasificado como tal. Sin embargo, si se seleccionan los seis vecinos más cercanos, la mayoría corresponden a la clase A, categorizando por tanto el nuevo elemento dentro de esta clase.

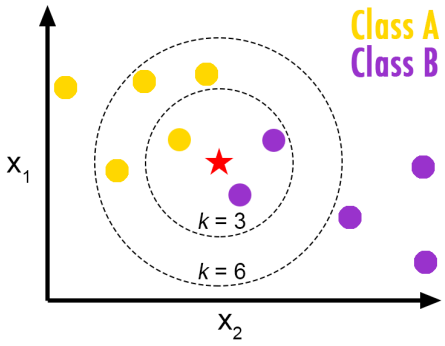

Figura 2.6: Ejemplo para la clasificación con *kNN*. Salcedo [2018]

2.3.6. Perceptrón

Esta última técnica es la más relevante para este trabajo, ya que es la base sobre la que se sustentan la mayoría de técnicas de aprendizaje profundo. Es necesario comprender su funcionamiento correctamente antes de adentrarse en las técnicas que se van a utilizar finalmente para la realización de la experimentación que dará solución al problema propuesto.

Perceptrón de una sola capa

Esta técnica se basa en el uso de "pesos". Cada una de las características es representada como un nodo inicial y por cada elemento de entrada se asigna un valor (algo parecido a una puntuación) a cada una de ellas, normalmente entre -1 y 1, estos son los llamados pesos. Una vez que todas las características tienen sus pesos asignados, se hace la suma total y se comprueba si se supera o no cierto umbral. Aquellos elementos de entrada cuya suma supere este umbral, pertenecerán a una clase, mientras que el resto serán categorizadas cono elementos del otro grupo.

La Figura 2.7 muestra lo descrito en el apartado anterior. En el último paso a realizar se aprecia que se utiliza una función de activación escalonada o *step function*. Las funciones de activación definen si el perceptrón se activa o no. Esta no es la única función de activación existente. A continuación se explican las más relevantes para esta trabajo además de la ya nombrada.

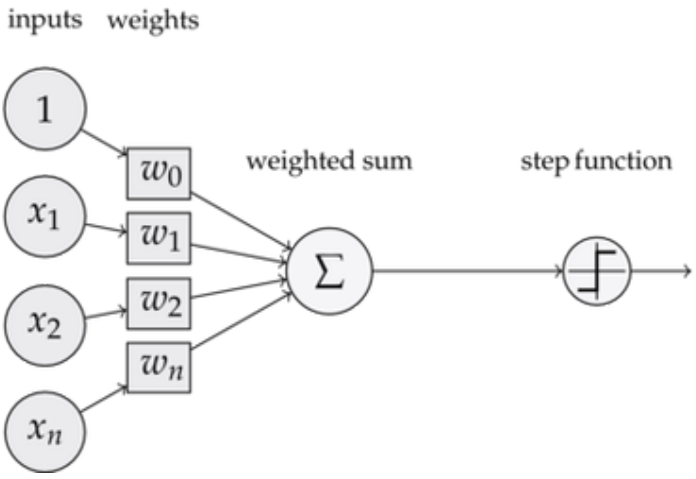

Figura 2.7: Ejemplo de estructura de un perceptrón. B [2017]

Funciones de activación

A la hora de decidir si el perceptrón se activa o desactiva es necesario definir la función de activación que se tendrá en cuenta. Existe una amplia variedad de las mismas, las más populares explicadas en Gupta [2020].

- **Lineal**: la activación es proporcional a la entrada, por tanto existen tantos valores de salida como los que hay de entrada. La Figura 2.8 muestra gráficamente esta función.

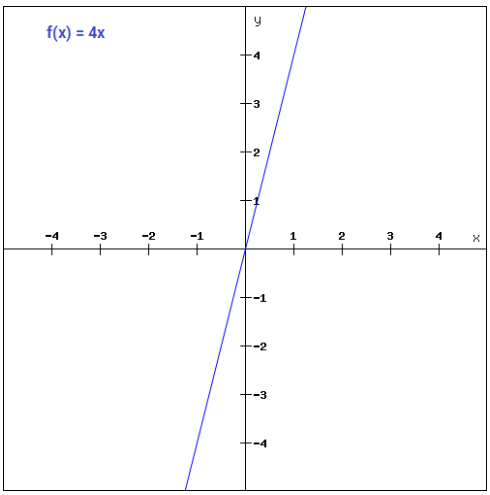

Figura 2.8: Función de activación lineal. Gupta [2020]

- **Escalonada**: mostrada en la Figura 2.9, simplemente se activará o desactivará el perceptrón si se supera o no un umbral. Esta función no es útil cuando se utiliza en una clasificación multiclase.

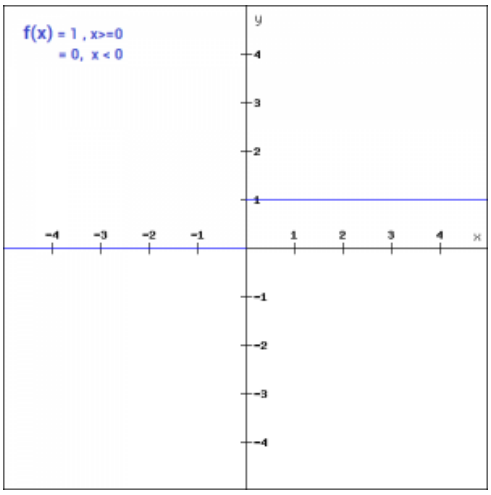

Figura 2.9: Función de activación escalonada. Gupta [2020]

- **Sigmoide**: transforma los valores a un rango entre 0 y 1. Es una función no lineal, lo que implica que la salida tampoco va a ser lineal. Esto se aprecia en la Figura 2.10.

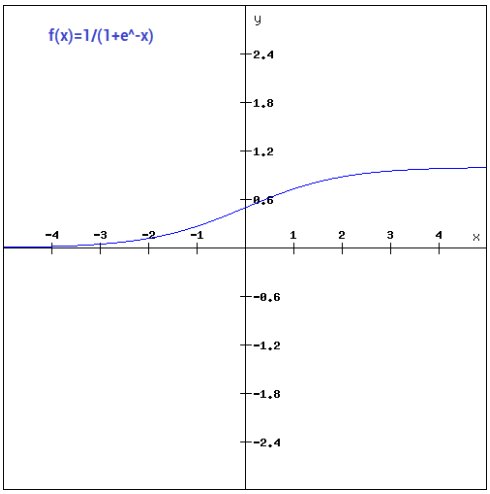

Figura 2.10: Función de activación sigmoide. Gupta [2020]

- ***Tanh***: mostrada en la Figura 2.11, es muy parecida a la función sigmoide, con la diferencia de que es simétrica con respecto al origen de coordenadas, teniendo el rango de valores entre -1 y 1.
- ***ReLU***: las siglas de esta función proceden del término *Rectified Linear Unit*, es decir, Unidad Lineal Rectificada. Como se aprecia en la Figura 2.12, para los valores negativos de entrada, se desactiva el perceptrón.
- ***Leaky ReLU***: versión mejorada de la función ReLU. Tal y como se muestra en la Figura 2.13, para los valores negativos, en lugar de asignar simplemente 0, se asigna una componente mínima de x.

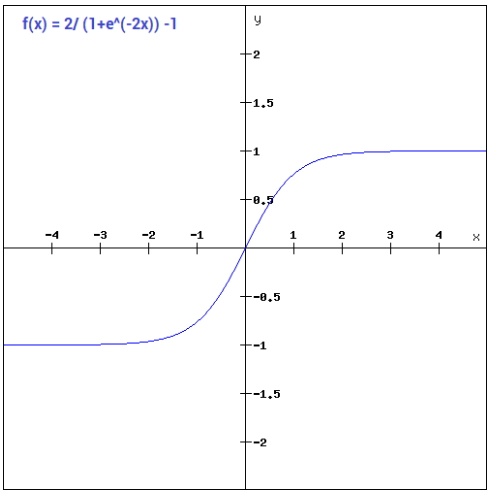

Figura 2.11: Función de activación *tanh*. Gupta [2020]

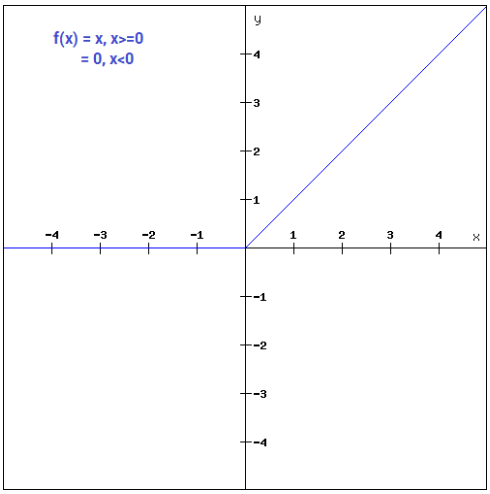

Figura 2.12: Función de activación *ReLU*. Gupta [2020]

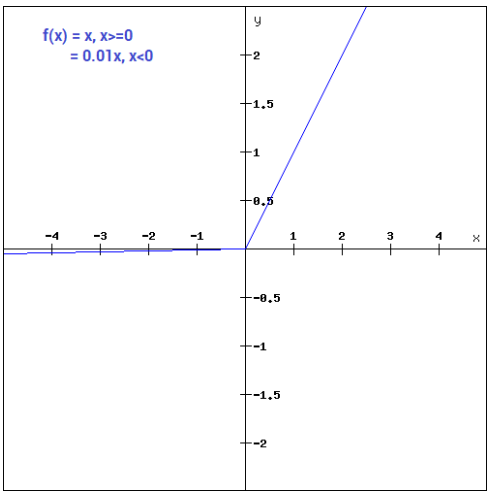

Figura 2.13: Función de activación *Leaky ReLU*. Gupta [2020]

- ***Softmax***: descrita como una combinación de múltiples funciones sigmoide, de forma que devuelve la probabilidad de que un dato corresponda a cada clase.

Perceptrón multicapa: Redes Neuronales Artificiales

Los perceptrones únicamente puede clasificar correctamente si los elementos son linealmente separables. Para solucionar este problema se crean los perceptrones multicapa, de forma que se crea un patrón de conexiones entre varios perceptrones. Esta es la idea básica de Red Neuronal Artificial, en la que cada uno de los perceptrones que la forman son las llamadas neuronas. Normalmente, se pueden clasificar las neuronas que forman parte de una red en tres tipos.

- **Neuronas de entrada**: reciben la información sin procesar.
- **Neuronas de salida**: en ellas se muestran los resultados del procesamiento.
- **Neuronas ocultas**: se encuentran entre la entrada y la salida, siendo las que realizan el procesamiento requerido.

2.4. Estado del Arte: Aprendizaje profundo

Una vez explicado todo lo relativo a la base sobre la que se sustenta la realización de este trabajo, se va a pasar a hablar sobre cuál es la técnica específica que se va a utilizar: las Redes Neuronales Convolucionales. Ya se ha hablado sobre las redes neuronales de forma general, pero antes de hablar sobre la funcionalidad de las redes convolucionales, es preciso tener una visión general del campo del aprendizaje profundo.

El aprendizaje profundo o *deep learning* se encarga de la resolución de problemas, tales como clasificación, segmentación y detección, en el campo del procesamiento de imágenes digitales. Para definirlo correctamente, y tal y como se introduce en Mahony et al. [2019], en primer lugar hay que diferenciar entre el análisis predictivo y descriptivo.

- **Análisis predictivo**: implica la creación de un conjunto de reglas que explican un fenómeno de forma que se minimice el error entre lo que se predice y la realidad.
- **Análisis descriptivo**: implica la definición de un modelo matemático que describa un fenómeno. Para ello es necesario recolectar datos y crear y validar hipótesis.

El aprendizaje profundo es parte del aprendizaje automático, centrándose principalmente en el uso de Redes Neuronales Artificiales, inspiradas en el funcionamiento del cerebro humano. En los últimos años este tipo de técnicas ha dominado el campo de la Inteligencia Artificial, consiguiendo resolver problemas que llevaban mucho tiempo en espera. Por esta razón, actualmente existe la cuestión de si las técnicas de visión computacional se están quedando obsoletas, ya que el aprendizaje profundo parece dar mejores resultados.

El campo de la visión por computador sí que ha perdido interés en los últimos años, pero siguen existiendo procedimientos sobre los que es necesario aplicar este tipo de técnicas. Incluso, se ha demostrado que la aplicación de técnicas tradicionales pueden mejorar los resultados del aprendizaje profundo.

2.4.1. Aprendizaje en Redes Neuronales Artificiales

Las Redes Neuronales Artificiales ajustan su arquitectura y parámetros para obtener un buen rendimiento. Antes de comenzar a explicar el funcionamiento concreto de las Redes Neuronales Convolucionales, conviene explicar el funcionamiento de las Redes Neuronales de forma general, centrando la explicación en las Redes *Feedforward* ya que, son las más estudiadas y utilizadas.

En el caso del aprendizaje supervisado, que es el más relevante para este trabajo, el objetivo es ajustar los pesos, de manera que los valores predichos (la salida) se correspondan lo más posible a los valores reales. Todo esto se explica detalladamente en Salas [2004].

Durante la fase de *feedforward*, se obtienen los valores de salida. Una vez hecho esto, comienza la fase de *backpropagation* o retropropagación, en la cual se calcula el error cometido comparando la salida obtenida y la deseada, utilizando para ello la suma de los errores al cuadrado o el promedio. De esta forma se van ajustando los pesos.

2.4.2. Redes neuronales convolucionales

En el pasado, tal y como se explica en Allison et al. [2016], se utilizaban sensores de humo y calor para la detección de incendios, con el inconveniente de que tenían que estar muy próximos al fuego para realizar la detección correctamente. Esto se traduce en la no detección de un incendio en su época más temprana, pudiendo llegar

a causar el descontrol del mismo. El avance de los ordenadores hace que se cambie esta perspectiva y se utilicen métodos computacionales para detectar fuego.

En el uso de los métodos tradicionales, explicados en Cetin et al. [2013] y en Yuan et al. [2015], se obtenían las características a mano y posteriormente se utilizaba un clasificador, pero las características no eran lo suficientemente específicas y se cometía demasiado error. Para solucionar esto, se comenzaron a usar las Redes Neuronales Artificiales, de forma que son estas las que obtienen las características y posteriormente clasifican.

Dentro del uso de las redes neuronales para encontrar una solución al problema, se llega a la conclusión de que las redes profundas perceptrón no funcionan bien con imágenes, ya que tratan todos los píxeles de la misma forma. Para ello, se empieza a considerar la idea de utilizar las Redes Neuronales Convolucionales.

Este tipo de redes trabajan con imágenes, por lo que la operación básica que realizan es la convolución. Para explicar más detalladamente el funcionamiento de estas, se van a explicar las diferentes capas que las pueden componer y cuál es su utilidad, todas ellas explicadas en Saha [2018].

Capa Convolucional

La entrada de este tipo de capas puede ser la imagen original o un mapa de características, lo cual indicaría que la imagen original ya ha pasado por alguna convolución. En definitiva se cuenta con una serie de valores en una matriz, a los que se va a aplicar un filtro, que puede ser de tamaño variable, pero fijo dentro de la capa actual. Cuando se utilizan filtros pequeños, se logran detectar los cambios locales, mientras que el uso de filtros grandes hace que se tenga una visión más global acerca de la información mostrada en la imagen. Al aplicar un filtro sobre un mapa de características, se obtiene como resultado algo más pequeño, por lo que si esto no es lo deseado, se puede utilizar un método llamado *zero-padding*, que rellena con ceros los bordes del mapa para así poder arrastrar el filtro correctamente por todos los valores de la matriz original. La salida de estas capas es un mapa de características.

Capa de reducción

De la misma forma que ocurría en la capa convolucional, en la capa de reducción o capa *pooling* se recibe un mapa de características y se aplican filtros, pero en este

caso el objetivo es otro, eliminar información redundante. Para ello, escogeremos el tamaño del filtro, por ejemplo 3×3 y se irá superponiendo sobre el mapa. De estos 9 valores, únicamente se escogerá uno, aquel que tenga el valor más grande, en caso de realizar *max-pooling*, o el valor de la media aritmética de todos, si escogemos el método *average-pooling*.

Capa totalmente conectada

Al igual que en las capas anteriores, se recibe un mapa de características, pero en este caso es necesario "aplanarlo", es decir, obtener un vector a partir de la matriz inicial. Una vez esto ocurre y tan y como se explica en Arc [2018], se realiza el cálculo oportuno para que, la capa de salida de la red sea capaz de realizar la clasificación correctamente.

2.5. Estudio previo

Con la idea de tener un conjunto de imágenes que pertenecieran a un lugar antes y durante un incendio, se podría utilizar un método para detectar cuáles son los cambios en las imágenes y así poder extraer las características concretas de una imagen que contiene fuego.

Para esto, se estudiaron las posibles técnicas de visión computacional adecuadas para la resolución del problema.

2.5.1. Índice de similitud estructural

Como se especifica en Rosebrock [2017], este algoritmo se encarga de calcular un valor llamado *structural similarity index* o índice de similitud estructural. En un rango entre -1 y 1 cuantifica cuánto se parecen dos imágenes, siendo -1 totalmente diferentes y 1 idénticas.

En el caso concreto de la diferenciación de imágenes, donde lo que se busca son posibles zonas de cambio, comparar directamente las dos imágenes no va a resultar beneficioso. Una posibilidad sería trocear ambas imágenes de la misma forma e ir calculando el índice para cada par de porciones. Si el índice supera cierto umbral, se

podría suponer que no existen diferencias y, en caso contrario, que es una zona de cambios.

2.5.2. Correspondencia de patrones

El objetivo de la correspondencia de patrones o *template matching* es encontrar una imagen dentro de otra de mayor tamaño, tal y como se explica en OpenCV_team [2021a]. Si se intenta buscar un patrón dentro de una imagen del mismo tamaño es 100 % probable que no se encuentre, por lo que lo único que se puede concluir es que las imágenes son diferentes, pero no se conocen los puntos concretos en los que se encuentran las diferencias.

Para obtener los cambios, se podría trocear una de las imágenes e ir buscando poco a poco todas esas partes en la otra imagen. En caso de que sí se encuentre ese patrón concreto y además esté localizado en el mismo lugar en ambas imágenes, se puede asumir que no existe un cambio. En caso contrario, se consideraría esa zona concreta como una diferencia.

2.5.3. Registrado de imágenes

Es posible que las imágenes de un mismo lugar en instantes diferentes no hayan sido tomadas exactamente desde el mismo ángulo o posición, por lo que previamente a la aplicación de uno de los dos algoritmos explicados anteriormente, sería recomendable alinear las imágenes, de forma que cada píxel aparezca en ambas imágenes en el mismo lugar.

El registrado de imágenes o *image registration* se encarga de esto. Este tipo de algoritmos siguen un procedimiento similar, especificado en Mallick [2018] y en Uberoi [2019].

1. Convierten las imágenes a escala de grises.
2. Con un algoritmo, detectan las características claves de las imágenes.
3. Utilizan un *matcher* o emparejador, que encuentra los k *keypoints* o elementos clave más parecidos.
4. Escoge los pares de elementos clave que más parecidos son dentro de los k seleccionados previamente y desecha los que no son tan parecidos.

5. Utiliza la transformada homográfica para encontrar el desplazamiento que ha ocurrido entre los planos para que haya un cambio entre las dos imágenes.

6. Aplica la transformada homográfica sobre la imagen sin alinear, en este caso la segunda.

Para que las imágenes se puedan alinear correctamente, es necesario que la iluminación y los colores de ambas sean lo más parecidos posible. En caso contrario, la alineación se realizará y resultará en una imagen distorsionada que será inutilizable.

2.6. Métricas de evaluación

Para la evaluación de los distintos sistemas con los que se realizará la experimentación es conveniente atender a una serie de métricas, todas ellas explicadas en Barrios Arce [2019].

- **Pérdida (*Loss*)**: diferencia entre el valor predicho por el modelo y el valor real.
- **Matriz de confusión**: muestra cuál ha sido el desempeño de la red en la clasificación. Concretamente nos da valores numéricos que representan la cantidad de elementos que ha clasificado correctamente o erróneamente. La Figura 2.14 muestra cómo están distribuidos los diferentes valores en la matriz. Existen una

Figura 2.14: Matriz de confusión.

serie de métricas calculables a partir de los valores de esta matriz.

  - **Exactitud (*Accuracy*)** (2.2): datos clasificados correctamente en relación al total.

$$accuracy = \frac{VP + VN}{VP + FN + FP + VN} \tag{2.2}$$

- **Precisión (*Precision*)** (2.3): proporción de verdaderos positivos respecto al total de positivos predichos por el modelo.

$$precision = \frac{VP}{VP + FP} \tag{2.3}$$

- **Sensibilidad (*Recall*)** (2.4): proporción de positivos detectados respecto a los verdaderos positivos reales.

$$recall = \frac{VP}{VP + FN} \tag{2.4}$$

- **Puntuación F1 (*F1-score*)** (2.5): combinación de las métricas *precision* y *recall*. Es de utilidad para comparar el rendimiento de sistemas atendiendo a la vez a ambas.

$$F1score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.5}$$

Existen más métricas para evaluar el rendimiento de un modelo, pero en el caso concreto de este trabajo, se van a tener en cuenta únicamente estas ya que, son suficientes para poder valorar si se está realizando la clasificación binaria correctamente.

Capítulo 3

OBJETIVOS

Una vez conocidos los antecedentes relevantes, y antes de proceder al desarrollo del trabajo realizado, conviene especificar cuáles son los objetivos que se quieren alcanzar. Estos marcarán el camino a seguir durante toda la realización del estudio y experimentación de las posibles soluciones al problema que se plantea.

3.1. Objetivo principal

El objetivo principal de este trabajo es el estudio de técnicas actuales relacionadas con la Ciencia de datos, de modo que se reúnan los conocimientos adquiridos a lo largo del Grado relacionados con este ámbito. Concretamente, se aborda la resolución de un problema real, proponiendo el diseño de un procedimiento automático para la detección incendios en áreas forestales a través del procesamiento y tratamiento inteligente de imágenes usando para ello técnicas de aprendizaje profundo (*deep learning*).

La creación de un modelo que ayude con la detección de incendios puede favorecer a los profesionales encargados de esta tarea, como son los agentes forestales. La automatización de este proceso supone una disminución de recursos humanos y una posible mejora en cuanto a la precisión, ya que puede llegar a evitar errores humanos. Además, la eficiencia de este tipo de sistemas, resulta beneficiosa a la hora de evitar la deforestación, ya que es posible detectar los incendios de forma temprana, haciendo así su extinción más fácil para los servicios encargados de ello.

3.2. Objetivos específicos

Una vez definido el objetivo principal, es necesario profundizar más en él y definir los objetivos parciales que llevarán a alcanzarlo. Estos marcarán las partes de las que consta este trabajo y más concretamente, los hitos a cumplir.

3.2.1. Ciencia de datos

Estudio general del campo de Ciencia de datos y los métodos de Minería de datos esenciales. En concreto, estudio de redes neuronales artificiales y técnicas de aprendizaje profundo.

A la hora de trabajar con imágenes, es idóneo hacer uso de redes neuronales convolucionales. Se hará un estudio previo de su funcionamiento en la Sección 2.4.2 para el posterior uso de las mismas como posible solución al problema que se aborda, teniendo en cuenta diferentes procedimientos dentro de este área de conocimiento.

3.2.2. Búsqueda de un conjunto de datos

Preparación de un conjunto de datos representativo que permita implementar un detector capaz de identificar la presencia de un incendio.

Concretamente, se pretende encontrar un conjunto de imágenes forestales de cualquier índole, que contenga imágenes en las que exista un incendio activo y también imágenes en las que no haya fuego ni ningún tipo de señal que indique que pueda estar ocurriendo un incendio.

Previamente a trabajar con el conjunto pertinente será necesario adaptarlo para su posterior uso en el entrenamiento, validación y evaluación de los distintos sistemas implementados. Esto puede suponer la aplicación de recorte, redimensionado e incluso etiquetado de las mismas en las categorías pertinentes.

3.2.3. Creación del sistema

Diseño de varias redes neuronales que se puedan emplear para abordar la tarea, detallando la arquitectura y fundamentación de cada una de ellas, teniendo siempre

presente la limitación temporal y de recursos.

Implementación de todos los sistemas de forma adecuada para entrenarlos a partir del conjunto de datos escogido.

Evaluación de los diferentes sistemas y comparación de resultados para escoger de forma razonada el mejor sistema que da solución al problema propuesto.

3.2.4. Redacción de documentación

Especificación del proceso seguido en todo momento para el cumplimiento de los objetivos detallados.

Esto conlleva la redacción de esta memoria, recogiendo toda la información necesaria para comprender el campo en el que se trabaja, los experimentos realizados y los resultados obtenidos.

Explicación detallada de la parte teórica de los sistemas utilizados y su diseño y adaptación final para ser utilizados con el fin ya especificado. Así como también, especificación íntegra de los materiales utilizados para el procesamiento de los datos y el uso de los procedimientos.

3.3. Planificación temporal

Para ver de una forma más clara cómo se han repartido las tareas principales en el tiempo, el diagrama de *Gantt* de la Figura 3.1 lo muestra de una forma visual.

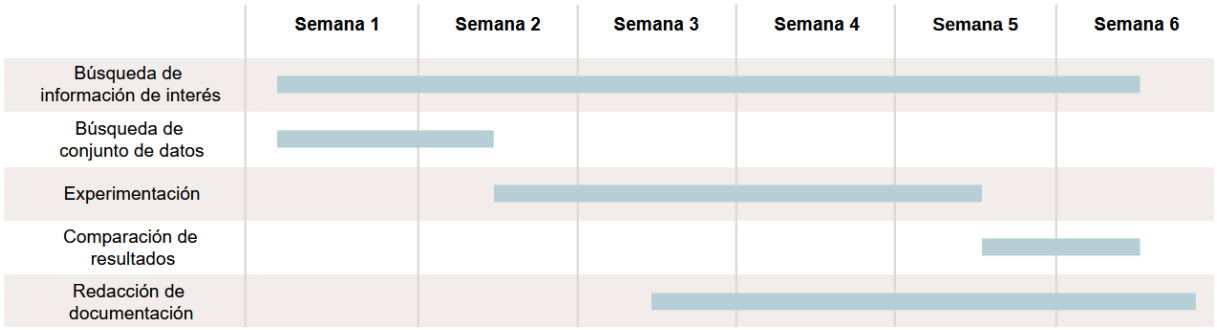

Figura 3.1: Diagrama de Gantt del proceso de realización del trabajo.

- **Búsqueda de información de interés**: esta tarea, extendida a lo largo de prácticamente la totalidad del tiempo, implica la adquisición de cualquier tipo de co-

nocimiento necesario para la redacción de la memoria y realización de experimentos.

- **Búsqueda de conjunto de datos**: antes de comenzar a diseñar los sistemas que den solución al problema tratado, es necesario encontrar una base de datos que sirva como entrada para los modelos. Este proceso se extendió durante más de una semana debido a la dificultad para encontrar conjuntos que se adecuaran a lo que inicialmente se buscaba.
- **Experimentación**: tras encontrar el conjunto de imágenes adecuado, el siguiente paso fue el diseño y creación de los modelos para su posterior entrenamiento y evaluación. Este proceso se alargó durante aproximadamente tres semanas.
- **Comparación de resultados**: una vez obtenidas las métricas de rendimiento de los modelos generados, encontrar las diferencias entre unos y otros es relativamente fácil, por lo que este trabajo no llevó más de una semana.
- **Redacción de documentación**: durante cerca de 4 semanas, se estuvo escribiendo esta memoria para que quedase completa y fuera legible. Sin duda es la parte más importante ya que, en ella se muestra todo el trabajo realizado en el resto de etapas.

Capítulo 4

MATERIALES Y MÉTODOS

Es necesario tomar decisiones a la hora de realizar los experimentos que nos llevarán a cumplir los objetivos especificados en el Capítulo 3.

En este capítulo se especifican todos los detalles con respecto al conjunto de datos que se utilizará y a los diferentes métodos que se analizarán para encontrar un sistema eficaz para la detección de incendios en imágenes.

4.1. Conjunto de datos

En la búsqueda de un conjunto de imágenes adecuado para la realización del trabajo y finalmente, alcanzar el objetivo propuesto, se barajan varias posibilidades en cuanto al tipo de imagen.

- Imágenes de satélite.
- Imágenes de un lugar concreto antes y durante un incendio.
- Imágenes tomadas a vista de cámara de fototrampeo.

4.1.1. Imágenes de satélite

En este tipo de imágenes se puede abarcar mucho terreno ya que están tomadas desde gran altitud. A la hora de detectar un incendio en tiempo real, serían necesarias

muchas menos imágenes de este tipo, ya que se podría abarcar una zona muy extensa sin necesidad de tener una imagen por cada pocos kilómetros cuadrados.

Un inconveniente que se puede encontrar en estas imágenes vendría dado por la presencia de humo, ya que este puede tapar por completo la zona sin permitir visualizar lo que hay debajo. El humo no necesariamente indica fuego, ya que puede que siga habiendo humo en una zona en la que el incendio ya se extinguió, por lo que cabe la posibilidad de que se estén detectando incendios en un lugar en el que realmente no lo hay.

Otro problema que presentan este tipo de imágenes es la posible falta de datos recientes de las zonas. Los satélites no se encuentran siempre sobre un lugar concreto, sino que van orbitando, cubriendo la mayor parte del planeta, por lo que por una zona concreta solamente pasaría cada X horas o incluso días. Esto haría que el intervalo de tiempo entre dos imágenes consecutivas sea demasiado, demorándose así la detección de posibles incendios.

4.1.2. Imágenes antes y durante un incendio

Si contamos con un par de imágenes que nos muestran cómo era una zona concreta antes de que ocurriera un incendio y cómo es la zona en el momento en el que está ocurriendo el incendio, se podrán comparar y comprobar si hay diferencias notables que denoten la presencia de fuego.

Para comprobar si el cambio que ha ocurrido se debe a un incendio se tendrán que estudiar cuáles son las características específicas que aparecen en las imágenes que contienen fuego. Esto puede llevar a confusión y finalmente detectar un incendio cuando simplemente el cambio sufrido por el terreno se debe a otras causas como desbordamientos o un cese del cultivo.

Este enfoque, al estar basado en comparación de imágenes, hace que sea muy probable que se necesite generar un modelo para cada ubicación, ya que cada lugar cuenta con características muy concretas. Lo que se pretende en este trabajo es generar un único modelo que sea capaz de identificar la presencia de un incendio independientemente a la ubicación y las cualidades de la misma.

4.1.3. Imágenes de fototrampeo

Las imágenes de fototrampeo son tomadas desde cámaras posicionadas en los árboles o incluso entre la vegetación. Normalmente son utilizadas para detectar la presencia de animales y realizar inventariado de especies. Este tipo de cámaras se disparan automáticamente con cualquier movimiento: animales que pasan, vegetación movida por el viento, cambio brusco de luz, etc. Esto es muy beneficioso ya que, el movimiento de las llamas también sería motivo de captación de imágenes. Por si esto no fuera suficiente, cabe la posibilidad de incorporar a estos dispositivos sensores de temperatura adicionales que puedan activarlos cuando esta supere cierto umbral.

En este tipo de imágenes se abarca una zona pequeña, teniendo en cuenta que los propios árboles y la vegetación impedirán ver qué hay más allá, por lo que se necesitan muchas de estas cámaras para poder hacer un estudio de una zona extensa.

Sin embargo, cabe destacar que estas imágenes permiten una muy buena visualización del fuego y el humo. Como se especifica en la Sección 4.1.1, el humo puede invisibilizar la zona y no permitir comprobar si hay fuego o no. En este caso, y puesto que las imágenes están tomadas desde una posición baja y el humo tiende a subir, se deja a la vista el posible fuego.

4.1.4. Conjunto de datos final

Finalmente, tras hacer una búsqueda de un conjunto de imágenes que perteneciera a cualquiera de las tres alternativas nombradas en las Secciones 4.1.1, 4.1.2 y 4.1.3, se encontró uno que contiene imágenes tomadas desde la perspectiva de un ser humano, que se podría incluir dentro del tercer grupo nombrado. El conjunto, obtenido de Gamaleldin et al. [2020], cuenta con dos carpetas de imágenes diferenciadas. Una de ellas contiene imágenes en las que está ocurriendo un incendio forestal y la otra contiene imágenes de zonas forestales sin incendio ni deforestación. Este conjunto de datos es adecuado para el entrenamiento y evaluación de un red neuronal convolucional.

	Número de imágenes
Incendio	755
No Incendio	244
Total	999

Tabla. 4.1: Imágenes totales en el conjunto de datos.

En la Tabla 4.1 se muestra la proporción de imágenes del conjunto. Como se puede

apreciar, cuenta con 999 imágenes en total, repartidas entre las dos categorías a estudiar.

4.2. Procesamiento inicial del conjunto de datos

El conjunto de datos escogido finalmente cuenta con imágenes muy variadas, incluidas imágenes que no son relevantes para este trabajo. Por ello, se hizo inicialmente una revisión manual para eliminar aquellas imágenes que cumplían alguna de las siguientes características.

- Contienen personas u objetos como vehículos, edificios, etc.
- Son claramente imágenes de ciudades.
- Son imágenes artificiales obtenidas de videojuegos o similares.

Figura 4.1: (1)Imágenes eliminadas. (2)Imágenes conservadas.

En la Figura 4.1 se muestran ejemplos de imágenes que cumplen alguna de las características nombradas con anterioridad en la fila superior. En la fila inferior se pueden ver imágenes que son completamente correctas y que se mantienen por tanto en el conjunto de datos.

Finalmente, tras eliminar todas las imágenes pertinentes del conjunto de datos inicial, se cuenta con 623 imágenes en total, de las que aproximadamente dos tercios corresponden a incendios y el tercio restante a imágenes forestales sin presencia de fuego o indicios de que lo haya. Esto queda plasmado en la Tabla 4.2

	Número de imágenes
Incendio	420
No Incendio	203
Total	623

Tabla. 4.2: Imágenes totales en el conjunto de datos tras eliminar las erróneas.

Para poder trabajar con una red neuronal convolucional, es preciso que todas las imágenes con las que se va a entrenar la red sean del mismo tamaño y misma relación de aspecto. Para ello, se realizó un código en Python para ajustar todas las imágenes a una relación 4:3, recortándolas como fuera necesario. Existen varias razones para escoger esta relación de aspecto concreta.

- La mayoría de las imágenes contaban con un ancho mayor al alto, por lo que convenía escoger una relación de aspecto que no eliminase demasiada información.
- Los incendios suelen estar centrados en la imagen, es coherente aprovechar toda la información posible del alto de la imagen, aunque haya que recortar por los lados. La adaptación a una relación de aspecto 4:3 recortará principalmente el ancho de la imagen, dejando el alto prácticamente intacto en la mayoría de las imágenes.

Previo a la realización del procesado pertinente de las imágenes, se cuenta con imágenes muy diferentes en tamaño y relación de aspecto. En la Figura 4.2 se muestran unos ejemplos de cada una de las categorías.

A continuación se mostrará un ejemplo concreto con una de las imágenes, siguiendo paso a paso el proceso llevado a cabo sobre todas las imágenes del conjunto.

La Figura 4.3 muestra una de las imágenes del conjunto de datos, que tiene un tamaño de 1440×960 píxeles. Para poder ajustarla a una relación 4:3, se atiende a cuál sería la altura adecuada para un ancho de 1440. La altura que respeta la relación de aspecto es de 1080 píxeles, la cual supera la altura original de la imagen, por lo que no podemos ajustarla de esta forma, ya que, habría que añadir franjas negras en las partes superior e inferior de la imagen, lo que podría llevar a nuestra red neuronal a confusión. En el caso de aquellas imágenes que no tuvieran este problema, recortaríamos el alto de la imagen para ajustarlo al obtenido.

Como alternativa, podemos repetir el proceso a la inversa, es decir, buscamos la anchura que respetaría la relación 4:3 dejando fija la altura. Para una altura de

Figura 4.2: Imágenes del conjunto de datos sin procesar. Tamaños de izquierda a derecha y de arriba a abajo: 730×363 píxeles, 1280×720 píxeles, 2048×1365 píxeles y 1354×668 píxeles.

Figura 4.3: Imagen original del conjunto de datos con tamaño 1440×960 píxeles.

960 píxeles, necesitamos una anchura de 1280 píxeles. Simplemente recortamos la imagen 80 píxeles en la zona izquierda y 80 en la derecha.

Este proceso, aplicado a todas las imágenes hace que se obtenga un conjunto de datos en la que todas las imágenes cuentan con la misma relación de aspecto. El resultado final sobre los ejemplos mostrados en la Figura 4.2 sería el que aparece en la Figura 4.5

La mayoría de las imágenes tenían un tamaño que superaba los 1000×1000 píxeles, por lo que era conveniente reducir este tamaño ya que de lo contrario sería más

Figura 4.4: Imagen Adaptada a una relación de aspecto de 4:3.

Figura 4.5: Imágenes del conjunto de datos procesadas.

costoso computacionalmente trabajar con ellas. Por tanto, tras ajustar todas las imágenes a una relación de aspecto de 4:3, proceso tras el cual se obtiene un resultado como el que se aprecia en la Figura 4.4, se redimensionaron a un tamaño de 256×192 píxeles.

4.2.1. Particionado del conjunto de datos

A la hora de trabajar con redes neuronales, y siempre que se trabaje con conjuntos de datos, es necesario dividir el conjunto de imágenes en varios grupos de forma que la evaluación se realice con datos que el modelo nunca ha visto y por tanto, sea totalmente honesta. Los porcentajes utilizados se especifican a continuación. Cabe destacar que esta configuración no es la única que se podría realizar, aunque es la más habitual.

- **Entrenamiento**: contiene el 60 % de las imágenes. Utilizado para entrenar el modelo.
- **Validación**: contiene el 20 % de las imágenes. Con este conjunto se realizan ajustes de parámetros.
- **Test**: contiene el 20 % de las imágenes. Este grupo se utiliza para evaluar el modelo y obtener las métricas de rendimiento

En la Tabla 4.3 se especifica la cantidad de imágenes que finalmente forman parte de cada conjunto.

	Porcentaje de imágenes	Número de imágenes
Entrenamiento	60 %	376
Validación	20 %	124
Test	20 %	124

Tabla. 4.3: Particionado del conjunto de imágenes.

Cabe destacar que, como se aprecia en la Tabla 4.2 de la Sección 4.2, el conjunto de datos está claramente desbalanceado. La cantidad de imágenes que presentan incendios es prácticamente el doble del número de imágenes que no contienen fuego. Este factor es algo que generalmente afecta de forma importante a los clasificadores y por tanto, al rendimiento de los sistemas con los que se va a trabajar.

4.3. Procedimientos

Para la realización de la experimentación correspondiente, se han empleado una serie de procedimientos con el objetivo de obtener un sistema basado en *deep learning* que ofrezca buenos resultados a la hora de detectar incendios forestales en imágenes.

4.3.1. Red Neuronal Convolucional

Con el objetivo de obtener el mejor sistema conviene analizar diferentes alternativas, todas ellas basadas en el uso de redes neuronales convolucionales de una forma u otra. En la Sección 2.4.2 se explica el fundamento teórico de estas.

Red neuronal desde cero

Las capas que componen una red neuronal pueden variar como el creador de la misma desee. La variación de la configuración de esta hará que el rendimiento mejore o empeore.

La configuración elegida para la red neuronal con la que se han realizado los experimentos de esta alternativa en concreto se detalla a continuación.

- Cuatro capas de convolución encargadas de obtener los mapas de características de la entrada.
- Tres capas *pooling* encargadas de eliminar información redundante, concretamente utilizando *max-pooling*, es decir, escogiendo siempre el mayor valor entre los que se elige.
- Una capa *flatten* para aplanar la entrada antes de pasarla a las capas totalmente conectadas.
- Tres capas totalmente conectadas que serán las que realicen la clasificación en las diferentes categorías.

La Figura 4.6 muestra un esquema con la estructura escogida para esta red especificando los parámetros de aquellas capas que los requieran.

En cuanto al resto de parámetros de la red, en la Tabla 4.4 se especifican todos ellos, con los valores que se escogieron para la realización de los experimentos. La decisión de definir *ReLU* como función de activación principal vino dada porque es la más usada y conocida, por lo que hay mucha documentación acerca de su funcionamiento. En cuanto al número de épocas, se fijó a 15 para evitar que el proceso de entrenamiento fuera demasiado prolongado. “adam” como optimizador facilita el trabajo debido a que se encarga de ajustar parámetros como el ratio de aprendizaje.

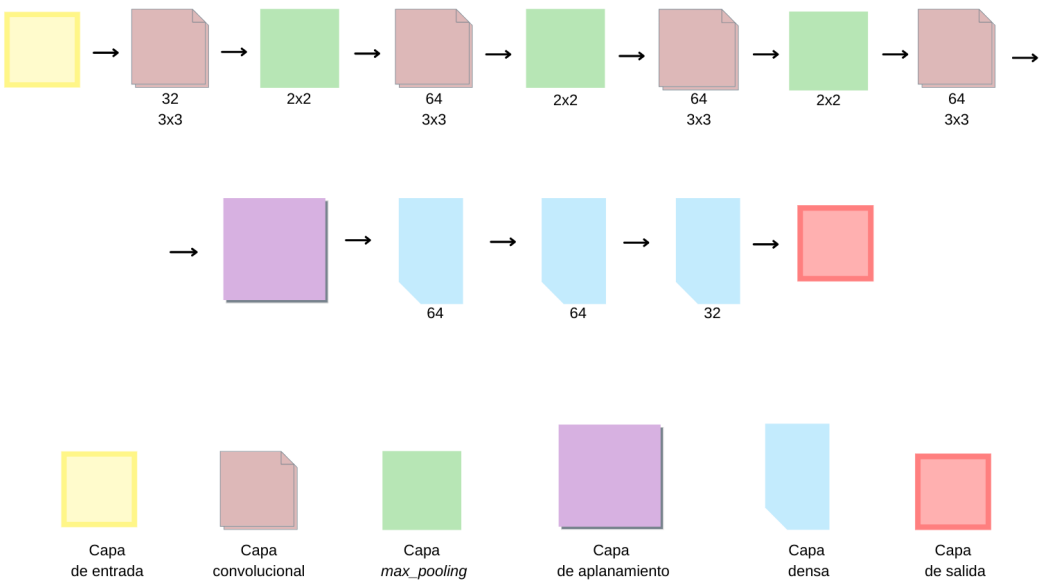

Figura 4.6: Arquitectura de la red creada desde cero.

Parámetro	Valor
Función de activación de las capas intermedias	*ReLU*
Función de activación de la capa de salida	*linear*
Batch size	32
Épocas	15
Optimizador	*adam*
Modelo	Secuencial

Tabla. 4.4: Parámetros importantes a tener en cuenta.

AutoKeras

Con el objetivo de encontrar un modelo de red neuronal que funcione de forma eficaz con el conjunto de datos escogido, existen infinitas configuraciones posibles con las que experimentar, pero sería imposible trabajar con todos estos modelos individualmente.

Autokeras, procedimiento explicado en Jin et al. [2019], ayuda a encontrar el mejor modelo y por tanto, la configuración de la red neuronal que mejor se adapta a los

datos, de forma automática. Esto no quiere decir que vaya a encontrar un modelo con el que obtengamos un porcentaje del 100 % de aciertos a la hora de la evaluación. Esto dependerá de varios factores como el número de pruebas que realice y por supuesto, el tiempo y los recursos que se le faciliten.

Transfer Learning

El llamado *transfer learning*, como su nombre indica, consiste en la transferencia de aprendizaje de una red neuronal a otra tal y como se especifica en Pan and Yang [2010]. El problema que se plantea en este trabajo pertenece a la clasificación de imágenes en categorías, por tanto, es posible, gracias a este concepto, tomar una red neuronal pre entrenada, que ya es capaz de distinguir ciertos patrones, y utilizar este aprendizaje con nuestro propio modelo.

Existe una gran cantidad de redes neuronales entrenadas con conjuntos de datos muy extensos que pueden descargarse fácilmente para hacer uso de ese aprendizaje que ya está realizado. El proceso a seguir una vez descargados los modelos es sencillo. En primer lugar se eliminará la última capa de la red neuronal, ya que esta es la encargada de realizar la clasificación en las distintas categorías con las que se cuenta en el conjunto de datos con el que se entrenó. A continuación se "congelarán" el resto de capas de forma que los pesos se queden como estaban, reteniendo así el aprendizaje. Por último, se añade una capa totalmente conectada para que esta vez haga la predicción según las categorías en las que se divide nuestro conjunto de datos. Esta última capa será la única que se entrena.

Este procedimiento ahorra tiempo y recursos, pero no siempre funcionará mejor que una red neuronal entrenada al completo, ya que las características que la red pre entrenada aprendió en primera instancia pueden ser demasiado genéricas.

A continuación se especifican cuáles son las redes pre entrenadas que se utilizarán en la experimentación de este trabajo y algunas características importantes de cada una de ellas.

- ***InceptionV3***: red profunda con 48 capas, Milton-Barker [2019], entrenada con el conjunto *ImageNet*, que cuenta con más de un millón de imágenes pertenecientes a 1000 clases. Esta red es más eficiente que sus dos versiones anteriores debido a la reducción de parámetros por medio del uso de convoluciones factorizadas y más pequeñas, tal y como se explica en Kurama [2020].
- ***ResNet50***: red que cuenta con 50 capas, como se especifica en Kaushik [2021].

48 capas convolucionales, una capa de *max-pooling* y una capa *average-pooling* forman este modelo que es uno de los más eficientes en la clasificación de imágenes. Entrenada con el conjunto *ImageNet*.

- ***Xception***: el nombre de esta red procede del término *Extreme Inception*, Chollet [2017]. Está compuesta por 36 capas convolucionales separables, lo que hace la modificación de la red muy sencilla, ventaja que presenta frente a redes como *InceptionV3*, también utilizada en este trabajo.

Existen artículos que realizan comparaciones entre las diferentes redes pre entrenadas. Un ejemplo sería el experimento realizado en Cho and Choi [2020], cuyas conclusiones sitúan a *InceptionV3* por encima de *Xception* en cuanto a exactitud.

4.3.2. *Data Augmentation*

Existe la posibilidad de que la cantidad de imágenes contenidas en el conjunto de datos escogido sea insuficiente para poder entrenar una red o, si es suficiente, es probable que no funcione correctamente. Para corregir este problema se hace uso de *Data Augmentation*, explicado en Xie et al. [2020]. El objetivo de utilizar este procedimiento es conseguir aumentar la cantidad de imágenes de manera artificial, partiendo de un conjunto de imágenes concreto.

Por cada imagen del conjunto de datos inicial, se creará una o más imágenes, a partir de la realización de operaciones como rotar, trasladar y voltear, sobre la imagen base.

Figura 4.7: Ejemplo de *Data Augmentation* sobre una imagen del conjunto de datos.

4.4. Software y Hardware

Una vez escogidos los procedimientos a utilizar y el conjunto de datos, es indispensable elegir el software y definir las capacidades del hardware del que se va a hacer uso para realizar la experimentación.

4.4.1. Lenguaje de programación

En la búsqueda de un lenguaje de programación que permita la ejecución de todos los experimentos a realizar es necesario que se cumplan una serie de requerimientos.

- Inclusión de paquetes que permitan el procesamiento inicial de las imágenes.
- Disponibilidad de bibliotecas que faciliten el uso de aprendizaje profundo.
- Facilidad a la hora de la obtención de métricas de rendimiento y la visualización de resultados para la comparación de los mismos.

Ya que el lenguaje de programación *Python* cumple todas estas exigencias y es el más usado en el campo de la ciencia de datos, es el elegido para la realización de la experimentación a realizar en este trabajo.

4.4.2. Entornos de programación

Es básico contar con un entorno de desarrollo para definir, entrenar y finalmente comprobar el funcionamiento de las redes neuronales de los distintos procedimientos. Se utilizarán varios entornos de programación para ahorrar en recursos y tiempo.

PyCharm

Se utilizará este IDE de Python en versión *Community*. A la hora de programar con Python, es unos de los entornos más conocidos y fáciles de usar. La experiencia previa con este entorno lleva a la habituación y por tanto a que sea mucho más fácil empezar a trabajar con él en lugar de hacerlo con otro entorno totalmente desconocido. Este software es totalmente gratuito en la versión que se va a utilizar, obteniéndose de Tucker et al. [2021]

Google Colaboratory

Al igual que PyCharm, es un entorno de programación de Python, Google [2018]. No necesita ser instalado, simplemente ejecutará nuestro código en un servidor accesible a través de Internet, utilizando el navegador como interfaz de usuario. La ejecución del código se puede realizar en bloques, lo que facilita el proceso, ya que se podrá ejecutar el mismo por partes separadas y en la nube, ahorrando tiempo y recursos. Además, permite el uso de gráficas con la biblioteca *matplotlib*, con la que podremos ver los resultados de una forma más visual.

4.4.3. Bibliotecas

Para trabajar con Python en los entornos de programación escogidos, se necesitan una serie de bibliotecas que nos permitan realizar la experimentación.

os

Esta biblioteca facilita la realización de tareas que dependen del sistema operativo en el que se esté ejecutando el código. Entre estas tareas se encuentran el acceso a directorios, edición de los mismos, etc.

En este trabajo, facilitará el trabajo con el conjunto de datos, de forma que podamos listar todas las imágenes y trabajar con ellas iterativamente.

En Python_Software_Foundation [2021a] se encuentra toda la documentación de este paquete.

cv2

En el procesamiento de imágenes y el aprendizaje profundo es imprescindible ya que, incluye todas aquellas herramientas necesarias para llevar a cabo tareas relacionadas con ello.

Con esta biblioteca en particular, se realizará el preprocesamiento de las imágenes explicado en la sección 4.2. Principalmente, ayudará a trabajar con las imágenes de forma sencilla. Las funciones que incluye permiten abrir, recortar y redimensionar las imágenes.

Toda la información necesaria para utilizar esta biblioteca se encuentra en OpenCV_team [2021b]

tensorflow

Es la biblioteca principal que contiene las funciones para crear, entrenar y evaluar una red neuronal convolucional. En los tres procedimientos explicados en la sección 4.3.1 será necesario hacer uso de esta biblioteca.

En la experimentación con aprendizaje profundo esta biblioteca es la más utilizada por los expertos y también por los principiantes, por la facilidad que ofrece para realizar tareas muy complejas y la gran cantidad de documentación disponible. Como se explica en Sanseviero [2019], no solamente es útil para el entrenamiento de redes neuronales, sino también para infinidad de proyectos como el aprendizaje por refuerzo.

matplotlib

Esta biblioteca, obtenida de Python_Software_Foundation [2021b], permite la creación de todo tipo de figuras y gráficas para la representación de cualquier tipo de datos. Puede ser utilizada en ejecutables de *Python*, consola e incluso en servidores de aplicación web.

Empleada en este trabajo para mostrar de forma visual los cambios de las métricas que sean necesarias durante las diferentes fases del entrenamiento de las redes. Esta biblioteca únicamente se utilizará en aquellos experimentos que se realicen en el entorno de programación *Google Colaboratory*.

NumPy

Según NumPy [2021], esta biblioteca ofrece herramientas para vectorización e indexación, así como para la realización de funciones matemáticas, álgebra lineal e incluso transformaciones como la de *Fourier*.

Se utilizará principalmente con dos objetivos.

- Establecer una semilla en el generador de números aleatorios para obtener resultados similares de distintas ejecuciones con la misma configuración.

- Crear los vectores de tipo *array* que contienen las imágenes de los distintos conjuntos especificados en la sección 4.2.1 y las etiquetas de las mismas.

4.4.4. Hardware

El hardware del equipo en el que se realizan los experimentos es el mostrado en la Tabla 4.5.

Componente	Características
CPU	i7-7700 3.60GHz
RAM	8GB DDR4
Almacenamiento	1TB HDD
GPU	-

Tabla. 4.5: Prestaciones del equipo.

Para el entrenamiento y validación de los modelos, y debido a la carencia de GPU en el equipo propio, se hizo uso de Google Colab, que ofrece una GPU *12GB NVIDIA Tesla K80* que, como se especifica en Bar-El [2018], puede ser utilizada hasta 12 horas de forma continuada.

Capítulo 5

RESULTADOS

En este capítulo se muestran todos los resultados obtenidos en los diferentes experimentos realizados con las tres alternativas especificadas en la Sección 4.3.1. Cabe destacar que que los resultados mostrados en las tablas corresponden siempre a la evaluación de los modelos usando datos de test.

Además, se compararán los resultados para analizar cuál es la configuración más adecuada y finalmente el mejor sistema.

5.1. *Data Augmentation*

El conjunto de datos escogido finalmente, especificado en la Sección 4.1.4, cuenta con un total de 623 imágenes. Esta cantidad puede llegar a ser insuficiente a la hora de entrenar, validar y evaluar una red neuronal, por lo que previamente a la realización de cualquier experimentación se aplica *data augmentation*, técnica cuyo funcionamiento se explica concretamente en la Sección 4.3.2.

El aumento de los datos se realiza de dos maneras diferentes para, posteriormente, poder escoger qué conjunto de imágenes favorece más a los modelos entrenados.

- Aumento de todas las imágenes del conjunto inicial, obteniendo más cantidad de todas ellas.
- Aumento de únicamente aquellas imágenes que no contienen incendios, para balancear el conjunto inicial.

A la hora de hacer el aumento, los resultados no fueron exactos en cuanto a la cantidad de imágenes que se obtenían con lo realmente esperado. Sin embargo, la diferencia entre las cantidades obtenidas y deseadas era mínima, por lo que se utilizaron igualmente.

5.1.1. Aumento de todo el conjunto

Un primer aumento de los datos supone crear más imágenes de ambos tipos. Para ello, por medio del uso de *data augmentation*, se crearon dos nuevas imágenes por cada imagen con incendio, y cuatro por cada imagen sin incendio. De esta forma, se intentó no crear un desbalanceo aun más grave que el existente en el conjunto inicial.

	Número inicial de imágenes	Aumento	Total
Incendio	420	806	1226
No Incendio	203	773	976
Total	623	1579	2202

Tabla. 5.1: Imágenes totales en el conjunto de datos tras aplicar *data augmentation*.

Tal y como se muestra en la Tabla 5.1, este aumento supone añadir 1579 nuevas imágenes al conjunto inicial, de forma que se pueda experimentar con esta nueva cantidad de imágenes y determinar si los modelos necesitan o no un número de datos mayor.

5.1.2. Aumento parcial del conjunto

Como segundo uso de *data augmentation*, se aumentan solamente aquellas imágenes que no contienen incendios para poder de esta forma reducir el desbalanceo existente, que puede provocar un bajo rendimiento de los modelos.

	Número inicial de imágenes	Aumento	Total
Incendio	420	0	420
No Incendio	203	197	400
Total	624	197	820

Tabla. 5.2: Imágenes totales en el conjunto de datos tras aplicar *data augmentation* parcialmente.

Por tanto, las imágenes con incendio no aumentan su cantidad, mientras que se

creará una nueva por cada imagen que no contiene incendio. En la Tabla 5.2 se especifica que se crean 197 nuevas imágenes, que hacen un total de 820 imágenes en el conjunto de datos.

5.2. Red convolucional creada desde cero

En la primera fase de la experimentación se comenzó trabajando con una red neuronal convolucional con la configuración especificada en la Sección 4.3.1.

Previamente a la realización de cualquier experimento, se ha de saber qué parámetros se pueden alterar. En cuanto a la estructura de la red, no se va a cambiar nada, sino que todo se centrará en el uso de diferentes interpoladores a la hora de reducir el tamaño de las imágenes del conjunto de datos a una resolución común, como ya quedó explicado en la Sección 4.2.

Los interpoladores que ofrece el paquete *cv2*, de los cuales se habla en la Sección 2.1.1, son cinco en total, y existe evidencia de que algunos de ellos no funcionan bien a la hora de reducir la resolución de una imagen. Sin embargo, a la hora de realizar los experimentos, el objetivo principal es que la red tenga un rendimiento máximo, por lo que aun sabiendo que existe un problema al emplear ciertos interpoladores, se realizan pruebas con todos ellos.

5.2.1. Sin *data augmentation*

En un primer experimento, se entrenó, validó y evaluó la red utilizando el conjunto de datos inicial, sin hacer uso de *data augmentation*.

	Pérdida	Exactitud	Precisión	Sensibilidad	F1
Vecino más cercano	0.55	87.0 %	90.5 %	90.5 %	90.5 %
Bilineal	0.45	87.0 %	92.5 %	88.0 %	90.2 %
Área	0.61	87.0 %	91.4 %	89.3 %	90.3 %
Bicúbico	0.47	91.1 %	92.9 %	94.0 %	93.4 %
Lanczos	0.36	91.9 %	96.2 %	91.7 %	93.9 %

Tabla. 5.3: Resultados de la experimentación sin realizar *data augmentation*.

Como se observa en la Tabla 5.3, los mejores resultados de las métricas que se están evaluando se obtienen al utilizar los interpoladores Bicúbico y Lanczos. Para

analizar cada uno de ellos con más detenimiento es útil visualizar las matrices de confusión que generan.

		Valor predicho	
		No incendio	Incendio
Valor real	No incendio	34	6
	Incendio	5	79

Tabla. 5.4: Matriz de confusión para el interpolador Bicúbico.

		Valor predicho	
		No incendio	Incendio
Valor real	No incendio	37	3
	Incendio	7	77

Tabla. 5.5: Matriz de confusión obtenida al utilizar el interpolador Lanczos.

Tal y como queda reflejado en las Tablas 5.4 y 5.5, las redes clasifican incorrectamente 11 y 10 imágenes respectivamente. En el problema que se trata en este trabajo, es de vital importancia que se detecten los incendios, es decir, que los falsos negativos sean prácticamente inexistentes. Siguiendo este criterio, es más interesante una red que detecte la mayor cantidad de incendios cuando sí los hay. En este caso, la red a la que se le facilitó el conjunto de datos redimensionado con el interpolador Bicúbico detecta 79 incendios de un total de 84. Por otro lado, al utilizar el interpolador Lanczos se detectan 77 incendios correctamente.

Por tanto, si se quiere optar por una red que categorice correctamente el mayor número de elementos, la opción más adecuada sería la red que está entrenada con los datos preprocesados utilizando el interpolador Lanczos. Si se prefiere una red que detecte la mayor cantidad de incendios reales, entonces será ligeramente más recomendable hacer uso de la red entrenada con el conjunto de imágenes redimensionado con el interpolador Bicúbico. Aunque como se aprecia en la Tabla 5.3, los resultados son muy parecidos.

5.2.2. Con *data augmentation* sobre todas las imágenes

De la misma forma que se realizaron los experimentos en la Sección 5.2.1, se vuelven a ejecutar todos ellos, esta vez, haciendo uso del conjunto de datos aumentado con *data augmentation*.

	Pérdida	Exactitud	Precisión	Sensibilidad	F1
Vecino más cercano	0.49	93.0 %	90.8 %	97.1 %	93.8 %
Bilineal	0.73	93.0 %	90.5 %	97.6 %	93.9 %
Área	0.52	92.0 %	91.3 %	94.7 %	93.0 %
Bicúbico	0.72	93.4 %	92.6 %	95.9 %	94.2 %
Lanczos	0.35	93.6 %	92.5 %	96.3 %	94.4 %

Tabla. 5.6: Resultados de la experimentación haciendo uso de *data augmentation*.

Como se observa en la Tabla 5.6, tres de los interpoladores tienen alguna métrica que alcanza el máximo, por lo que para poder escoger entre ellos es aconsejable visualizar las matrices de confusión obtenidas al evaluar las redes.

		Valor predicho	
		No incendio	Incendio
Valor real	No incendio	170	25
	Incendio	6	239

Tabla. 5.7: Matriz de confusión para el interpolador Bilineal.

		Valor predicho	
		No incendio	Incendio
Valor real	No incendio	170	19
	Incendio	10	235

Tabla. 5.8: Matriz de confusión para el interpolador Bicúbico.

		Valor predicho	
		No incendio	Incendio
Valor real	No incendio	176	19
	Incendio	9	236

Tabla. 5.9: Matriz de confusión obtenida al utilizar el interpolador Lanczos.

En las Tablas 5.7, 5.8 y 5.9 se visualizan perfectamente cuántos son los elementos que cada red ha clasificado correctamente e incorrectamente. Tal y como se explicaba en la Sección 5.2.1, lo que más interesa a la hora de escoger una red en el problema que trata este trabajo sería la cantidad de imágenes que clasifica correctamente o el número de incendios que es capaz de detectar.

Para poder comparar todas las posibilidades que existen en este experimento más fácilmente, la Tabla 5.10 muestra únicamente la cantidad de imágenes que cada una de las redes está clasificando incorrectamente y el número de incendios que se detectan. El mejor interpolador a utilizar si buscamos que la red clasifique el mayor número

	Vecino más cercano	Bilineal	Área	Bicúbico	Lanczos
Datos clasificados incorrectamente	31	31	35	29	28
Incendios clasificados correctamente	238	239	232	235	236

Tabla. 5.10: Resumen de datos interesantes haciendo uso de *data augmentation*.

de imágenes en su grupo correspondiente sería Lanczos. En cualquier caso, si lo que buscamos es que detecte la mayor cantidad de incendios, el interpolador escogido sería Bilineal.

5.2.3. Con *data augmentation* sobre las imágenes sin incendios

Los experimentos realizados hasta el momento con una red neuronal creada desde cero son buenos, por lo que no sería necesario realizar más. La pregunta que queda por resolver es si el uso de *data augmentation* es imprescindible para poder entrenar las redes.

	Pérdida	Exactitud	Precisión	Sensibilidad	F1
Sin *data augmentation*	0.36	91.9 %	96.2 %	91.7 %	93.9 %
Con *data augmentation*	0.35	93.6 %	92.5 %	96.3 %	94.4 %

Tabla. 5.11: Mejores resultados obtenidos en ambas situaciones.

Comparando las mejores métricas de los resultados obtenidos, tal y como se observa en la Tabla 5.11, se puede analizar en qué difieren. Las diferencia en la precisión y sensibilidad es de un 3.7 % y un 4.6 % respectivamente, siendo el primer modelo más preciso, mientras que el segundo da como resultado menos falsos negativos. De nuevo, todo depende del criterio que se siga a la hora de escoger uno u otro. Dicho esto, la puntuación F1 solamente se diferencia en un 0.5 %, por lo que en este procedimiento en concreto y analizando el rendimiento global de los modelos, el uso de *data augmentation* no es estrictamente necesario para obtener un mejor sistema.

En la Sección 4.1.4 se especifica que existe un desbalanceo en cuanto a la cantidad de imágenes del conjunto de datos. Hay 203 imágenes que no contienen incendios frente a las 420 que sí que contienen. Por tanto, ya que el aumento de todos los datos no mejoró los resultados, se puede hacer una nueva experimentación aumentando únicamente las imágenes que no contienen incendios, de forma que se balancee el conjunto de datos.

	Pérdida	Exactitud	Precisión	Sensibilidad	F1
Vecino más cercano	0.39	88.4 %	100 %	77.4 %	87.2 %
Bilineal	0.70	85.4 %	100.0 %	71.4 %	83.3 %
Área	0.36	92.0 %	100.0 %	84.5 %	91.6 %
Bicúbico	0.34	87.2 %	100.0 %	75.0 %	85.7 %
Lanczos	0.27	89.6 %	92.4 %	86.9 %	89.6 %

Tabla. 5.12: Resultados de la experimentación haciendo uso de *data augmentation* parcialmente.

En esta fase de experimentación, se obtienen los resultados mostrados en la Tabla 5.12. De nuevo, es conveniente visualizar las matrices de confusión de los modelos que obtienen las mejores métricas.

		Valor predicho	
		No incendio	Incendio
Valor real	No incendio	80	0
	Incendio	13	71

Tabla. 5.13: Matriz de confusión para el interpolador de Área.

		Valor predicho	
		No incendio	Incendio
Valor real	No incendio	74	6
	Incendio	11	73

Tabla. 5.14: Matriz de confusión obtenida al utilizar el interpolador Lanczos.

En este caso, la red entrenada con el conjunto de datos preprocesado con el interpolador de Área solamente categoriza incorrectamente 13 elementos, como se observa en la Tabla 5.13, todos ellos siendo falsos negativos, que concretamente es el valor menos deseado. Sin embargo, esta red es la que menos error comete.

Por otro lado, utilizando el interpolador Lanczos, y tal como se muestra en la Tabla 5.14 la red resultante clasifica incorrectamente 17 elementos, pero solamente 11 serían falsos negativos. Por tanto, la elección de esta red o la nombrada anteriormente depende de cuál de estos valores se prefiere minimizar.

Si comparamos los resultados obtenidos en este experimento con aquellos que se produjeron al no hacer ningún tipo de aumentación de datos, se aprecia claramente que no hay ningún tipo de mejora. La Tabla 5.15 muestra claramente la bajada de la mayoría de las métricas. Esto indica que no es necesario realizar *data augmentation* sobre las imágenes sin incendios, mientras que este procedimiento sí que puede dar

	Pérdida	Exactitud	Precisión	Sensibilidad	F1
Sin data augmentation	0.36	91.9 %	96.2 %	91.7 %	93.9 %
Con data augmentation parcial	0.27	89.6 %	92.4 %	86.9 %	89.6 %

Tabla. 5.15: Mejores resultados obtenidos en ambas situaciones.

alguna mejora en cuanto a sensibilidad cuando lo realizamos sobre todas las imágenes del conjunto como se vio en la Sección 5.2.2.

5.3. *Transfer Learning*

Para la realización de este experimento, se escogieron tres redes pre entrenadas como base para el modelo final: *InceptionV3*, *ResNet50* y *Xception*.

Tras los experimentos realizados en la Sección 5.2 con los diferentes interpoladores, para las siguientes ejecuciones se utilizará únicamente el interpolador Lanczos, ya que destacó globalmente clasificando el mayor número de imágenes correctamente frente al resto en todas las pruebas realizadas.

En cuanto al uso de *data augmentation*, aunque quedó claro que no era necesario, se aplicará de igual forma por si hubiera posibles cambios con respecto a la red creada desde cero.

5.3.1. Sin *data augmentation*

	Pérdida	Exactitud	Precisión	Sensibilidad	F1
InceptionV3	17.4	78.2 %	76.6 %	97.6 %	85.9 %
ResNet50	0.51	96.8 %	97.6 %	97.6 %	97.6 %
Xception	16.23	83.9 %	84.0 %	94.0 %	88.8 %

Tabla. 5.16: Resultados de *transfer learning* sobre el conjunto de datos inicial.

Es evidente que el uso de una red pre entrenada en concreto ofrece mejores resultados que las otras dos. Esto queda perfectamente reflejado en la Tabla 5.16.

La red creada con *ResNet50* como base, cuya matriz de confusión se encuentra reflejada en la Tabla 5.17, muestra un rendimiento muy bueno, clasificando incorrectamente únicamente 4 imágenes, 2 de ellas siendo falsos negativos. Lo ideal es compa-

		Valor predicho	
		No incendio	Incendio
Valor real	No incendio	38	2
	Incendio	2	82

Tabla. 5.17: Matriz de confusión obtenida al utilizar la red pre entrenada *ResNet50*.

rar estos resultados con aquellos que se obtuvieron al entrenar una red creada desde cero.

	Pérdida	Exactitud	Precisión	Sensibilidad	F1
Red desde cero utilizando *Lanczos*	0.36	91.9 %	96.2 %	91.7 %	93.9 %
Tranfer learning con *ResNet50*	0.51	96.8 %	97.6 %	97.6 %	97.6 %

Tabla. 5.18: Comparación entre resultados de la red creada desde cero y el uso de *transfer learning*.

Teniendo en cuenta que el interpolador utilizado en este experimento para preprocesar las imágenes del conjunto es Lanczos lo idóneo es comparar estos resultados con los obtenidos en la Sección 5.2.1 con este interpolador en concreto. La Tabla 5.18 muestra todas las métricas relevantes para poder realizar una comparativa. Es indudable que el uso de *transfer learning* mejora los resultados obtenidos con anterioridad.

5.3.2. Con *data augmentation* sobre todas las imágenes

En la Sección 5.2.2 quedó demostrado que la aplicación de *data augmentation* sobre el conjunto de datos no mejoraba de forma global el rendimiento de las redes entrenadas. En este caso, al estar trabajando con redes que han sido pre entrenadas con una gran cantidad de imágenes, cabe la posibilidad de que sí que sea preciso aumentar el número de imágenes con el que se entrenan las últimas capas añadidas.

	Pérdida	Exactitud	Precisión	Sensibilidad	F1
InceptionV3	5.45	76.4 %	73.0 %	91.4 %	81.2 %
ResNet50	0.10	99.0 %	100.0 %	98.4 %	99.2 %
Xception	10.46	83.0 %	77.6 %	97.6 %	86.4 %

Tabla. 5.19: Resultados de la experimentación haciendo uso de *data augmentation*.

De nuevo, se realiza el experimento con las tres redes pre entrenadas que se escogieron anteriormente. Los resultados, en la Tabla 5.19, muestran algo parecido a lo

que ocurría en el experimento en el que no se hizo uso de *data augmentation*, la red *ResNet50* mejora enormemente a las otras dos.

		Valor predicho	
		No incendio	Incendio
Valor real	No incendio	195	0
	Incendio	4	241

Tabla. 5.20: Matriz de confusión obtenida al utilizar la red pre entrenada *ResNet50*.

En concreto, la red que cuenta con *ResNet50* como base, clasifica erróneamente 4 imágenes de un total de 440, dato mostrado en la Tabla 5.20.

	Pérdida	Exactitud	Precisión	Sensibilidad	F1
Sin *data augmentation*	0.51	96.8 %	97.6 %	97.6 %	97.6 %
Con *data augmentation* sobre todo el conjunto	0.10	99.0 %	100.0 %	98.4 %	99.2 %

Tabla. 5.21: Comparación entre experimentos de *transfer learning*.

La pregunta a responder vuelve a ser si es necesario aplicar *data augmentation* sobre el conjunto inicial. Si se presta atención a la puntuación F1, de la Tabla 5.21, que es una combinación de precisión y sensibilidad, al utilizar *data augmentation* se obtiene un 1.6 % más, por lo que depende del umbral que se tome en cuenta como un cambio significativo, se podría decir que el uso de este procedimiento sí que mejora los resultados obtenidos con anterioridad.

5.3.3. Con *data augmentation* sobre las imágenes que no contienen incendios

Una vez determinado que aumentar la cantidad de todas las imágenes del conjunto de datos no es preciso, se puede realizar otra experimentación, aumentando únicamente las imágenes del conjunto que no contienen incendios, tal y como se llevó a cabo en la Sección 5.2.3.

	Pérdida	Exactitud	Precisión	Sensibilidad	F1
InceptionV3	3.23	79.9 %	88.1 %	70.2 %	78.1 %
ResNet50	0.43	97.6 %	100.0 %	95.2 %	97.6 %
Xception	15.77	78.7 %	73.3 %	91.7 %	81.5 %

Tabla. 5.22: Resultados de la experimentación haciendo uso de *data augmentation* parcialmente.

El resultado obtenido en esta última prueba es bueno. En concreto, la red que tiene *ResNet50* como base, alcanza un 97.6 % de puntuacion F1, como se observa en la Tabla 5.22.

		Valor predicho	
		No incendio	Incendio
Valor real	No incendio	80	0
	Incendio	4	80

Tabla. 5.23: Matriz de confusión obtenida al utilizar la red pre entrenada *ResNet50*.

Para analizar más específicamente su rendimiento se puede visualizar su matriz de confusión en la Tabla 5.23. Este modelo comete únicamente 4 errores de un total de 164 posibles. Concretamente, se obtienen 4 falsos negativos, que es el fallo menos recomendable en el problema que se aborda.

	Pérdida	Exactitud	Precisión	Sensibilidad	F1
Sin *data augmentation*	0.51	96.8 %	97.6 %	97.6 %	97.6 %
Con *data augmentation* parcial	0.43	97.6 %	100.0 %	95.2 %	97.6 %

Tabla. 5.24: Comparación entre experimentos de *transfer learning*.

Viendo que este modelo es bueno, lo idóneo es compararlo con el modelo que mejor rendimiento presentaba sin realizar ningún tipo de *data augmentation*. En la Tabla 5.24 se observa, que en términos globales, realizar *data augmentation* de forma parcial no mejora el rendimiento de la red. Las puntuaciones F1 obtenidas en ambos casos son del 97.6 %.

Con el objetivo de visualizar todas las diferentes opciones en cuanto al modelo a escoger, la Tabla 5.25 recoge los resultados de los diferentes modelos. Se puede apreciar que el modelo que presenta los mejores resultados globales es aquel que se generó con el procedimiento *transfer learning*, empleando *ResNet50* como red neuronal base y el conjunto de datos ampliado con *data augmentation* sobre todas las imágenes. La diferencia entre la puntuación F1 de este modelo y la del resto hace que destaque y se pueda considerar el mejor sistema creado hasta el momento.

Para finalizar, conviene visualizar los resultados de una manera más gráfica que nos permita ver la diferencia entre los modelos. La figura 5.1 muestra las métricas de todos los modelos comparados anteriormente. Los modelos generados con *transfer learning* son claramente los que mejor rendimiento presentan, obteniéndose mejores resultados cuando se realiza *data augmentation*, alcanzando todas las métricas unos

		Pérdida	Exactitud	Precisión	Sensibilidad	F1
(1)	Sin *data augmentation*	0.36	91.9 %	96.2 %	91.7 %	93.9 %
	Con *data augmentation* sobre todos los datos	0.35	93.6 %	92.5 %	96.3 %	94.4 %
	Con *data augmentation* parcial	0.27	89.6 %	92.4 %	86.9 %	89.6 %
(2)	Sin *data augmentation*	0.51	96.8 %	97.6 %	97.6 %	97.6 %
	Con *data augmentation* sobre todos los datos	0.10	99.0 %	100.0 %	98.4 %	99.2 %
	Con *data augmentation* parcial	0.43	97.6 %	100.0 %	95.2 %	97.6 %

Tabla. 5.25: Comparativa de todos los modelos generados. (1)Red creada desde cero. (2) *Transfer Learning*.

valores muy cercanos al 100 %. En cuanto a las redes creadas desde cero, ninguna logra superar o igualar a los modelos de *transfer learning* en niguno de los tres casos estudiados.

5.4. *AutoKeras*

En el uso de esta herramienta, los parámetros que se pueden variar a la hora de realizar distintas ejecuciones son principalmente el número de intentos y las épocas. El número de intentos o *max_trials* indica cuántas configuraciones de red diferentes se van a probar y por tanto, a mayor cantidad, aumenta la probabilidad de encontrar el mejor modelo. En las ejecuciones de este procedimiento se fijaron estos parámetros a 10 intentos y 10 épocas.

Debido al rendimiento global del interpolador Lanczos en los experimentos realizados con la red neuronal creada desde cero, este es también el elegido para realizar el redimensionado de las imágenes previamente al entrenamiento de los modelos.

Ya que no existen más parámetros que puedan ser alterados, en las distintas ejecuciones de *Autokeras* simplemente se cambió el conjunto de datos con el que se

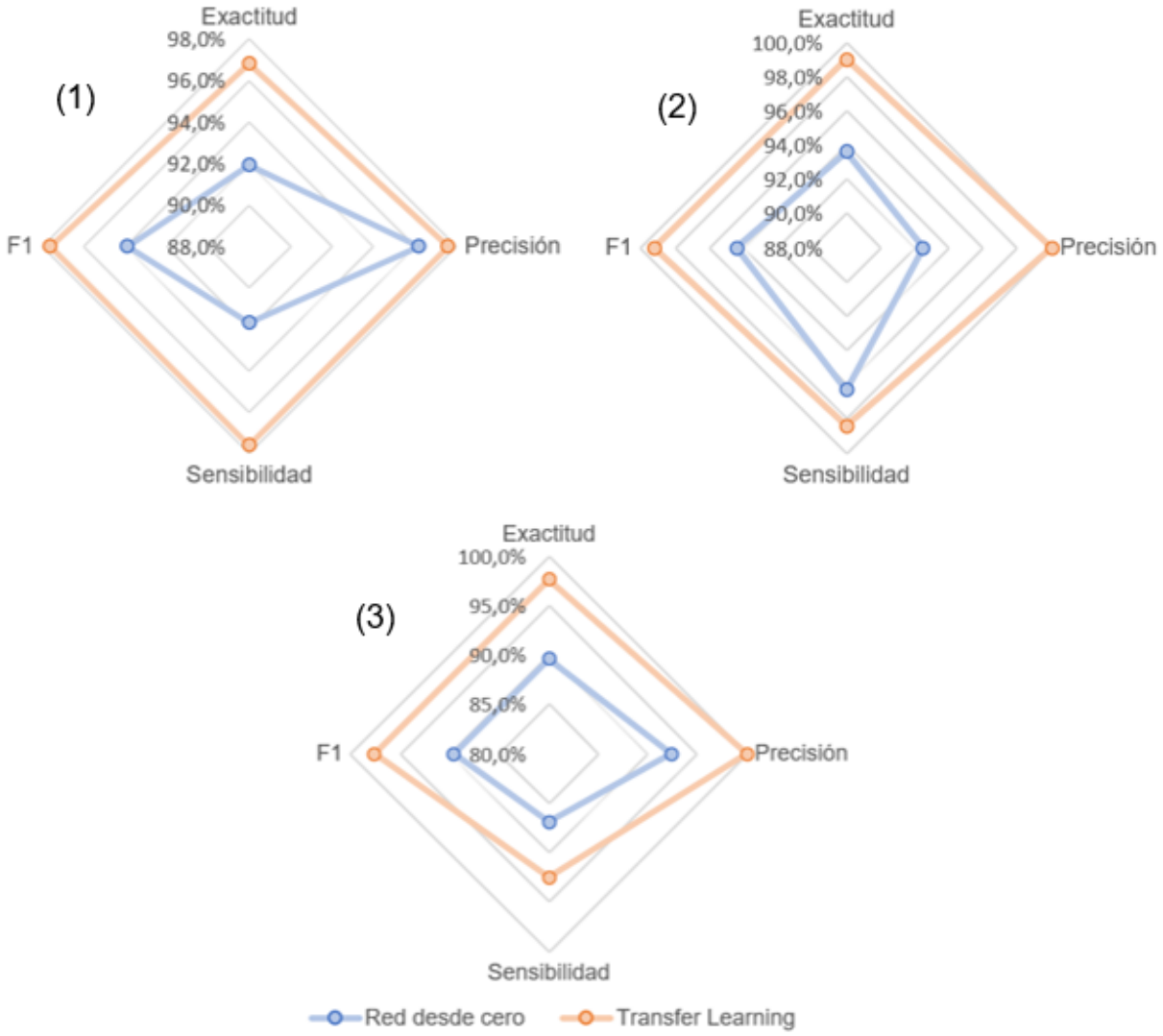

Figura 5.1: Métricas de los modelos generados hasta el momento. (1)Sin *data augmentation*. (2)Con *data augmentation* sobre todas las imágenes. (3)Con *data augmentation* sobre las imágenes sin incendios.

entrenan y evalúan los modelos entre los tres explicados en la Sección 5.1.

	Pérdida	Exactitud	Precisión	Sensibilidad	F1
Sin *data augmentation*	0.56	90.3 %	95.0 %	90.5 %	92.7 %
Con *data augmentation*	0.01	99.7 %	99.6 %	100.0 %	99.8 %
Con *data augmentation* parcial	0.18	93.3 %	98.7 %	88.1 %	93.1 %

Tabla. 5.26: Resultados del mejor modelo generado por *Autokeras* en cada uno de los casos.

Como se aprecia en la Tabla 5.26, los mejores resultados son ofrecidos por el modelo generado utilizando como conjunto de datos aquel con aumento sobre todas las

imágenes. La diferencia entre las puntuaciones F1 es suficientemente grande como para justificar que el uso de *data augmentation* es muy beneficioso y realmente necesario. La métrica que más llama la atención es la sensibilidad con un 100 %. Que se alcance este porcentaje implica que el modelo no comete ningún error en cuanto a falsos negativos, es decir, detecta todos los positivos que existen. A la hora de la detección de incendios, es indispensable que esto ocurra.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 192, 256, 3)]     0
_________________________________________________________________
cast_to_float32 (CastToFloat (None, 192, 256, 3)       0
_________________________________________________________________
normalization (Normalization (None, 192, 256, 3)       7
_________________________________________________________________
random_translation (RandomTr (None, 192, 256, 3)       0
_________________________________________________________________
random_flip (RandomFlip)     (None, 192, 256, 3)       0
_________________________________________________________________
resizing (Resizing)          (None, 224, 256, 3)       0
_________________________________________________________________
efficientnetb7 (Functional)  (None, None, None, 2560)  64097687
_________________________________________________________________
global_average_pooling2d (Gl (None, 2560)              0
_________________________________________________________________
dense (Dense)                (None, 1)                 2561
_________________________________________________________________
classification_head_1 (Activ (None, 1)                 0
=================================================================
Total params: 64,100,255
Trainable params: 63,789,521
Non-trainable params: 310,734
_________________________________________________________________
```

Figura 5.2: Configuración del mejor modelo generado por *Autokeras* con *data augmentation*.

En concreto, la estructura de la red que *AutoKeras* generó como el mejor modelo cuando se utiliza el conjunto de imágenes aumentado es el mostrado en la Figura 5.2

La matriz de confusión de este modelo, visualizada en la Tabla 5.27 muestra que el único error que se comete es un falso negativo de un total de 440 imágenes a clasificar.

Si se comparan estos resultados con los obtenidos en las secciones anteriores, se puede comprobar que el modelo generado por *AutoKeras* los mejora de forma global. Si atendemos a la puntuación F1, mostrada en la Tabla 5.28, el modelo que

		Valor predicho	
		No incendio	Incendio
Valor real	No incendio	194	1
	Incendio	0	245

Tabla. 5.27: Matriz de confusión para el mejor modelo generado por *Autokeras* sin *data augmentation*.

	Pérdida	Exactitud	Precisión	Sensibilidad	F1
Red creada desde cero sin *data augmentation*	0.36	91.9 %	96.2 %	91.7 %	93.9 %
ResNet50 con *data augmentation*	0.10	99.0 %	100.0 %	98.4 %	99.2 %
AutoKeras con *data augmentation*	0.01	99.7 %	99.6 %	100.0 %	99.8 %

Tabla. 5.28: Mejores resultados obtenidos en los distintos experimentos.

más se acerca es el generado con el procedimiento *transfer learning*, que también dio un resultado muy bueno. En el resto de métricas, también los supera, exceptuando la precisión, que llega a un 100 % en el modelo generado con *transfer learning*.

Por tanto, si buscamos el máximo rendimiento sin duda se puede llegar a él por medio del uso de *AutoKeras*, teniendo en cuenta que es necesario aplicar aumentación de datos. Si esto fuera un problema por carencia de recursos o de tiempo, el modelo escogido sería el obtenido a través de la red pre entrenada *ResNet50* sin hacer uso de *data augmentation*, que contaba con una puntuación F1 del 97.6 %.

Capítulo 6

CONCLUSIONES

Para finalizar esta memoria, en este capítulo se hará una recapitulación de los resultados obtenidos en la experimentación, analizando si se han llegado a cumplir los objetivos marcados al inicio y el grado de satisfacción con el trabajo realizado.

6.1. Conclusiones de los resultados

Las técnicas utilizadas para la resolución del problema han dado en general buenos resultados, teniendo los modelos generados una exactitud mínima del 89.6 %. Esto puede ser señal de que las características que muestra una imagen con incendio son muy claras y, por tanto, fáciles de aprender por una red neuronal convolucional.

El conjunto de datos inicial parecía tener pocas imágenes, insuficientes para poder entrenar, validar y evaluar una red neuronal. El uso de *data augmentation* ha resultado ser beneficioso en, únicamente, uno de los tres procedimientos utilizados, que casualmente es el que genera los mejores resultados globales. Por tanto, la elección de utilizar esta técnica o no, queda a elección del usuario.

En cuanto a la red neuronal que se creó desde cero, aunque dio unos resultados muy positivos, acabó siendo el procedimiento por el que se obtenían modelos con las métricas más bajas en general. Por medio de este método se podrían llegar a alcanzar resultados mucho mejores, pero esto supondría la búsqueda de un modelo óptimo, lo que requiere recursos computacionales y más concretamente, tiempo.

Los modelos obtenidos a través del uso de *transfer learning* mejoraron los resultados de la red creada desde cero, incluso en aquellos casos en los que no se utilizó

data augmentation. Concretamente, empleando *ResNet50* como red pre entrenada base para los modelos, se obtuvo una precisión del 100 %, valor que únicamente se alcanzó por medio de este método.

Sin duda, el mejor modelo fue el generado por *AutoKeras* utilizando como conjunto de datos aquel en el que se aplicó *data augmentation* sobre todas las imágenes. Además, con este modelo se obtuvo una sensibilidad del 100 %, lo que implica que no hubo ningún falso negativo, es decir, ni un solo incendio se quedó sin detectar.

Por tanto, si se decide realizar *data augmentation* sobre el conjunto de datos, lo ideal es utilizar *AutoKeras* como procedimiento para encontrar el sistema. En cambio, si no se decide hacer uso de esta técnica, la mejor opción recae sobre *transfer learning*.

6.2. Posibles mejoras

Es innegable que los resultados son muy buenos, pero siempre podrían mejorarse aún más teniendo en cuenta algunos aspectos como contar con un conjunto de datos de mayor tamaño o dedicar más tiempo a ajustar manualmente la red neuronal.

Como sistema final, se podría plantear la incorporación del modelo a los centros de vigilancia que obtienen las imágenes, siendo así más rápida la detección de incendios. Si además se optimiza el modelo para que ocupe lo menos posible, podría adaptarse a cámaras modernas que ya cuentan con capacidad de procesamiento o a una placa computadora como *Raspberry Pi* que, conectada a una cámara, pueda procesar las imágenes prácticamente a la vez que se toman. De cualquiera de estas formas se tendría un sistema autónomo capaz de alertar de un incendio activo.

6.3. Cumplimiento de objetivos

En cuanto al objetivo principal de este trabajo, sí se ha conseguido diseñar un procedimiento automático para la detección de incendios en áreas forestales. Durante el proceso de creación del mismo, se han estudiado las técnicas actuales de aprendizaje profundo.

Se ha realizado un estudio general de los campos de Ciencia de datos y Minería de datos para poder tener una idea más clara sobre el funcionamiento de las redes

neuronales convolucionales.

La búsqueda de un conjunto de datos adecuado fue exitosa, encontrando un grupo de imágenes que cumplía con las expectativas iniciales. Esto ha facilitado el trabajo a la hora de preprocesarlas y etiquetarlas correctamente para hacer uso de ellas en el entrenamiento, validación y evaluación de los sistemas creados.

La evaluación de los sistemas y comparación de resultados se ha realizado correctamente, pudiéndose comprobar en el Capítulo 5 de esta memoria.

Toda la información necesaria para entender y recrear los experimentos realizados se encuentra en esta memoria, realizada de la mejor manera posible para ser legible por cualquier persona que pueda ser ajena al campo estudiado.

6.4. Satisfacción personal

En general, y habiendo cumplido los objetivos definidos al inicio de este trabajo, mi grado de satisfacción con todo lo realizado es alto.

El estudio realizado previamente a la realización de cualquier tipo de experimento ha hecho que se haya adquirido conocimiento, que en un inicio era prácticamente nulo, acerca del campo de la Ciencia de datos y la Minería de datos. Este aprendizaje será, sin duda, una ventaja a la hora de comenzar la etapa profesional.

Por otra parte, teniendo en cuenta el problema real tratado en este trabajo, como son los incendios forestales, se ha logrado tener una mayor conciencia en cuanto al daño que genera el fuego y las consecuencias que puede ocasionar. Además, se ha visto la importancia de prevenir estos altercados en la medida de lo posible, ya que todos los incendios causados por humanos son, sin duda, evitables.

Bibliografía

W. Ali, S. M. Shamsuddin, and A. S. Ismail. Intelligent naïve bayes-based approaches for web proxy caching. *Knowledge-Based Systems*, 31:162–175, 07 2012. doi: 10.1016/j.knosys.2012.02.015.

R. Allison, J. Johnston, G. Craig, and S. Jennings. Airborne optical and thermal remote sensing for wildfire detection and monitoring. *Sensors (Basel, Switzerland)*, 16, 2016.

Arc. Convolutional neural network, 2018. URL https://towardsdatascience.com/convolutional-neural-network-17fb77e76c05. comprobado en 2021-07-17.

J. B. Introduction to perceptron: Neural network, 2017. URL https://blog.knoldus.com/introduction-to-perceptron-neural-network/. comprobado en 2021-07-17.

O. Bar-El. Getting the most out of your google colab, 2018. URL https://medium.com/@oribarel/getting-the-most-out-of-your-google-colab-2b0585f82403. comprobado en 2021-07-09.

J. I. Barrios Arce. La matriz de confusión y sus métricas, 2019. URL https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/. comprobado en 2021-07-14.

D. Berrar. Bayes' theorem and naive bayes classifier. In *Encyclopedia of Bioinformatics and Computational Biology*, 2019.

J. Brownlee. A gentle introduction to computer vision, 2019. URL https://machinelearningmastery.com/what-is-computer-vision/. comprobado en 2021-07-17.

A. Cetin, K. Dimitropoulos, B. Gouverneur, N. Grammalidis, O. Günay, Y. H. Habiboglu, B. U. Töreyin, and S. Verstockt. Video fire detection - review. *Digit. Signal Process.*, 23:1827–1843, 2013.

S. Chakravarty, S. Ghosh, C. P. Suresh, A. Dey, and G. Shukla. Deforestation: Causes, effects and control strategies. 2012.

W. K. Cho and S. Choi. Comparison of convolutional neural network models for determination of vocal fold normality in laryngoscopic images. *Journal of voice : official journal of the Voice Foundation*, 2020.

F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2017. doi: 10.1109/CVPR.2017.195.

J. Conejero and PTeam. Interpolation algorithms in pixinsight, 2018. URL https://pixinsight.com/doc/docs/InterpolationAlgorithms/InterpolationAlgorithms.html#__section001__. comprobado en 2021-07-16.

Daicy. Support vector machine, 2020. URL https://rpubs.com/daicy/SVM. comprobado en 2021-07-17.

G. Di Lallo. *Efficient, innovative, and inclusive options to overcome monitoring challenges in the early phases of REDD+*. PhD thesis, 01 2017.

El_Pais. El amazonas devorado por los incendios, en imágenes, 2019. URL https://elpais.com/elpais/2019/08/24/album/1566645226_292535.html#foto_gal_9. comprobado en 2021-07-12.

Gabri. ¿qué es la interpolación bilineal?, 2018. URL https://acolita.com/que-es-la-interpolacion-bilineal/. comprobado en 2021-07-16.

A. Gamaleldin, A. Atef, H. Saker, and A. Shasheen. Fire dataset, 2020. URL https://www.kaggle.com/phylake1337/fire-dataset. comprobado en 2021-05-25.

Google. Introduction to colab and python, 2018. URL https://colab.research.google.com/notebooks/intro.ipynb?utm_source=scs-index. comprobado en 2021-07-13.

G. Guo, H. Wang, D. Bell, Y. Bi, and K. R. C. Greer. Knn model-based approach in classification. In *OTM*, 2003.

D. Gupta. Fundamentals of deep learning – activation functions and when to use them?, 2020. URL https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/. comprobado en 2021-07-17.

P. Jadhav. Evaluation metrics for multi-label classification, 2021. URL https://medium.datadriveninvestor.com/a-survey-of-evaluation-metrics-for-multilabel-classification-bb16e8cd41cd. comprobado en 2021-07-16.

H. Jin, Q. Song, and X. Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '19, page 1946–1956, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330648. URL https://doi.org/10.1145/3292500.3330648.

A. Kaushik. Understanding resnet50 architecture, 2021. URL https://iq.opengenus.org/resnet50-architecture/. comprobado en 2021-07-10.

S. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica (Slovenia)*, 31:249–268, 2007.

M. Kubát. An introduction to machine learning. In *Springer International Publishing*, 2017.

V. Kurama. A review of popular deep learning architectures: Resnet, inceptionv3, and squeezenet, 2020. URL https://blog.paperspace.com/popular-deep-learning-architectures-resnet-inceptionv3-squeezenet/. comprobado en 2021-07-10.

L. Llamas. Explicación del teorema de muestreo de nyquist sin ecuaciones, 2020. URL https://www.luisllamas.es/explicacion-del-teorema-de-muestreo-de-nyquist-sin-ecuaciones/. comprobado en 2021-07-16.

L. Lourenço. Determination of forest fire causes in portugal (1996-2010). 2013.

N. O. Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. Velasco-Hernández, L. Krpalkova, D. Riordan, and J. Walsh. Deep learning vs. traditional computer vision. In *CVC*, 2019.

S. Mallick. Feature based image alignment using opencv (c++/python), 2018. URL https://learnopencv.com/feature-based-image-alignment-using-opencv-c-python/. comprobado en 2021-06-09.

A. Milton-Barker. Inception v3 deep convolutional architecture for classifying acute myeloid/lymphoblastic leukemia, 2019. URL https://software.intel.com/content/www/us/en/develop/articles/inception-v3-deep-convolutional-architecture-for-classifying-acute-myeloidlymphobla
html. comprobado en 2021-07-10.

K. Murphy. Machine learning - a probabilistic perspective. In *Adaptive computation and machine learning series*, 2012.

NumPy. The fundamental package for scientific computing with python, 2021. URL https://numpy.org/. comprobado en 2021-07-10.

OpenCV_team. Template matching, 2021a. URL https://docs.opencv.org/master/d4/dc6/tutorial_py_template_matching.html. comprobado en 2021-03-17.

OpenCV_team. opencv-python 4.5.2.54, 2021b. URL https://pypi.org/project/opencv-python/. comprobado en 2021-07-10.

S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191.

S. Preet. Python opencv – bicubic interpolation for resizing image, 2021. URL https://www.geeksforgeeks.org/python-opencv-bicubic-interpolation-for-resizing-image/. comprobado en 2021-07-16.

Python_Software_Foundation. Miscellaneous operating system interfaces, 2021a. URL https://docs.python.org/3/library/os.html. comprobado en 2021-07-10.

Python_Software_Foundation. Python plotting package, 2021b. URL https://pypi.org/project/matplotlib/. comprobado en 2021-07-10.

A. Rosebrock. Image difference with opencv and python, 2017. URL https://www.pyimagesearch.com/2017/06/19/image-difference-with-opencv-and-python/. comprobado en 2021-03-19.

S. Saha. A comprehensive guide to convolutional neural networks — the eli5 way, 2018. URL https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. comprobado en 2021-07-17.

R. Salas. Redes neuronales artificiales. *Universidad de Valparaıso. Departamento de Computación*, 1:1–7, 2004.

L. Salcedo. Introducción al machine learning 9 - k vecinos más cercanos (clasificación y regresión), 2018. URL https://pythondiario.com/2018/01/introduccion-al-machine-learning-9-k.html. comprobado en 2021-07-19.

O. Sanseviero. Introducción a tensorflow, 2019. URL https://medium.com/ai-learners/introducci%C3%B3n-a-tensorflow-parte-1-840c01881658. comprobado en 2021-07-10.

P. Santos, P. H. Ruas, J. Neves, P. Silva, S. Dias, L. Zárate, and M. Song. Implicpbdd: A new approach to extract proper implications set from high-dimension formal contexts using a binary decision diagram. *Journal of Information*, 9:50–57, 10 2018. doi: 10.3390/info9110266.

D. Soni. Introduction to bayesian networks, 2018. URL https://towardsdatascience.com/introduction-to-bayesian-networks-81031eeed94e. comprobado en 2021-07-17.

R. Szeliski. Computer vision - algorithms and applications. In *Texts in Computer Science*, 2011.

B. Tucker, M. Mantosh, N. Islam, and P. Everitt. Pycharm, 2021. URL https://www.jetbrains.com/es-es/pycharm/. comprobado en 2021-06-17.

A. Uberoi. Image registration using opencv | python, 2019. URL https://www.geeksforgeeks.org/image-registration-using-opencv-python/. comprobado en 2021-06-09.

S. Verma and S. Jayakumar. Impact of forest fire on physical, chemical and biological properties of soil: A review. 2012.

Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le. Unsupervised data augmentation for consistency training. *arXiv: Learning*, 2020.

C. Yuan, Y. Zhang, and Z. Liu. A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques. *Canadian Journal of Forest Research*, 45:783–792, 2015.