

**“Desarrollo de un modelo predictivo para la predicción de la presencia de tumores en imágenes de tomografía computarizada, aplicando técnicas de aprendizaje automático”.**

**Aplicación al cribado de pulmón.**



**Universitat  
Oberta  
de Catalunya**

---

**Beatriz Spa Gómez**

MU Bioinf. i Bioest.  
Bioinformàtica estadística y  
aprendizaje automático

**Nombre Tutor/a de TF**

Romina Astrid Rebrij

**Profesor/a responsable de  
la asignatura**

Carles Ventura Royo

**Fecha Entrega**

14-Enero-2024

## **C) Copyright**

© (2023-Beatriz Spa Gómez)

Reservados todos los derechos. Está prohibido la reproducción total o parcial de esta obra por cualquier medio o procedimiento, comprendidos la impresión, la reprografía, el microfilme, el tratamiento informático o cualquier otro sistema, así como la distribución de ejemplares mediante alquiler y préstamo, sin la autorización escrita del autor o de los límites que autorice la Ley de Propiedad Intelectual.

## FICHA DEL TRABAJO FINAL

|   |   |
|---|---|
| <b>Título del trabajo:</b>  | Desarrollo de modelos predictivos para la predicción de la presencia de tumores en imágenes de tomografía computarizada, aplicando técnicas de aprendizaje automático |
| <b>Nombre del autor:</b>  | <i>Beatriz Spa Gómez</i>  |
| <b>Nombre del consultor/a:</b>  | <i>Romina Astrid Rebrij</i>   |
| <b>Nombre del PRA:</b>  | <i>Carles Ventura Royo</i>  |
| <b>Fecha de entrega (mm/aaaa):</b>  | <i>01/2023</i>  |
| <b>Titulación o programa:</b>   | Master Universitario en Bioestadística y Bioinformática   |
| <b>Área del Trabajo Final:</b>  | Bioinformática estadística y aprendizaje automático   |
| <b>Idioma del trabajo:</b>  | <i>Castellano</i>   |
| <b>Palabras clave</b>   | <i>Redes neuronales, Oncología, Imagen médica</i>   |
| <b>Resumen del Trabajo</b>  |   |
| <p>La detección precoz, a través de los programas de cribado, es uno de los pilares fundamentales para el control de cáncer, y la aplicación del aprendizaje automático y su incorporación en distintos programas de cribado, puede mejorar su eficacia y disponibilidad. Como resultado del trabajo; se desarrolla un algoritmo, basado en redes neuronales, para evaluar distintas imágenes procedentes del cribado, con las que se determina si en ellas se encuentra la presencia de un tumor maligno, más concretamente, de pulmón. Se obtienen sensibilidades y especificidades en torno al 0.98 y 0.99, siendo la clase positiva 'Maligna'. Además, dicho algoritmo se implementa en una AppWeb con una interfaz sencilla y cuyo código se deja a disposición en un repositorio.</p> |   |
| <b>Abstract</b>   |   |
| <p>Early detection through screening programs is a fundamental pillar in cancer control. The application of machine learning techniques and their integration into various screening programs can improve their effectiveness and accessibility. That's why the aim of this project is to develop an algorithm based on neural networks to analyze various CT images to determine whether they indicate the</p>   |   |

presence of a malignant tumor, particularly in the lungs. Sensitivities of 0.98 and 0.99 are obtained, respectively, with the positive class being 'Malignant'. Moreover, said algorithm is implemented in a AppWeb and would be available in a repository.

# Índice

## Contenido

|   |          |
|---|----------|
| Lista de figuras.....   | 5        |
| <b>Introducción .....</b>   | <b>6</b> |
| 1.1. Contexto y justificación del Trabajo .....                   | 6        |
| 1.2. Objetivos del Trabajo.....                                   | 8        |
| 1.3. Impacto en sostenibilidad, ético-social y de diversidad..... | 8        |
| 1.4. Enfoque y método seguido .....                               | 9        |
| 1.5. Planificación del Trabajo.....                               | 12       |
| 1.6. Breve resumen de productos obtenidos.....                    | 17       |
| 1.7. Breve descripción de los otros capítulos de la memoria ..... | 18       |
| Estado del arte .....   | 19       |
| Materiales y métodos.....   | 21       |
| 3.1 Redes Neuronales Artificiales .....                           | 21       |
| 3.1.1 Neurona.....  | 21       |
| 3.1.2 Función de coste.....                                       | 22       |
| 3.1.3 Método de descenso por gradiente. ....                      | 22       |
| 3.2 Arquitectura de una red neuronal convolucional .....          | 23       |
| 3.2.1 Operación de convolución.....                               | 24       |
| 3.2.2 Capa de activación .....                                    | 24       |
| 3.2.3 Capa de agrupación.....                                     | 25       |
| 3.2.4 Capa totalmente conectada.....                              | 25       |
| 3.2.5 VGG.....  | 25       |
| 3.3 Implementación .....  | 26       |
| 3.3.1 Hardware y software utilizado .....                         | 26       |
| 3.3.2 Conjunto de datos. ....                                     | 27       |
| 3.3.3 Construcción del modelo .....                               | 28       |
| 3.3.4 Entrenamiento .....   | 32       |
| 3.3.5 Medición CO2.....   | 34       |
| 3.3.6 Implementación en App.....                                  | 34       |
| Resultados .....  | 36       |
| Conclusiones y trabajos futuros .....                             | 42       |
| Glosario .....  | 44       |

|                    |    |
|--------------------|----|
| Bibliografía ..... | 45 |
| Anexos .....       | 49 |

# Lista de figuras

|  |    |
|--|----|
| Figura 1. Algoritmos de aprendizaje supervisado [9].....   | 10 |
| Figura 2- Esquema de red neuronal convoucional para procesamiento de imágenes. [7]   | 10 |
| Figura 3-Extracto diagrama de Gantt, ver anexo “Diagrama de Gantt TFM”.....  | 15 |
| Figura 4- Captura de búsqueda de CNN en PubMed® .....  | 20 |
| Figura 5- Esquema de neurona artificial. [16].....   | 22 |
| Figura 6-Velocidades de aprendizaje y su influencia en la función de error [17].....   | 23 |
| Figura 7-Representación de aprendizaje.[18].....   | 24 |
| Figura 8- Representación del agrupamiento [19]. .....  | 25 |
| Figura 9- Descripción de las librerías principales utilizadas en aprendizaje profundo [10]   | 26 |
| Figura 10- Balance de clases del modelo, podemos ver el número correspondiente a las clases Benigno, Maligno y Normal antes y después de rebalancear. .... | 28 |
| Figura 11-Representación de las capas.....   | 31 |
| Figura 12-Gráficos del entrenamiento del modelo, precisión y pérdida del modelo según la época.....  | 33 |
| Figura 13- Herramienta NGROK .....   | 35 |
| Figura 14- Report de clasificación de la librería sklearn.....   | 36 |
| Figura 15- Gráfico representado la matriz de confusión.....  | 37 |
| Figura 16- Captura de la interfaz de la AppWeb generada. ....  | 38 |
| Figura 17- Captura de la interfaz de la AppWeb generada. ....  | 39 |

# Introducción

## 1.1. Contexto y justificación del Trabajo

La detección temprana de tumores es clave en el tratamiento oncológico. Se considera que la detección precoz (prevención secundaria) junto con las estrategias de promoción de la salud (prevención primaria) podría reducir los costes totales del cáncer en alrededor de 9000 millones de euros [1].

Los tumores más frecuentemente diagnosticados en el mundo son los de mama (que ocupan la primera posición) y pulmón con más de 2 millones de nuevos casos. El mayor número de fallecimientos por cáncer a nivel mundial se corresponde con el cáncer de pulmón (18,2% del total de muertes por cáncer). Se estima que, sin medidas preventivas adecuadas, se alcanzarán los 2.300.000 fallecimientos al año por cáncer de pulmón en el año 2030 [22].

La detección precoz, a través de los programas de cribado, es uno de los pilares fundamentales para el control de cáncer, que consisten en la realización de pruebas diagnósticas a personas, en principio sanas, pero que se encuentran en un grupo de riesgo, para detectar lesiones precancerosas y así intentar mejorar su pronóstico [2]. Con un buen método de diagnóstico se podrían salvar muchas vidas.

Tras que se hayan demostrados beneficios, tanto en la disminución del número de casos, como de la mortalidad, en cáncer de mama, colon y cérvix, se está empezando a obtener evidencia sobre su beneficio en el de cáncer de pulmón. La única prueba de detección recomendada para el cáncer de pulmón es la tomografía computarizada (TC) con dosis bajas, dirigido a aquellas personas que;

- Tengan un antecedente de consumo de cigarrillo de 20 años-paquete<sup>1</sup>, y
- Que fuman ahora o han abandonado el hábito en los últimos 15 años, y
- Tienen entre 50 y 80 años.

El desarrollo de un software basado en aprendizaje automático que facilite la tarea de evaluar todas las imágenes médicas resultantes de dicho cribado es, entre otras, de las principales aplicaciones que se quieren desarrollar en los próximos años, pues disminuye costos y aumentan la productividad de los profesionales de la salud. Se pueden aplicar

---

<sup>1</sup> El término año-paquete hace referencia al promedio de paquetes consumidos al día por año, pudiendo ser 20 años-paquete la resultante de fumar un paquete al día durante 20 años o dos paquetes al día durante 10 años.



varias técnicas de aprendizaje automático, mayormente basadas en redes neuronales, sobre imágenes médicas (CT de pulmón), para determinar un modelo que clasifique automáticamente según el resultado del cribado. [3]

En este proyecto se aplican varias técnicas de aprendizaje automático basadas en redes neuronales sobre imágenes médicas de cribado de CT, provenientes de una base de datos internacional, y se determina el modelo que permita predecir la prevalencia de la enfermedad de la manera más eficiente. Además, se implementa dicho modelo en una aplicación web y el código del modelo se deja almacenado en un repositorio de manera públic

## 1.2. Objetivos del Trabajo

### A. Objetivo general:

Desarrollar un algoritmo que permita predecir en base a una serie de imágenes CT en cuál de ellas existe un tumor de pulmón maligno.

### B. Objetivos específicos:

1. Optimizar el modelo seleccionado para lograr una precisión mínima del 75%.
2. Evaluar diversas técnicas de clasificación para determinar el mejor modelo predictivo, teniendo en cuenta el impacto en sostenibilidad.
3. Desarrollar una aplicación web basada en el modelo para su aplicación.

## 1.3. Impacto en sostenibilidad, ético-social y de diversidad

De acuerdo con los objetivos transversales de sostenibilidad, comportamiento ético y responsabilidad social, junto con diversidad y derechos humanos, se procede a una evaluación del contenido de este trabajo en dichos apartados.

- El resultado de este trabajo puede tener efecto en la huella ecológica, por el uso de servidores GPU, pudiendo ser reducido por ser implementado en Google Colab, que permite implementar técnicas para su medida y optimización [10]. El efecto negativo es inherente a la producción y uso de un software informático, quedando a responsabilidad del propietario la minimización del impacto. Los impactos positivos en la innovación y accesibilidad de este mismo trabajo, que pudiesen reducir dichas emisiones, no son a priori medibles, pero una razón inherente a este trabajo es la de permitir que, si la herramienta fuese exitosamente implementada, reducir con ello costes asociados del correspondiente programa de cribado, con su consecuente disminución en huella de carbono, uso de materiales, gastos económicos y de infraestructura etc, que repercutan en dicho objetivo de sostenibilidad.
- En lo referente al apartado ético y social, se procura obtener los datos de entrenamiento de una base de datos pública, que respete el Reglamento General de Protección de Datos (RGPD) y que dicha base de datos de datos no pueda producir sesgos en el modelo, escogiendo para ello un set de imágenes de pacientes con variabilidad de sexo, clase social, ideología y localización. Mención aparte, es conocida la discusión sobre que el uso de inteligencias artificiales (IA) potencialmente puede destruir puestos de trabajo, pero no se considera que la implementación de este algoritmo sea sustitutiva si no que sea complementario al trabajo del médico especialista en Radiología, mejorando la calidad y la

efectividad de la práctica diagnóstica. Se considera que un programa de cribado eficaz contribuye al bien común de la sociedad.

- Finalmente, se pretende que este algoritmo sea usado para mejorar los programas de cribado en cuanto a la parte diagnóstica, siendo la adherencia al mismo (o la falta de ella) un problema que no puede ser tratado de manera directa, pero que se puede incentivar al acortar los tiempos de respuesta. Un problema potencial es no conseguir una muestra suficientemente amplia al momento de entrenar el modelo, pero se pretenderá dejar el algoritmo suficientemente abierto para posibles revisiones y que se vaya completando la base de datos de entrenamiento. Es necesaria aun una normativa europea que regule el uso de modelos predictivos o redes neuronales en sanidad, que se ajuste a los derechos y valores europeos de supervisión humana, seguridad, privacidad y transparencia.

#### 1.4. Enfoque y método seguido

El aprendizaje automático en la actualidad es ampliamente utilizado en el campo de la salud, con el objetivo de procesar datos y obtener información relevante a partir de los mismos. Para el desarrollo del presente trabajo se utiliza la base de datos *"The IQ-OTH/NCCD lung cancer dataset"* (Mendeley data). Se trata de un conjunto de cortes de exploraciones CT a pacientes, extraído de "Hospital de Enseñanza de Oncología de Irak/Centro Nacional de Enfermedades Oncológicas (IQ-OTH/NCCD)" durante el último cuatrimestre de 2019, del Instituto Nacional del Cáncer de los Estados Unidos (National Cancer Institute, NCI) y otros colaboradores. La base de datos consta de 1190 imágenes, pertenecientes a 110 pacientes, agrupados en tres categorías; normal, benigno y maligno. [6]

Los escaners fueron originalmente obtenidos en formato DICOM, utilizando el protocolo estándar de CT de 120 kV, ancho de corte de 1 mm y un rango en Unidades Hounsfield de 350 a 1200. El modelo de escáner utilizado es SOMATON, de Siemens. Los casos varían en género, edad, educación y área de residencia, pudiendo obtener a partir de ello un modelo de aprendizaje robusto e inclusivo.

Existen varios algoritmos de aprendizaje supervisado [9] y sus características varían:

| Model                                 | Learning task      |
|---------------------------------------|--------------------|
| <b>Supervised Learning Algorithms</b> |                    |
| Nearest Neighbor                      | Classification     |
| Naive Bayes                           | Classification     |
| Decision Trees                        | Classification     |
| Classification Rule Learners          | Classification     |
| Linear Regression                     | Numeric prediction |
| Regression Trees                      | Numeric prediction |
| Model Trees                           | Numeric prediction |
| Neural Networks                       | Dual use           |
| Support Vector Machines               | Dual use           |

Figura 1. Algoritmos de aprendizaje supervisado [9]

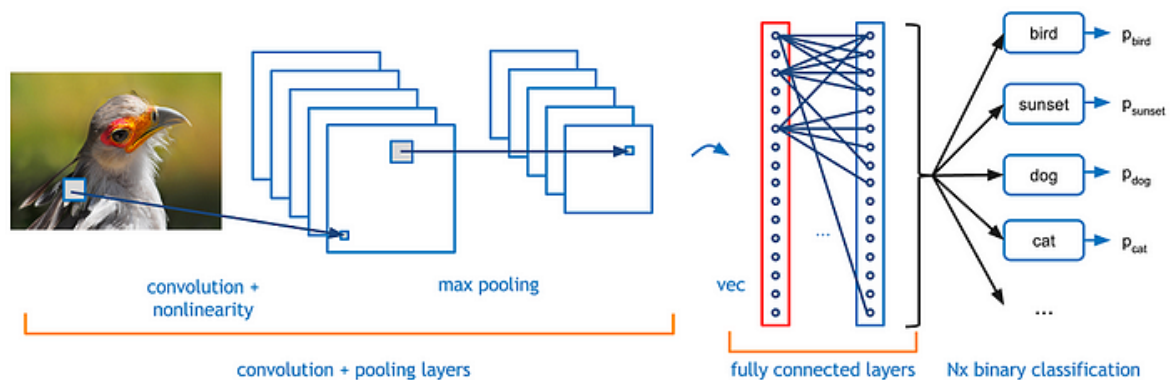


Figura 2- Esquema de red neuronal convolucional para procesamiento de imágenes. [7]

El objetivo es clasificar a los pacientes del screening, por lo que los algoritmos de clasificación basados en redes neuronales convolucionadas son los indicados para este caso, pues se caracterizan por resolver problemas de clasificación de imágenes. Para la implementación de este tipo de algoritmos en general se siguen los siguientes pasos [9]:

1. Recopilar datos.
2. Explorar y preprocesar datos.
3. Construir y entrenar un modelo en los datos.
4. Evaluar el rendimiento del modelo
5. Mejorar el rendimiento del modelo
6. Evaluar el modelo mejorado mediante distintos análisis
7. Implementarlo en una AppWeb

Al realizar un breve análisis exploratorio de los datos, se encontró que 40 casos fueron diagnosticados como malignos, 15 casos diagnosticados como benignos y 55 casos

clasificados como sanos.

Pueden darse varios problemas derivados de escoger una base de datos con más imágenes que datos. El principal de ellos es que vamos a tener imágenes altamente correlacionadas o redundantes, lo que produciría un sobreajuste en el modelo. Los métodos que se pueden utilizar para esto es un correcto preprocesado y selección de los datos, hacer particiones de los datos de entrada e introducir técnicas de regularización o penalización para evitar redes excesivamente complejas.

Otro problema derivado de esto es que nos reduce la cantidad real de casos que tenemos, pudiendo tener un conjunto de imágenes insuficiente y/o desbalanceado para tener la sensibilidad y especificidad que queremos. Los métodos que se pueden utilizar para paliar esto son técnicas de aumento de datos y/o transferencia de aprendizaje. El primero de estos métodos consiste en aplicar transformaciones aleatorias sobre los datos originales cada vez que se introduzcan en la red neuronal, con el fin de obtener muestras ligeramente diferentes pero que responden a la misma clasificación, de modo que permiten a la red desenvolverse mejor en la fase de inferencia. La transferencia de aprendizaje consiste en utilizar modelos pre-entrenados y reutilizarlos para la nueva tarea, solo modificando las últimas capas de la red neuronal para evitar el sobreajuste.

El lenguaje escogido para implementar el código es Python, implementándolo en GoogleColab, con el fin de que se cumpla el objetivo de accesibilidad y versatilidad. También es con Python como se crea la interfaz web sencilla con el modelo escogido, que puede hacerse mediante librerías como Streamlit u otros recursos, alojado en los servidores gratuitos de GitHub.

### 1.5. Planificación del Trabajo

Para realizar el proyecto, será necesario contar con un soporte para el código, que será escrito en Python.

Para poder realizar la planificación temporal se ha utilizado el programa GanttProjet 3.2, que permite realizar diagramas de Gantt.

#### A. Tareas:

| Descripción  | Fecha de inicio    | Fecha de fin       |
|--|--------------------|--------------------|
| <b>Definición del plan de trabajo. PEC1.</b>                       | <b>27/sep/2023</b> | <b>16/oct/2023</b> |
| Recopilación y análisis exploratorio de los datos.                 | 27/sep/2023        | 30/sep/2023        |
| Estudio de las técnicas a realizar y recopilación de bibliografía. | 1/oct/2023         | 3/oct/2023         |
| Definición de los modelos que serán aplicados.                     | 4/oct/2023         | 8/oct/2023         |
| Desarrollo del plan de trabajo. PEC1.                              | 9/oct/2023         | 12/oct/2023        |
| Entrega y retroalimentación del plan de trabajo.                   | 13/oct/2023        | 16/oct/2023        |

|  |                    |                    |
|--|--------------------|--------------------|
| <b>Desarrollo del trabajo. Fase 1. PEC2.</b>           | <b>17/oct/2023</b> | <b>20/nov/2023</b> |
| Exploración de las técnicas y herramientas necesarias. | 17/oct/2023        | 20/oct/2023        |

|   |               |               |
|---|---------------|---------------|
| Entender el formato de entrada de los datos que requiere cada modelo. | 21/oct/2023   | 23/oct/2023   |
| Preprocesamiento de datos.  | 24/oct/2023   | 25/oct/2023   |
| Construcción del modelo.  | 26/oct/2023   | 30/oct/2023   |
| Entrenamiento del modelo.   | 31/oct/2023   | 1/nov/2023    |
| Documentación de todo el trabajo realizado para la memoria.           | 2/nov/2023    | 2/ nov /2023  |
| Mejorar el rendimiento de los modelos construidos.                    | 3/ nov /2023  | 6/ nov /2023  |
| Validar el rendimiento de los modelos mejorados.                      | 7/ nov /2023  | 9/ nov /2023  |
| Optimizar esquema y presentación de la PEC 2.                         | 10/ nov /2023 | 14/ nov /2023 |
| Entrega y retroalimentación de la fase 1.                             | 15/ nov /2023 | 20/ nov /2023 |

|  |                    |                    |
|--|--------------------|--------------------|
| <b>Desarrollo del trabajo.<br/>Fase 2. PEC3.</b>       | <b>21/nov/2023</b> | <b>23/dic/2023</b> |
| Análisis de resultados.                                | 21/nov/2023        | 26/nov/2023        |
| Selección del modelo que alcance la precisión deseada. | 27/nov/2023        | 30/nov/2023        |
| Optimización del algoritmo.                            | 1/dic/2023         | 6/dic/2023         |
| Implementación del modelo en una aplicación web.       | 7/dic/2023         | 13/dic/2023        |
| Documentación y escritura de borrador de la memoria.   | 14/dic/2023        | 18/dic/2023        |
| Optimizar esquema y presentación de la PEC 3.          | 19/dic/2023        | 23/dic/2023        |

|   |                    |                    |
|---|--------------------|--------------------|
| <b>Cierre de la memoria.<br/>PEC4.</b>              | <b>27/dic/2023</b> | <b>14/ene/2024</b> |
| Optimizar esquema y presentación de la memoria.     | 27/dic/2023        | 4/ene/2024         |
| Optimizar esquema y presentación del código fuente. | 5/ene/2024         | 16/ene/2024        |

|   |                    |                    |
|---|--------------------|--------------------|
| <b>Elaboración de la presentación.<br/>PEC5a.</b> | <b>15/ene/2024</b> | <b>21/ene/2024</b> |
|---|--------------------|--------------------|



|                         |             |            |
|-------------------------|-------------|------------|
| Defensa pública. PEC5b. | 22/ene/2024 | 2/feb/2024 |
|-------------------------|-------------|------------|

B. Calendario:

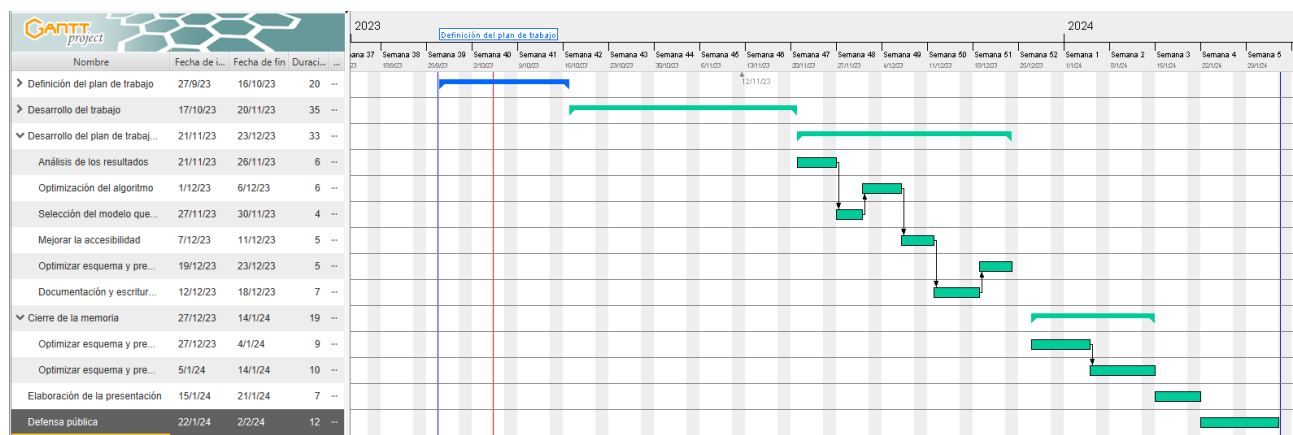


Figura 3-Extracto diagrama de Gantt, ver anexo “Diagrama de Gantt TFM”.

C. Hitos:

| Descripción                             | Fecha      |
|---|------------|
| Entrega del plan de trabajo.            | 16/10/2023 |
| Entrega desarrollo del trabajo. Fase 1. | 20/11/2023 |
| Entrega desarrollo del trabajo. Fase 2. | 23/12/2023 |
| Entrega de la memoria                   | 16/01/2024 |
| Entrega de la presentación.             | 16/01/2024 |
| Defensa pública.                        | 2/02/2024  |

D. Análisis de riesgos:

| <b>Descripción del riesgo</b>                      | <b>Severidad</b> | <b>Probabilidad</b> | <b>Mitigación</b>   |
|--|------------------|---------------------|---|
| Problemas con la importación de archivos dicom     | Alta             | Baja                | Investigar sobre importación de imágenes al soporte                       |
| Problemas con el pre procesamiento de las imágenes | Alta             | Baja                | Investigar técnicas de pre-procesamiento                                  |
| No alcanzar el rendimiento mínimo objetivo         | Alta             | Moderada            | Investigar sobre los parámetros que influyen en el rendimiento del modelo |
| No crear la Web App con conexión estable           | Moderada         | Moderada            | Plantear métodos para mejorar la portabilidad.                            |

## 1.6. Breve resumen de productos obtenidos

### A. Plan de trabajo:

Documento inicial en formato PDF que contenga las pautas y tiempos estimados de ejecución de todas las tareas necesarias para el desarrollo del trabajo y cumplimiento de objetivos.

### B. Memoria:

Documento en formato PDF que detalle toda la investigación, desarrollo, resultados y conclusiones obtenidas a lo largo del trabajo de fin de máster.

### C. Producto:

- Algoritmo de clasificación de presencia o no de tumores malignos en base a imágenes de CT. El código utilizado se compartirá en un repositorio público.

<https://github.com/BSGuoc/TFM/blob/main/C%C3%B3digo.ipynb>

- Aplicación web con el modelo implementado depositada en un repositorio.

Aplicación:

<https://t8lxkeg8wapdlrk3krfclz.streamlit.app/>

Repositorio del código de la AppWeb:

- Implementación de la App facilitada

<https://github.com/BSGuoc/TFM/blob/main/bsapp.py>

- Implementación local

<https://github.com/BSGuoc/TFM/blob/main/Programa.ipynb>

### D. Presentación virtual:

Presentación en formato PPT para exponer el trabajo realizado.

### 1.7. Breve descripción de los otros capítulos de la memoria

Se pretende con cada capítulo el siguiente objetivo;

- Materiales y métodos; se especificarán los métodos utilizados para el desarrollo del modelo, así como su implementación en la aplicación web, además de cómo se han ido paliando los problemas encontrados en el proceso.
- Resultados; principales resultados obtenidos, donde se detallará la eficacia del modelo, así como las distintas técnicas de evaluación del mismo.
- Conclusiones y trabajos futuros; se evaluará si los objetivos del trabajo se han conseguido, se hará la reflexión sobre si se han cumplido los objetivos transversales de comportamiento ético y global, se analizarán los defectos principales del TFM y se resumirán las principales líneas de tendencia del proyecto a futuros.
- Glosario; catálogo de términos o expresiones utilizadas a lo largo de la memoria.
- Bibliografía; se hará una descripción numerada de las principales fuentes de información para el desarrollo del trabajo.
- Anexos; relación de anexos que complementan la información que se desarrolla en el TFM.

## Estado del arte

Con este apartado se pretende exponer el estado actual del uso de redes neuronales en el diagnóstico temprano del cáncer de pulmón.

El cáncer de pulmón es una de las formas más comunes y mortales de cáncer a nivel mundial. Según la Agencia Internacional para la Investigación del Cáncer (IARC), es responsable de un alto porcentaje de las nuevas incidencias de cáncer y representa una carga significativa en términos de morbilidad y mortalidad. En 2020, se estimó que el cáncer de pulmón fue responsable de aproximadamente el 11.4% de todos los casos nuevos de cáncer y del 18% de todas las muertes por cáncer a nivel mundial. La Asociación Española Contra el Cáncer publicó en 2022 que en España se detectan 30000 casos y mueren a causa de esta enfermedad 23000 personas [22]. El cáncer de pulmón es la principal causa de muerte por cáncer. También explica que uno de los factores fundamentales que explica la cifra de supervivencia es que más del 70% se encuentra en fase avanzada de la enfermedad en el momento del diagnóstico.

Estudios como el realizado por Venkadesh, K.(2021) [20] han demostrado la eficacia de las CNN en la detección de enfermedades pulmonares, incluido el cáncer de pulmón, a partir de imágenes de TC. Utilizando grandes conjuntos de datos y técnicas de aprendizaje profundo, estos estudios han logrado resultados prometedores en la identificación temprana de anomalías.

El interés en las redes neuronales en el ámbito médico es creciente, relacionándose con tareas de clasificación, predicción, segmentación etc. Esto se refleja en la cantidad creciente de artículos relacionados con el tema que se pueden encontrar en buscadores de artículos médicos.

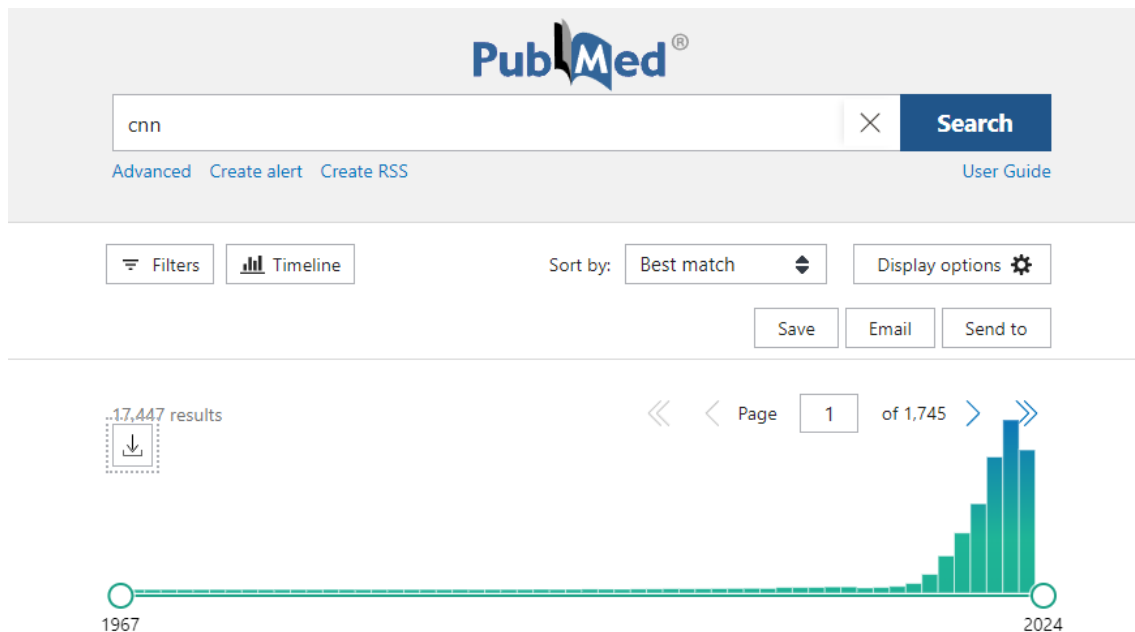


Figura 4- Captura de búsqueda de CNN en PubMed®

Más específicamente, la clasificación de imágenes médicas desempeña un papel esencial en los tratamientos clínicos y las tareas de enseñanza. Sin embargo, el método tradicional ha alcanzado su límite en rendimiento [11]. Además, al utilizarlo, se requiere mucho tiempo y esfuerzo para extraer y seleccionar características de clasificación.

Especialmente, la red neuronal convolucional domina con los mejores resultados en diversas tareas de clasificación de imágenes. Sin embargo, los conjuntos de datos de imágenes médicas son difíciles de recopilar porque se necesita mucha experiencia profesional para etiquetarlos. Los resultados de los experimentos muestran que la aumentación de datos generalmente es una forma efectiva para que las distintas redes mejoren su rendimiento.

La idea fundamental de utilizar una red neuronal convolucional en la clasificación de imágenes es que puede ser utilizado para mejorar el cribado de pulmón, u optimizarlo. Al realizar la imagen CT sobre los potenciales pacientes, la App donde está integrada la red neuronal puede avisar si hay altas posibilidades de existencia de una masa tumoral maligna, dándose prioridad sobre el resto de imágenes para que el médico especialista revise y confirme el diagnóstico, pues el tiempo en el desarrollo del cáncer juega un factor crítico y poder reducir el tiempo entre la adquisición de la imagen y la confirmación tumoral mejoraría enormemente el pronóstico de los pacientes. Existen líneas de investigación relacionadas con el tema [21] pero aún no están muy desarrolladas o implementadas.

# **Materiales y métodos**

En este apartado se pretende establecer una base teórica común sobre las redes neuronales artificiales y su funcionamiento, los elementos que las compone y su aplicación en el proyecto.

## **3.1 Redes Neuronales Artificiales**

Una red neuronal artificial es un modelo matemático y computacional, que proviene del Aprendizaje Profundo (Deep Learning) que se utiliza para relacionar entradas con salidas. Se compone de capas de unidades funcionales, llamadas neuronas, de manera similar a las redes neuronales biológicas, donde cada neurona capta los impulsos nerviosos que emiten otras neuronas, los procesan y los transmiten mediante un impulso nervioso hacia las contiguas.

### **3.1.1 Neurona**

Una red neuronal está formada por la interconexión de neuronas. La neurona se define como el elemento fundamental y más básico de una red neuronal. Su funcionamiento se basa en realizar cierta operación o procesamiento sobre dichas entradas y generar así su propia salida. Estos son los tres componentes principales de la neurona: las entradas, la operación que realiza sobre ellas y la salida, producto del procesamiento de las entradas. Una de las operaciones más comunes que realizan las neuronas es la función de activación, que impone un límite que se debe sobrepasar para poder proseguir a la neurona siguiente. Entrenar una red neuronal consiste en ajustar cada uno de los pesos de las entradas de todas las neuronas que forman parte de la red neuronal, para que las respuestas de la capa de salida se ajusten lo más posible a los datos que conocemos. Es la diferencia entre las predicciones y las salidas reales lo que define la función de coste. El objetivo fundamental del entrenamiento es minimizarla.

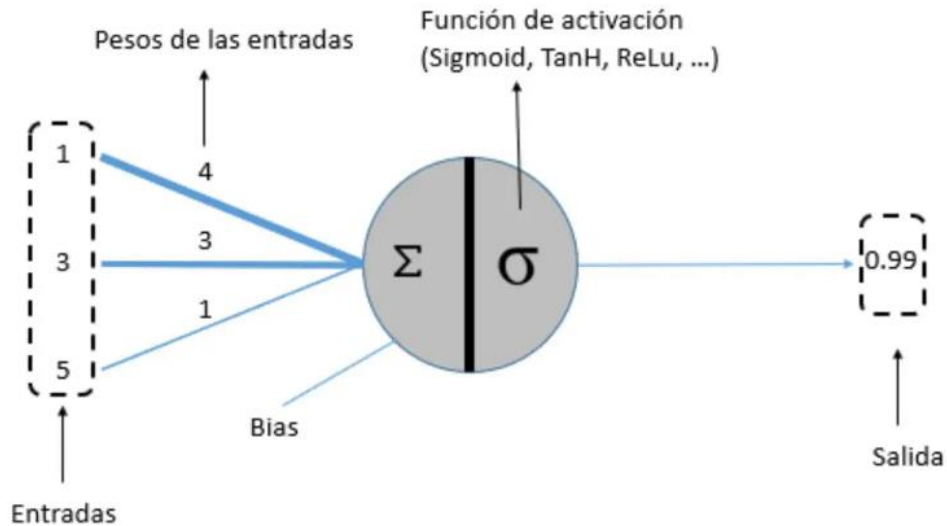


Figura 5- Esquema de neurona artificial. [16]

### 3.1.2 Función de coste

La función de coste determina el error que existe entre el valor de la predicción de nuestra red y el valor real esperado, para así optimizar los parámetros de la red. El proceso de optimización implica el uso de algoritmos como el descenso de gradiente (SGD) para ajustar los pesos y minimizar la función de costo pues es en función de esta diferencia se produce el ajuste de los pesos (de las funciones) de las neuronas.

Existen diferentes funciones de coste, escogiéndose en función de nuestro problema particular. La función de coste más utilizada en problemas de clasificación con es la función Entropía Cruzada, o en inglés Cross-Entropy, donde si es clasificación multiclase se considera clasificación categórica (no binaria).

### 3.1.3 Método de descenso por gradiente.

Después de identificar el error en la clasificación, la siguiente fase del algoritmo consiste en retroceder este error hacia atrás (backpropagation), desde la capa de salida hacia las capas ocultas de la red. Dicho error se propaga a todas las neuronas de la red que han influido en la clasificación del ejemplo, recibiendo cada una la “porción” del error correspondiente en función de su aportación. Esto provoca que se ajusten los pesos de manera proporcional, para que los nuevos valores reduzcan el error de clasificación. El



algoritmo empleado para esta optimización suele ser el de descenso del gradiente, que busca el mínimo para una función dada.

En lo relativo al entrenamiento de redes neuronales, consiste en encontrar el valor de los pesos y el sesgo que minimizan el valor de la función de coste. El algoritmo hace uso del gradiente para encontrar la dirección de descenso más pronunciada y moverse en el sentido negativo, que disminuye el valor de la función. Aquí entra en función el hiperparámetro factor o velocidad de aprendizaje, en inglés learning rate, que define la magnitud con la que se realiza el cambio en los parámetros, pesos y sesgo, para cada iteración del algoritmo. No se debe escoger ni una velocidad de aprendizaje excesiva ni una muy reducida porque no llega al mínimo de optimización. Un valor muy pequeño puede llevar a una convergencia lenta y que no se llegue a los valores necesarios mientras que un valor muy grande puede causar divergencia, ya que se “salta” el mínimo de optimización como se puede ver en la figura 8.

La elección de una velocidad de aprendizaje adecuada a nuestro problema es crucial para un entrenamiento eficiente y existen técnicas para que se vaya ajustando conforme avanza el proceso de entrenamiento.

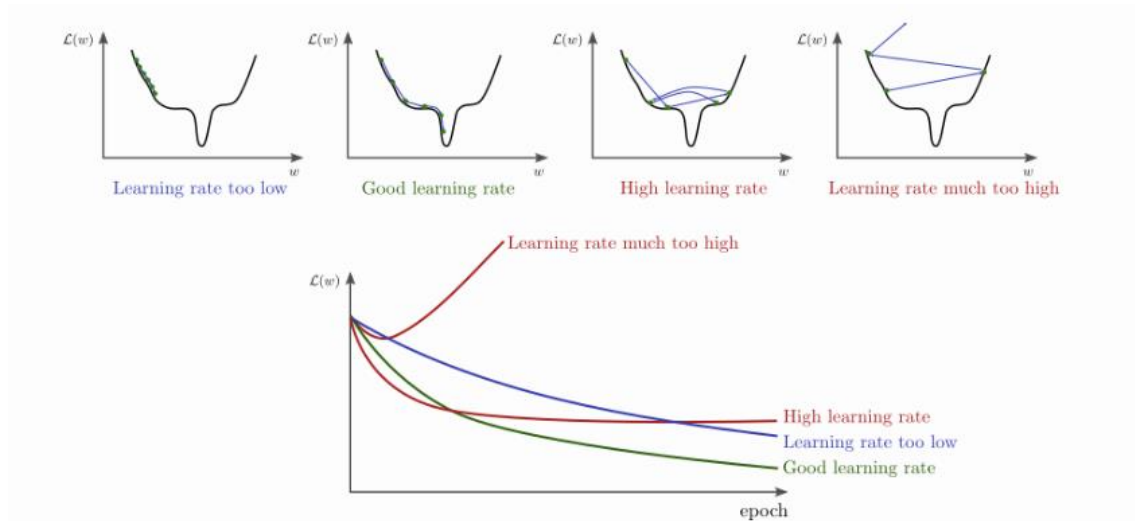
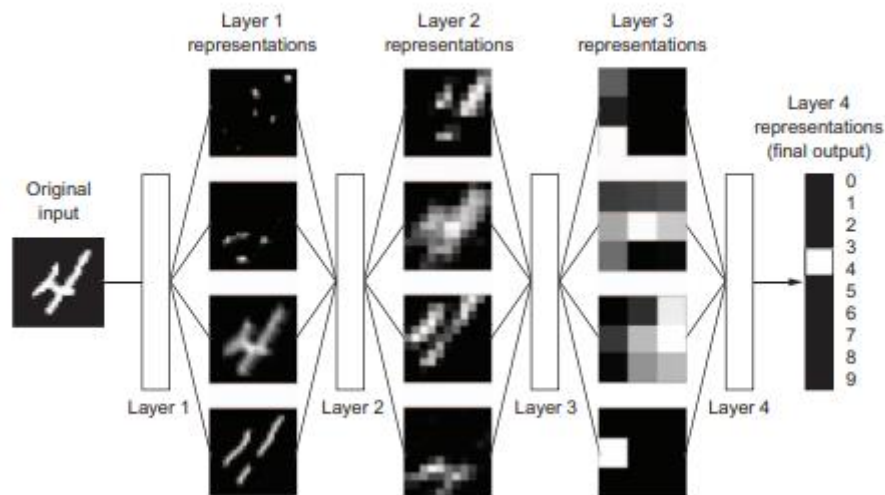


Figura 6-Velocidades de aprendizaje y su influencia en la función de error [17]

### 3.2 Arquitectura de una red neuronal convolucional

Las redes neuronales convolucionales (CNN) son un tipo especializado de red neuronal diseñadas para procesar datos de tipo cuadrícula, como imágenes. La arquitectura de una CNN se compone de varias capas, cada una con una función específica para aprender características jerárquicas de los datos de entrada. La arquitectura típica de

una red CNN es una sucesión de capas convolucionales y pooling, siendo habitual que la última capa sea una red neuronal totalmente conectada (FC), que se encarga de calcular las puntuaciones obtenidas por la imagen de entrada para cada una de las clases o categorías definidas en el problema.



*Figura 7-Representación de aprendizaje.[18]*

### 3.2.1 Operación de convolución

Consiste en aplicar un filtro (también llamado kernel) a pequeñas regiones de la entrada para extraer características locales. Mientras se desplaza el filtro por toda la entrada, se generan mapas de características que resaltan patrones específicos, como bordes, texturas, formas o colores.

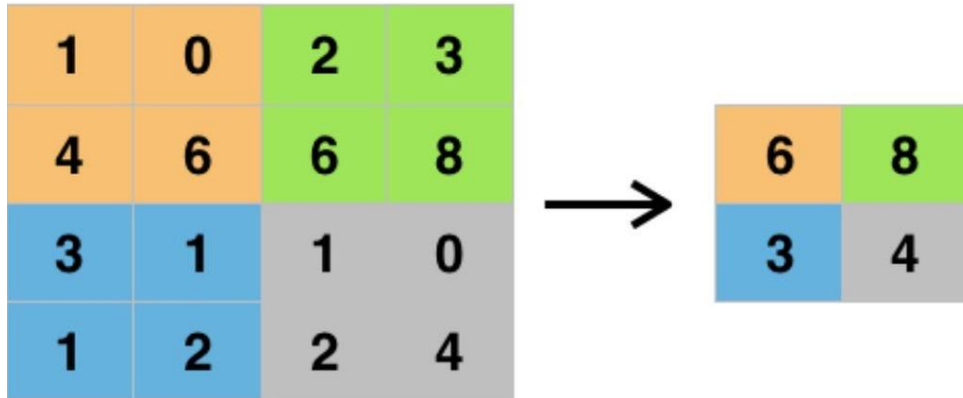
### 3.2.2 Capa de activación

Tras cada operación de convolución, se aplica una función de activación, comúnmente ReLU (Rectified Linear Unit). Esta función permite no linealidades en la red, permitiendo que esta aprenda patrones más complejos y no lineales.

$$f(x)=\max(0, x)$$

### 3.2.3 Capa de agrupación

Las capas de agrupación, también llamadas capas de pooling, se utilizan para reducir la dimensionalidad espacial de los mapas de características y, al mismo tiempo, conservar la información más relevante. El pooling suele realizarse mediante operaciones como el max pooling, que selecciona el valor máximo de un grupo de valores, o el average pooling, que calcula el promedio.



*Figura 8- Representación del agrupamiento [19].*

### 3.2.4 Capa totalmente conectada

También llamada Fully connected layer, conecta todas las neuronas de una capa con los elementos de la capa siguiente y se usa para clasificar las imágenes de diferentes categorías por entrenamiento, utilizando la función de activación Softmax

### 3.2.5 VGG

VGG (Visual Geometry Group) es una arquitectura de CNN propuesta por el grupo de investigación Visual Geometry Group de la Universidad de Oxford. Es conocida por su simplicidad y profundidad. La característica distintiva de VGG es que utiliza filtros convolucionales de tamaño muy pequeño (3x3) con capas de convolución en cascada, seguido de capas de agrupamiento (en inglés pooling). Esta estructura profunda pero simple ha demostrado ser efectiva en tareas de clasificación de imágenes.

### 3.3 Implementación

En este apartado se pretende mostrar el desarrollo de la implementación del modelo y entrenamiento, así como el procesamiento de los resultados.

#### 3.3.1 Hardware y software utilizado

Ambas implementaciones se han llevado a cabo en la plataforma Google Colab. Esta plataforma permite ejecutar código Python desde el navegador, además de acceso a GPUs, indispensable para tareas de entrenamiento de redes, debido al alto coste computacional que estas demandan.

Python es el lenguaje de programación más empleado en el ámbito del aprendizaje automático, contando con numerosas librerías; siendo las principales Pandas, Matplotlib, Numpy etc[10] [15]

| Nombre       | Funcionalidad principal   |
|--------------|---|
| Pandas       | Análisis y manipulación de estructuras de datos de alto nivel.  |
| Matplotlib   | Creación de visualizaciones estáticas, animadas e interactivas. |
| NumPy        | Computación de datos en forma de matrices multidimensionales.   |
| Scikit-learn | Procesamiento de datos y algoritmos de aprendizaje automático.  |
| TensorFlow   | <i>Deep learning.</i>   |

Figura 9- Descripción de las librerías principales utilizadas en aprendizaje profundo [10]

En la realización de este trabajo se ha utilizado una computadora modelo OMEN by HP 40L Gaming Desktop GT21-0xxx

Nombre del procesador: AMD Ryzen 7 5800X 8-Core Processor

Velocidad del procesador: 3,8 GHz

Cantidad de procesadores: 8

Memoria RAM: 16,0 GB

Gráficos: NVIDIA GeForce RTX 3070

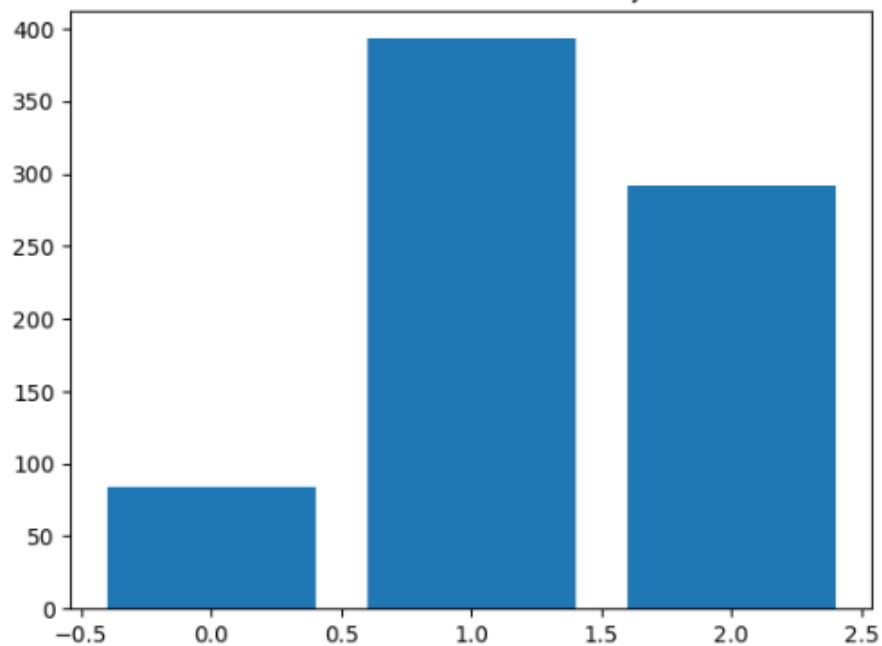
### 3.3.2 Conjunto de datos.

Con conjunto de datos se hace referencia a los datos con los que entrena la red. También se conoce como conjunto de ejemplos o de muestras, siendo también conocido como dataset.

Este dataset ha de ser dividido en tres conjuntos diferentes; el conjunto de entrenamiento con el que se entrena la red, el conjunto de validación, con el que se mide el rendimiento de la red, y finalmente el conjunto de prueba, que es el conjunto de datos con el que se mide el rendimiento real y final de la red.

El conjunto de datos escogido es “The IQ-OTH/NCCD lung cancer dataset” (Mendeley data) [6], que se compone de 1190 imágenes de cortes de CT. Se divide en un set de entrenamiento y uno de validación, cada uno de estos sets tiene 3 clases diferentes; casos benignos, casos malignos y casos normales.

Distribución de Clases antes del Aumento - Conjunto de Entrenamiento



Distribución de clases antes del aumento - Conjunto de entrenamiento

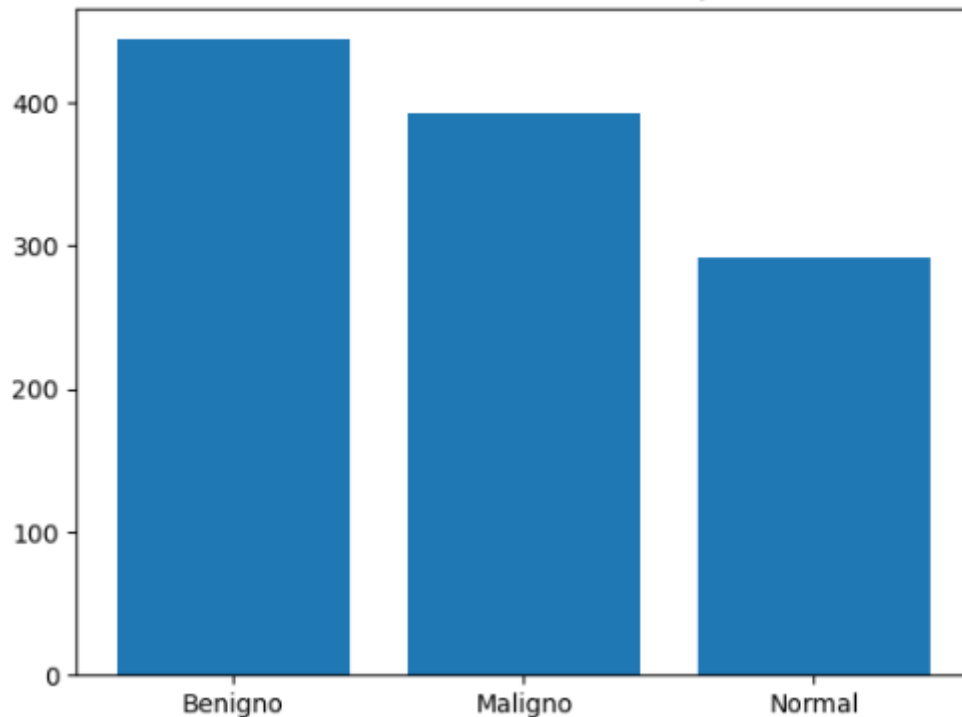


Figura 10- Balance de clases del modelo, podemos ver el número correspondiente a las clases Benigno, Maligno y Normal antes y después de balancear.

Originalmente tenemos en total 120 casos de clase Benigna, 561 de clase Maligno y 416 de clase Normal. Tenemos la clase Benigno desbalanceada frente al resto por lo que utilizamos una técnica de aumento de datos específica de la clase minoritaria, para que tengamos un caso más proporcionado, como se puede ver en la Figura 10. Se le ha aplicado una técnica de aumentado de datos mediante la función de Python ImageDataGenerator [18].

### 3.3.3 Construcción del modelo

Se ha construido el modelo basándose en una red neuronal tipo VGG16.

- Se añaden capas de activación tipo ReLU (acrónimo de Unidad Lineal Rectificada).

- Se añade dilución (más usado el término en inglés dropout) que es una técnica de regularización para reducir el sobreajuste en redes neuronales.

-Se añaden técnicas de regularización tipo L1 y L2. La regularización L1 y L2 son técnicas comunes en el aprendizaje automático para evitar el sobreajuste y mejorar la generalización de un modelo. Ambas técnicas penalizan los coeficientes del modelo y la elección entre L1 y L2 depende de los requisitos específicos del problema y las características de los datos. La regularización L1 es más adecuada cuando se tiene interés en la selección de características y la L2 si se busca una penalización suave en todos los coeficientes.

-Se añade en la última capa, una función de activación tipo softmax, para la clasificación categórica.

-Se añaden técnicas de parada temprana, (conocido también como EarlyStopping). La forma más común de implementar la parada temprana es monitorear la pérdida o alguna métrica relevante en el conjunto de validación durante el entrenamiento. Cuando la pérdida o la métrica deja de mejorar durante un número especificado de épocas consecutivas (llamado "patience"), el entrenamiento se detiene.

- Se añade también una forma de disminuir la velocidad de aprendizaje de la red conforme se va alcanzando una asíntota en la precisión.

- Se utiliza una red previamente entrenada (técnica de aprendizaje por transferencia). Haciendo uso de la librería Keras, se carga el modelo VGG-16 pre-entrenado con pesos del set "imagenet". Durante el entrenamiento "congelaremos" las primeras capas del modelo y sólo permitiremos el entrenamiento de las capas de salida de la clasificación.

|             |         |                       |
|-------------|---------|-----------------------|
| vgg16_input | input:  | [(None, 216, 216, 3)] |
| InputLayer  | output: | [(None, 216, 216, 3)] |



|            |         |                     |
|------------|---------|---------------------|
| vgg16      | input:  | (None, 216, 216, 3) |
| Functional | output: | (None, 6, 6, 512)   |



|          |         |                   |
|----------|---------|-------------------|
| conv2d_3 | input:  | (None, 6, 6, 512) |
| Conv2D   | output: | (None, 4, 4, 32)  |



|                       |         |                  |
|-----------------------|---------|------------------|
| batch_normalization_1 | input:  | (None, 4, 4, 32) |
| BatchNormalization    | output: | (None, 4, 4, 32) |



|              |         |                  |
|--------------|---------|------------------|
| activation_7 | input:  | (None, 4, 4, 32) |
| Activation   | output: | (None, 4, 4, 32) |



|                 |         |                  |
|-----------------|---------|------------------|
| max_pooling2d_3 | input:  | (None, 4, 4, 32) |
| MaxPooling2D    | output: | (None, 2, 2, 32) |



|          |         |                  |
|----------|---------|------------------|
| conv2d_4 | input:  | (None, 2, 2, 32) |
| Conv2D   | output: | (None, 2, 2, 64) |



|              |         |                  |
|--------------|---------|------------------|
| activation_8 | input:  | (None, 2, 2, 64) |
| Activation   | output: | (None, 2, 2, 64) |



|                 |         |                  |
|-----------------|---------|------------------|
| max_pooling2d_4 | input:  | (None, 2, 2, 64) |
| MaxPooling2D    | output: | (None, 1, 1, 64) |



|          |         |                   |
|----------|---------|-------------------|
| conv2d_5 | input:  | (None, 1, 1, 64)  |
| Conv2D   | output: | (None, 1, 1, 128) |



|              |         |                   |
|--------------|---------|-------------------|
| activation_9 | input:  | (None, 1, 1, 128) |
| Activation   | output: | (None, 1, 1, 128) |





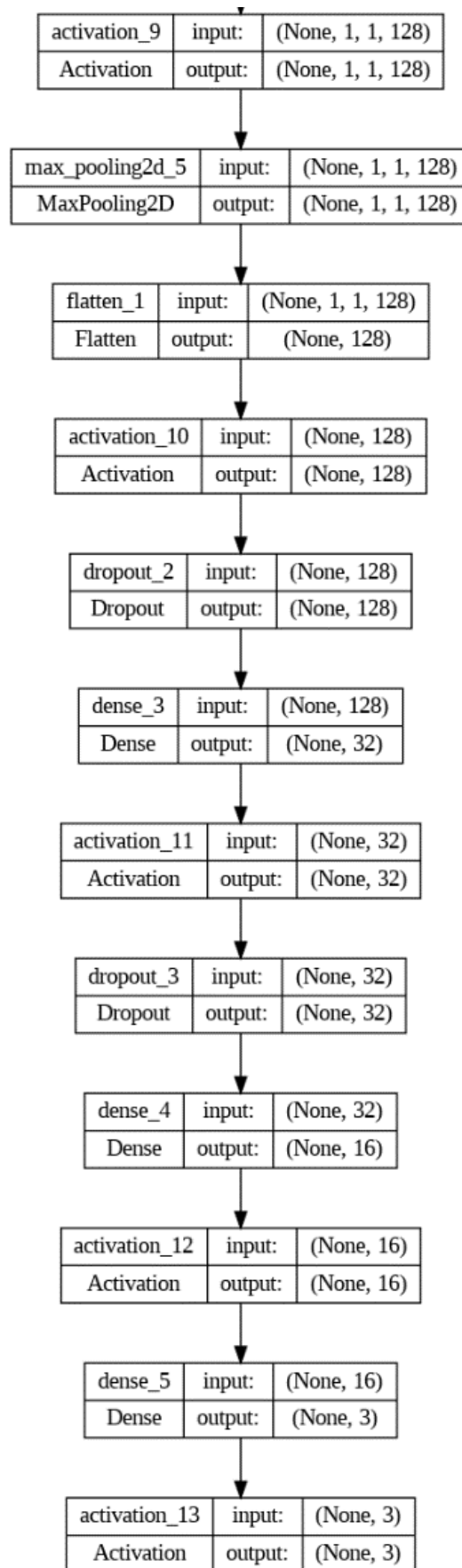
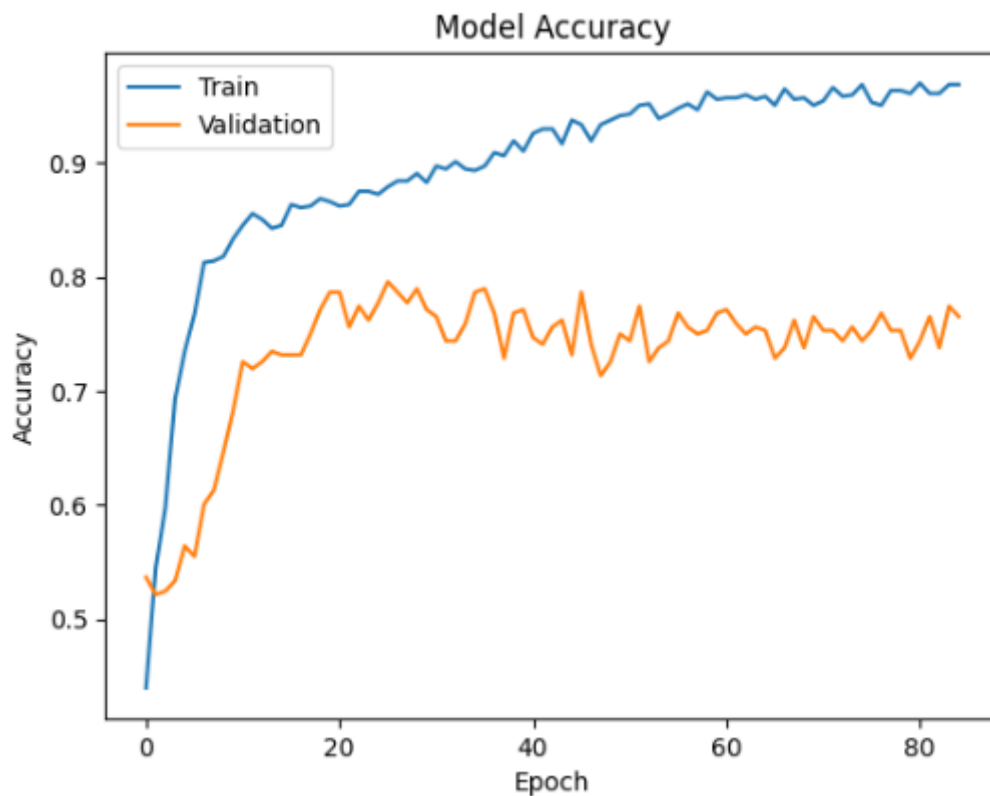


Figura 11-Representación de las capas.

### 3.3.4 Entrenamiento

Para llevar a cabo el entrenamiento del modelo se utilizan 100 épocas con un batch de entrenamiento de 12 imágenes, siendo esta configuración la que se encontraron óptimas para nuestro caso tras ensayo y error. Para el proceso de entrenamiento se utilizan técnicas de regularización de parada temprana y formas de reducir la velocidad conforme llega a un resultado asintótico.



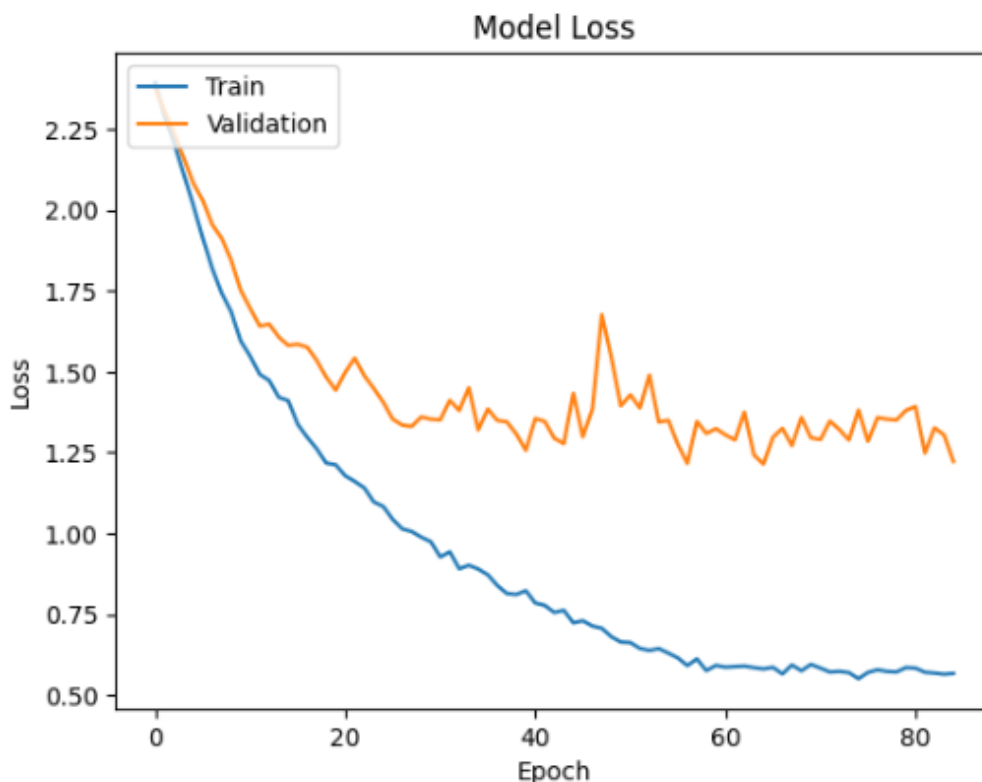


Figura 12-Gráficos del entrenamiento del modelo, precisión y pérdida del modelo según la época.

#### 3.3.4.1 Problema de rendimiento

El problema de rendimiento común a todos los entrenamientos que pueda tener una red neuronal artificial se conoce como sobreajuste u overfitting. Esto ocurre cuando la red se sobre ajusta a los datos con los que se ha entrenado, siendo incapaz de acertar y determinar la predicción correcta cuando se presenta un ejemplo que nunca ha visto, que esté fuera del conjunto de entrenamiento. Provoca un alto rendimiento con el conjunto de entrenamiento, pero un mal rendimiento con el conjunto de prueba. Para solucionar el problema del sobreajuste existen distintas técnicas que se han comentado ya, como la regularización, parada temprana, dilución y también existe la técnica del aumentado de datos.

#### 3.3.4.2 Aprendizaje por transferencia

El aprendizaje por transferencia es una técnica del Deep Learning que consiste en que, en lugar de entrenar un modelo desde cero, se reutiliza el conocimiento aprendido de tareas anteriores utilizando un modelo pre-entrenado, que es congelado para que no se puedan modificar sus pesos y al que se añaden capas personalizadas. Este es descargado a partir de una librería propia de Python; Keras. La gran ventaja del aprendizaje por transferencia es que se necesitan menos datos de entrenamiento y se mejora el rendimiento inicial.

Para lograr esto, tras importar la librería de Python Keras, se importa el modelo VGG16 [23]. Cuando se importa, a este modelo se puede indicar el formato de entrada de los datos que se va a implementar, que por defecto es formato RGB, y también los pesos con los que puede venir pre-entrenado. En este caso, se importa con unos pesos pre-entrenados a partir del set de imágenes ImageNet, propiedad de la Universidad de Princeton y Stanford [24].

### 3.3.5 Medición CO2

La medición de CO2 estimado en el uso del algoritmo es un objetivo específico del trabajo, con la idea de poder estimar el impacto medioambiental del proceso de creación e implementación del modelo.

Se implementa mediante CodeCarbon. CodeCarbon es una librería Python que permite rastrear y calcular las emisiones de carbono producidas por la nube, o por los recursos informáticos personales que hayan sido utilizados durante la ejecución de un determinado código en experimentos de aprendizaje automático. Para proporcionar dicha estimación, CodeCarbon tiene en cuenta estos parámetros:

- Infraestructura informática (GPU,CPU).
- Ubicación.
- Uso.
- Tiempo de ejecución.

El gasto de CO<sub>2</sub> varía en función de los anteriores parámetros estando en torno al equivalente de 0,08 kg, por cada vez que se ha entrenado el modelo.

### 3.3.6 Implementación en App

Se hace mediante la librería Streamlit, que genera la interfaz de usuario, para ello se descarga la librería en Python y se crea una aplicación sencilla en formato .py, que consiste en una página que pide cargar una imagen y al pulsar el botón de predecir, calcula a que clase pertenece (Benigno, Maligno o Normal)

Se implementa el modelo construido en un código fuente que será el que dé soporte a la App, cuya función es que solicite cargar una imagen, la muestre y después aplique el modelo CNN para predecir a que clase pertenece.

Ngrok es una herramienta que proporciona un túnel entre el servidor local y la nube; es lo que permite exponer el código, que está escrito de manera local en Streamlit, al público. Se utiliza para evitar implementar la aplicación en un servidor en la nube. Dicho túnel puede ser local o externo.

Tras crear la AppWeb, se deja en el código en GitHub, que es una plataforma para almacenar y gestionar código fuente, para que se pueda copiar la aplicación. Dicho código toma el modelo desde un enlace de Drive público, lo cual es más eficiente que volver a generarlo y se permite que el usuario la ejecute desde su servidor con su propia clave (token) de acceso.

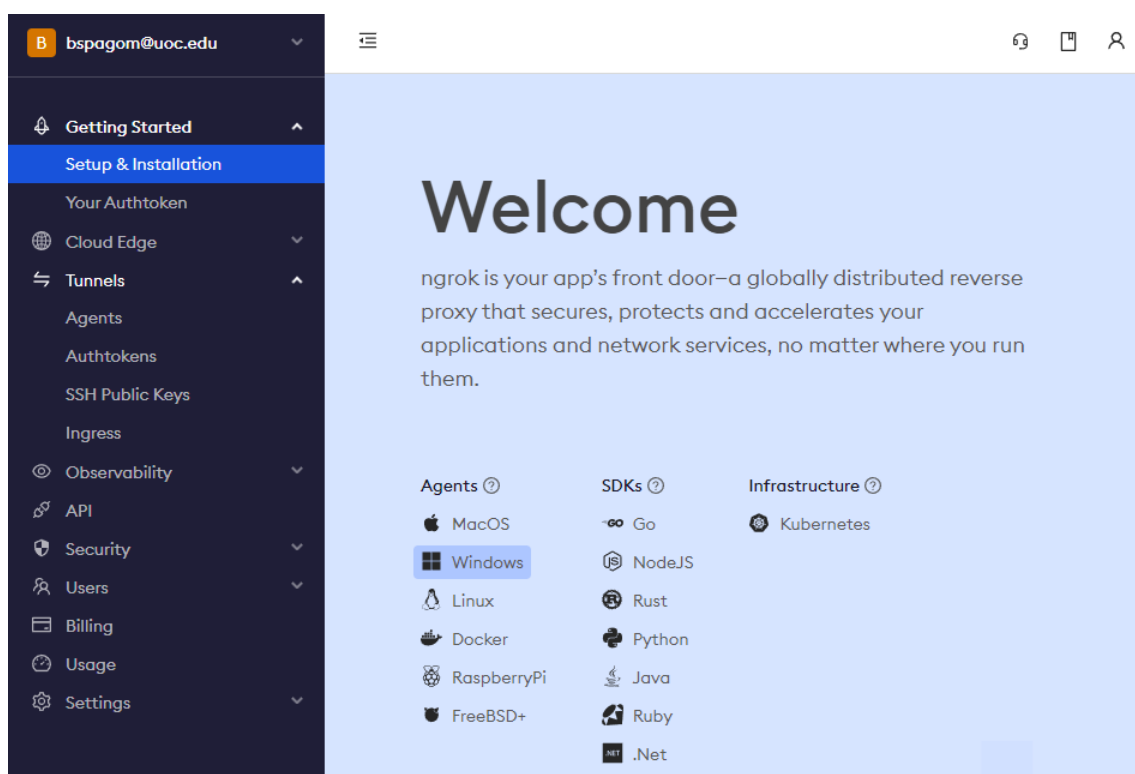


Figura 13- Herramienta NGROK

No se ha implementado directamente en GitHub Actions debido a que este soporte no admite aplicaciones dinámicas. Otros soportes más avanzados (Heroku) no cumplen el requisito de ser gratuitos o no necesitar registrarse, lo que impediría tener una accesibilidad suficiente.

También se implementa utilizando Streamlit.io, que toma el código de Github, que a su vez toma el modelo desde AWS (que es la nube propia de Amazon Web Services), que permite tener la AppWeb siempre accesible.

# Resultados

## 4.1 Resultados del modelo.

Tras entrenar el modelo se proponen varias métricas para evaluarlo, y cada modificación realizada tanto en la construcción como en el entrenamiento se pueden ir comprobando con lo gráficos generados.

El primer resultado a analizar es el report de clasificación, a través de una función de Sklearn (una de las librerías de Python utilizadas) se genera un report con las principales métricas de clasificación, mostrada para cada clase. La primera métrica es la precisión, que mide el número de imágenes acertadas dentro del total de imágenes clasificadas como de esa misma clase, es el cociente entre los verdaderos positivos y la suma de los identificados como positivos. Recall (en castellano sensibilidad) mide la fracción de verdaderos positivos entre el total de identificados como positivos. F1 score es una métrica que combina la precisión y la sensibilidad. Además, se ha calculado manualmente la sensibilidad y la especificidad tomando como única clase positiva la clase maligna.

```
Clasification Report

              precision    recall  f1-score   support

   Benigno      0.84        0.61      0.70         71
    Maligno      0.99        0.98      0.99        310
     Normal      0.86        0.95      0.90        219

 accuracy              0.93         600
 macro avg      0.90        0.85      0.87         600
weighted avg      0.93        0.93      0.92         600

Sensibilidad;0.9806451612903225
Especificidad; 0.993103448275862
```

Figura 14- Report de clasificación de la librería sklearn

Un gráfico que resulta muy útil para evaluar la clasificación es mediante una tabla a color de la matriz de confusión, que compara en una tabla de doble entrada las clases verdaderas con las predichas por el modelo.

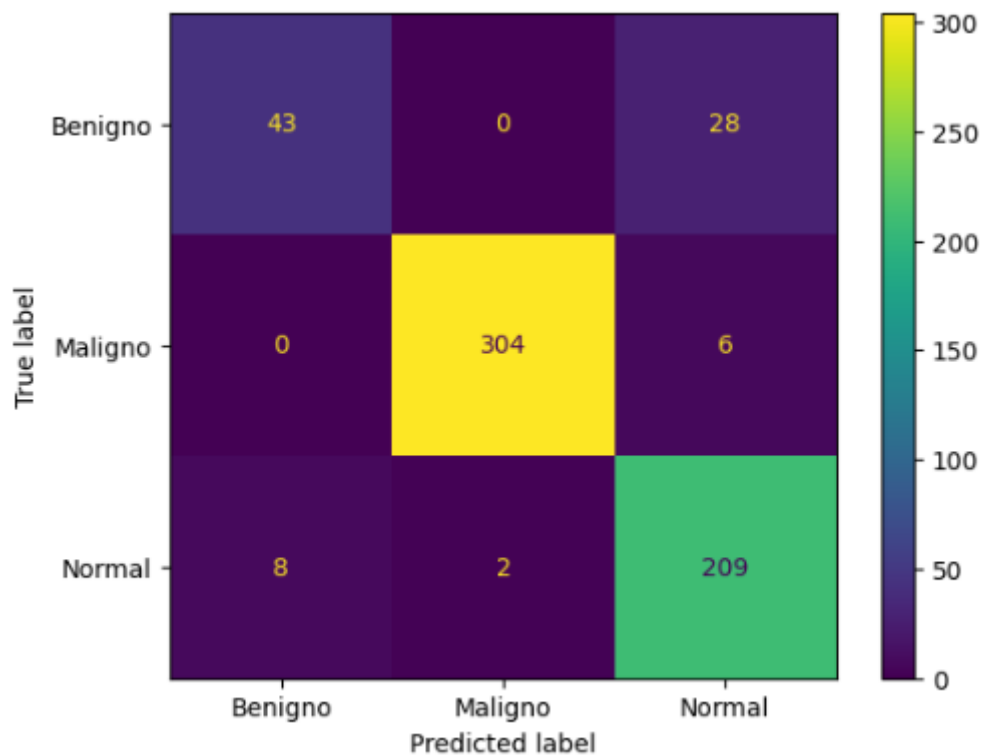


Figura 15- Gráfico representado la matriz de confusión

## 4.2 Resultados de la implementación web


Se genera la AppWeb mediante Streamlit con una interfaz sencilla, donde solamente hay que seguir las indicaciones de arrastrar la imagen a predecir y pulsar el botón para que se ejecute dicha predicción. El aspecto de la AppWeb es el de la figura 16.

## Predicción de imagen de CT

Con esta app se pretende predecir una imagen de CT de pulmón, con el fin de predecir si es un caso maligno o benigno

Solo tienes que cargar o arrastrar una imagen y pinchar en el botón de predecir que aparecerá abajo

Subir una imagen

 Drag and drop file here  
Limit 200MB per file • JPG, JPEG, PNG

Browse files

*Figura 16- Captura de la interfaz de la AppWeb generada.*





Malignant case (2).jpg 123.7KB



Imagen subida

Predecir

## Resultado de la predicción:

Clasificación: Maligno

Probabilidad estimada: 99.95%

*Figura 17- Captura de la interfaz de la AppWeb generada.*

Se deja el código en un repositorio público de GitHub con las instrucciones de uso.

Cuando se ejecuta, dicho código coge el modelo de la red neuronal desde un Drive y lo utiliza para crear un software que pide una imagen y realiza la predicción en base a dicho modelo. A continuación, guarda esta aplicación en un archivo .py.

Una vez se ha generado el archivo .py, hay que abrir un túnel Ngrok para que se muestre. Para ello realizamos las descargas necesarias y seguidamente se define la clave de seguridad "auth token", que se puede utilizar uno propio, sustituyendo 'NGROK\_AUTH\_TOKEN' o se puede utilizar el facilitado en este repositorio, aunque tiene una conexión más inestable. Por último, ejecutamos el archivo .py a la vez que se crea el túnel, generando la URL.

#### 4.3 Resultados de la medición de carbono e impacto en sostenibilidad, ético-social y diversidad.

- El resultado de este trabajo puede tener efecto en la huella ecológica, cosa que se intenta monitorizar con el uso de funciones propias de Python3. El efecto negativo es inherente a la producción y uso de un software informático, por lo que se pretende más el objetivo de optimizar la producción de CO<sub>2</sub> y que el gasto neto sea 0, es decir, que el uso de esta aplicación reduzca los gastos del propio programa de cribado. Haciendo uso de la librería de Python, codecarbon, con la herramienta EmissionTrack() realizamos una medida del gasto equivalente en CO<sub>2</sub> de 0,014 kg para el modelo final utilizado
- En lo referente al apartado ético y social, si el programa de cribado del cáncer de pulmón, se estima que se podría reducir en un 20% la mortalidad [26]. A pesar de no ocupar el primer puesto en incidencia en España, este tumor si es el más letal, dejando en España más de 20000 personas fallecidas, pues más de un 70% de los casos de pulmón se diagnostican en fase avanzada, por lo que se podrían estar salvando muchas vidas, cada vez más teniendo en cuenta que la incidencia va aumentando. El proyecto Cassandra, de la Sociedad Española de Neumología y Cirugía Torácica (SEPAR), está procurando desarrollar los protocolos de dicho programa [25].
- Finalmente, se pretende que este algoritmo sea usado para mejorar los programas de cribado en cuanto a la parte diagnóstica, dando importancia a que la aplicación pueda ser inclusiva y accesible para los usuarios, es por ello que se deja en un repositorio público y se intenta facilitar su uso.

- Se deja un contacto a disposición para poder recopilar opiniones y sugerencias de las partes interesadas para poder mejorar en las cuestiones éticas del proyecto.

# Conclusiones y trabajos futuros

En este trabajo se ha conseguido un algoritmo que realiza una clasificación de imágenes de cortes de CT con la siguiente precisión:

- 84% para la clasificación de la clase Benigno
- 99% para la clasificación de la clase Maligno
- 86% para la clasificación de la clase Normal

Estando por encima de la precisión del objetivo de precisión mínima del 75%. Esto se ha conseguido probando distintas combinaciones hasta encontrar una que sea un compromiso entre tiempo de clasificación y precisión, evitando, además, los problemas de sobreajuste. Además:

- Se han obtenido las competencias para la creación de una base de datos de imágenes y las técnicas de aumento de datos si se requieren.
- Se han conseguido competencias necesarias para la utilización de los paquetes Keras y TensorFlow.
- Se ha conseguido desarrollar competencias sobre implementación del algoritmo en una aplicación web mediante el paquete Streamlit.
- Se ha dejado el código en un repositorio público, garantizando la accesibilidad.
- También se ha conseguido estimar parte del coste medioambiental mediante el paquete codecarbon y la función EmissionTrack() [14].

Para líneas futuras de este trabajo estaría el conseguir una base de datos de pacientes más extensa, que pudiese reflejar más situaciones y alcanzar precisiones mejores, diseñar una interfaz de usuario más atractiva y fácil de usar e implementar en el código el manejo de errores y excepciones.

Como se ha visto, el uso de redes neuronales en programas de cribado tiene un gran potencial, quedando aún por abordar los desafíos éticos de diversidad, privacidad y accesibilidad. La implementación exitosa de dichas herramientas para la reducción de la mortalidad del cáncer de pulmón dependerá de la colaboración de distintos

profesionales; médicos, investigadores, ingenieros, políticos etc, teniendo como fin, el progreso.

# Glosario

TC: tomografía computarizada, de sus siglas en inglés

CNN: Red Neuronal Convolucionada, de sus siglas en inglés, convolutional neural network.

GPU: Graphics Processing Unit (unidad de procesamiento de gráficos)

ReLU: Rectified Linear Unit

RGB: Red Green Blue, modelo de color basado en la composición de los colores primarios de la luz

Overfitting: sobreajuste

Dropout; dilución

SGD: stocastic gradient dence

EarlyStopping: parada temprana

Deep Learning; aprendizaje profundo

RGPD: reglamento general de protección de datos

# Bibliografía

1. Asociación española contra el cancer (AECC)(2020)  
<https://blog.contraelcancer.es/cribado-cancer/>
2. El Impacto Económico y social del cáncer en España (2021) Consultores Estratégicos Internacionales. Available at:  
<https://www.oliverwyman.es/es/media-center/2020/feb/el-impacto-economico-y-social-del-cancer-en-espana.html> (Accessed: 29 September 2023).
3. Ruiz, Rorcio. and Velásquez, Juan. (2023) ‘Inteligencia artificial Al Servicio de la Salud del Futuro’, Revista Médica Clínica Las Condes, 34(1), pp. 84–91. doi:10.1016/j.rmcl.2022.12.001.
4. Centros para el Control y la Prevención de Enfermedades (CDC por sus siglas en ingles). Cancer de pulmón.  
[https://www.cdc.gov/spanish/cancer/lung/basic\\_info/screening.htm#:~:text=La%20C3%BAnica%20prueba%20de%20detecci%C3%B3n,que%20tienen%20un%20riesgo%20alto.](https://www.cdc.gov/spanish/cancer/lung/basic_info/screening.htm#:~:text=La%20C3%BAnica%20prueba%20de%20detecci%C3%B3n,que%20tienen%20un%20riesgo%20alto.)
5. McClintock, Matt. (2023) Clasificación de Imágenes Mediante Aprendizaje Automático: Los Detalles de la clasificación de imágenes, ProcessMaker. Available at: <https://www.processmaker.com/es/blog/image-classification-using-machine-learning-ins-and-outs/> (Accessed: 05 October 2023).
6. Alyasriy, Hamdalla; AL-Huseiny, Muayed (2023), “The IQ-OTH/NCCD lung cancer dataset”, Mendeley Data, V4, doi: 10.17632/bhmdr45bh2.4
7. Convolutional Neural Network. Towards Data Science. (2019).  
<https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529>

8. Rubio Camacho, Eduardo (2023) Redes neuronales convolucionales para la determinación de la profundidad del melanoma. TFG por la Universidad de Sevilla, Sevilla.
9. Lantz, Brett. (2015) *Machine learning with R: Discover how to build machine learning algorithms, prepare data, and dig deep into data prediction techniques with R*. Birmingham: Packt Publishing.
10. Sanchez Sanchez, Maria Isabel (2023) Herramientas para evaluar el impacto medioambiental generado por modelos de aprendizaje automático. TFG por la Universidad Rey Juan Carlos, Madrid.
11. Yadav, S.S., Jadhav, S.M. Deep convolutional neural network based medical image classification for disease diagnosis. J Big Data 6, 113 (2019). <https://doi.org/10.1186/s40537-019-0276-2>
12. "Colaboratory." <https://colab.research.google.com/?hl=es> (accessed Aug. 29, 2023).
13. Andrés Anaya-Isaza, Leonel Mera-Jiménez, Martha Zequera-Diaz, An overview of deep learning in medical imaging, Informatics in Medicine Unlocked, Volume 26, 2021, 100723, ISSN 2352-9148, <https://doi.org/10.1016/j.imu.2021.100723>.
14. Camilo Arenas, Juan. (2020). Codecarbon [Python]. <https://pypi.org/project/codecarbon/>
15. Adrián Ceja. ¿Por qué los Data Scientist utilizan Python? <https://www.neoland.es/blog/por-que-los-data-scientistutilizan-python>. Dic. de 2023.
16. Redes Neuronales artificiales: Qué son y cómo se entrenan es-ES. Sep. de 2019. uRI: <https://www.xeridia.com/blog/redes- neuronales- artificiales- queson-y- como-se-entrenan-parte-i> (visitado 03-02-2023).



17. Fei-Fei, Li. (Primavera 2023). CS231n: Convolutional Neural Networks for Visual Recognition. Stanford University. <http://cs231n.stanford.edu/>
  
18. Chollet, Francois. (2017). Deep learning with python. Manning Publications. ISBN 9781617294433
  
19. Yurek, Ilyurek. (2022, 12 de enero). What is Max Pooling and Why Do We Need Max Pooling? \*Medium\*. <https://medium.com/@ilyurek/what-is-max-pooling-and-why-do-we-need-max-pooling-57247a3fbca9>
  
20. Venkadesh, Kiran Vaidhya. et al. (2021) ‘Deep learning for malignancy risk estimation of pulmonary nodules detected at low-dose screening CT’, *Radiology*, 300(2), pp. 438–447. doi:10.1148/radiol.2021204433.
  
21. Pandian, Bala Murali. *et al.* (2022) ‘Detection and classification of lung cancer using CNN and google net’, *Measurement: Sensors*, 24, p. 100588. doi:10.1016/j.measen.2022.100588.
  
22. Asociación española contra el cancer (AECC)(2020) <https://www.contraelcancer.es/es/todo-sobre-cancer/tipos-cancer/cancer-pulmon/evolucion-cancer-pulmon>.
  
23. Simonyan, K. and Zisserman, A. (2015) *Very deep convolutional networks for large-scale image recognition*, *arXiv.org*. Available at: <https://arxiv.org/abs/1409.1556> (Accessed: 30 December 2023).
  
24. Deng, Jia. et al., 2009. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255.

25. Sociedad Española de Neumología y Cirugía Torácica (SEPAR). (2022). Cassandra: [Cribado de pulmón]. Recuperado de [http://proyectocassandra.com/proyecto.php]
26. US Preventive Services Task Force. Screening for Lung Cancer: US Preventive Services Task Force Recommendation Statement. *JAMA*. 2021;325(10):962–970. doi:10.1001/jama.2021.1117

## Anexos

Código generación del modelo, guardado del mismo y evaluación.

```
# Librerías importadas
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from google.colab import drive
from tabulate import tabulate
from keras import metrics
from keras import callbacks
from keras import losses
from keras import optimizers
from keras import models
from keras import layers
from keras import regularizers
from keras.utils import plot_model
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau, TensorBoard
import tensorflow as tf
!pip install codecarbon
import codecarbon
from codecarbon import EmissionsTracker
import time
from PIL import Image
import os,shutil
from collections import Counter
from sklearn.metrics import ConfusionMatrixDisplay,
classification_report,confusion_matrix
from sklearn.preprocessing import label_binarize
from sklearn.metrics._plot.roc_curve import roc_curve, auc
from collections import Counter
!pip install streamlit -q
!pip install streamlit-lottie
!pip install Pillow
from tf.keras.applications.vgg16 import preprocess_input

from tensorflow.keras.applications import VGG16
from sklearn.utils.class_weight import compute_class_weight
from tensorflow.keras import layers, models, optimizers,
regularizers

# Montaje en drive
```

```

drive.mount('/content/drive')

# Comando en entorno de ejecución, importamos las imagenes
!cp /content/drive/MyDrive/TFM/dataset.zip /content/
!unzip /content/dataset.zip -d /content/dataset/

tracker=EmissionsTracker()
tracker.start()
# Vemos una imagen de muestra de un caso benigno y otro maligno
ruta1="/content/dataset/dataset/Bengin cases/Bengin case (1).jpg"
imagen1=mpimg.imread(ruta1)
plt.title("Caso benigno")
plt.imshow(imagen1)

ruta2="/content/dataset/dataset/Malignant cases/Malignant case
(1).jpg"
imagen2=mpimg.imread(ruta2)
plt.title("Caso maligno")
plt.imshow(imagen2)

# Definimos el tamaño de la imagen
img_width, img_height = 216,216

# Tratamiento de imágenes para aumentado de muestra, creamos la
función train_datagen que modificará tamaños, reescalará y hará el
split.
# test
train_datagen = ImageDataGenerator(
    rescale=1./255,
    width_shift_range=0.25,
    height_shift_range=0.25,
    zoom_range=[0.8, 1.2],
    validation_split=0.3,
    fill_mode='nearest'
)

test_datagen = ImageDataGenerator(
    rescale=1./255,
)

# Split de imagenes de entrenamiento y validación
train_gen=train_datagen.flow_from_directory(directory="/content/dat
aset/dataset",
                                             class_mode="categorical"
,
                                             target_size=(img_width,i
mg_height),

```

```

        batch_size=40,
        subset="training")
    # color_mode="grayscale",

val_gen=train_datagen.flow_from_directory(directory="/content/dataset/dataset",

        class_mode="categorical",
        target_size=(img_width,img_height),
        subset="validation",
        batch_size=40)
        #color_mode="grayscale")

# Tras representar el número de clases del set de entrenamiento,
vemos que esta
# desbalanceado

# Verificar la antigua distribución de clases, se representa
unique_classes, class_counts = np.unique(train_gen.classes,
return_counts=True)
plt.bar(unique_classes, class_counts)
plt.title('Distribución de clases antes del aumento - Conjunto de
entrenamiento')
plt.show()

# Definir el generador de aumentos solo para la clase minoritaria
minority_datagen = ImageDataGenerator(
    rescale=1./255,
    width_shift_range=0.25,
    height_shift_range=0.25,
    zoom_range=[0.8, 1.2],
    fill_mode='nearest')

for i in range(0,3):
# Generador adicional solo para la clase minoritaria
    minority_gen = minority_datagen.flow_from_directory(
        directory="/content/dataset/dataset",
        class_mode="categorical",
        target_size=(img_width, img_height),
        batch_size=40,
        subset="training")

# Se concatena la clase minoritaria que es la clase Benigna
(classes=0)
    train_gen.classes = np.concatenate([train_gen.classes,
minority_gen.classes[minority_gen.classes==0]])

```

```

# Verificar la nueva distribución de clases
unique_classes, class_counts = np.unique(train_gen.classes,
return_counts=True)
plt.bar(unique_classes, class_counts)
plt.title('Distribución de Clases después del Aumento - Conjunto de
Entrenamiento')
plt.show()
# Las etiquetas corresponden a;
unique_classes=["Benigno","Maligno","Normal"]

# Visualización de la distribución de clases del conjunto de
entrenamiento frente al de validación
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.bar(unique_classes, class_counts)
plt.title('Distribución de Clases - Conjunto de Entrenamiento')

# Repite lo mismo para el conjunto de validación
unique_classes_val, class_counts_val = np.unique(val_gen.classes,
return_counts=True)
unique_classes_val=["Benigno","Maligno","Normal"]
plt.subplot(1, 2, 2)
plt.bar(unique_classes_val, class_counts_val)
plt.title('Distribución de Clases - Conjunto de Validación')
plt.xticks(unique_classes_val)
plt.show()

# Clasificación- CNN
# Comenzamos a definir el modelo, para ello primero importamos una
red VGG16
base_model = VGG16(weights='imagenet', include_top=False,
input_shape=(img_width, img_height, 3))

# Congelar capas preentrenadas
for layer in base_model.layers:
    layer.trainable = False

# Mediante .Sequential() definimos el modelo, que se llamará model
y comenzamos a añadir capas
model = models.Sequential()
model.add(base_model)

# Capa de convolución 1
model.add(layers.Conv2D(32, (3, 3), input_shape=(img_width,
img_height, 1), kernel_regularizer=regularizers.l2(0.001)))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2), padding='same'))

```

```

# Capa de convolución 2
model.add(layers.Conv2D(64, (3, 3),
kernel_regularizer=regularizers.l2(0.005), padding='same'))
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2), padding='same'))

# Capa de convolución 3
model.add(layers.Conv2D(128, (3, 3),
kernel_regularizer=regularizers.l2(0.005), padding='same'))
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2), padding='same'))

# Capa completamente conectada
model.add(layers.Flatten())
model.add(layers.Activation('relu'))
model.add(layers.Dropout(0.3))
model.add(layers.Dense(32,
kernel_regularizer=regularizers.l2(0.01)))
model.add(layers.Activation('relu'))
model.add(layers.Dropout(0.3))
model.add(layers.Dense(16,
kernel_regularizer=regularizers.l2(0.01)))
model.add(layers.Activation('relu'))
model.add(layers.Dense(3))
model.add(layers.Activation('softmax')) # Función de activación
softmax para problemas de clasificación binaria

model.summary()

# Crear una etiqueta de EmissionTracker, para la medida de CO2

tracker=EmissionsTracker()
tracker.start()

# Callback y optimizador

reduce_lr = ReduceLRonPlateau(monitor='accuracy', factor=0.2,
patience=5, min_lr=0.000001)

Callback =
callbacks.EarlyStopping(monitor='val_loss',patience=20,restore_best
_weights=True)

optimizer = optimizers.RMSprop(learning_rate = 0.0001)

NAME="Modelo CNN de CT"

```

```

tensorboard=TensorBoard(log_dir='logs/{}'.format(NAME))

# Como se observa que le modelo no clasifica correctamente la clase
"Benign, se opta
# por modificar los pesos en el proceso de entrenamiento

class_weights_dict = {0: 3.5, 1: 1.0, 2: 1.0}

# Compilación del modelo
model.compile(
optimizer=optimizer,
loss="categorical_crossentropy",
metrics=['accuracy'])

# Entrenamos el modelo
history = model.fit(train_gen,
epochs=100, steps_per_epoch=len(train_gen),
validation_data=val_gen,
validation_steps=len(val_gen),
callbacks=[tensorboard, Callback,
reduce_lr], verbose=2, class_weight=class_weights_dict)

model.save('/content/drive/MyDrive/TFM/modelo_FINAL.h5')
# Gráfica de función de coste y exactitud durante entrenamiento

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
#

# Para realizar el análisis usamos matrices de confusión, report de
clasificación y recuento de verdaderos positivos, verdaderos
negativos, sensibilidad y especificidad.

```



```

test_gen = test_datagen.flow_from_directory(
    directory="/content/dataset/dataset",
    target_size=(img_width,img_height),
    batch_size=600,
    class_mode='categorical',
    # color_mode="grayscale",
    )

x, y = test_gen.next()

predichos = []
pred_array = []

# Replica el único canal para emular imágenes en RGB
for i in range(600):
    img = x[i]
    img_rgb = np.repeat(img, 1, axis=-1)

# Cambiamos las dimensiones de la matriz para que coincidan con las
que requiere el modelo
    img_rgb = np.expand_dims(img_rgb, axis=0)

# Realiza la predicción
    test_predict = model.predict(img_rgb)
    predichos = np.argmax(test_predict,axis=1)
    pred_array.append(predichos)

pred_array=np.array(pred_array).flatten()

cell_dict_or = {0:"Benign", 1:"Malignant", 2:"Normal"}

clases=list(test_gen.class_indices.keys())

# Almacenamos los valores de la predicción y los reales
prediccion_f = {}
actual_val = {}

k=0
for arr in y[:600]:
    actual_val[k] = cell_dict_or[np.argmax(arr)]
    k+=1
k=0

for pred in pred_array:
    prediccion_f[k] = cell_dict_or[pred]
    k+=1

print("Valor Real:", actual_val)

```

```

print("Prediccion:", prediccion_f)

print('\n Confusion Matriz\n')
print(classes)
cm=confusion_matrix(list(actual_val.values()),
list(prediccion_f.values()))
disp=ConfusionMatrixDisplay(cm,display_labels=["Benigno","Maligno",
"Normal"])
disp.plot()
plt.show()

print('\nClasification Report\n')
report=classification_report(list(actual_val.values()),
list(prediccion_f.values()),target_names=["Benigno","Maligno","Norm
al"])
print(report)

# Verdadero positivo
vp=cm[1,1]
# Verdadero negativo
vn=cm[0,0]+cm[2,2]+cm[0,2]+cm[2,0]
# Falso positivo
fp=cm[0,1]+cm[2,1]
# Falso negativo
fn=cm[1,0]+cm[1,2]
# Sensibilidad
print(f'Sensibilidad; {vp/(vp+fn)}')
# Especificidad
print(f'Especificidad; {vn/(vn+fp)}')

emision: float=tracker.stop()
print(f"Emisiones estimadas de carbono: {emision} kg de CO2
equivalentes.")

```

```

# Ponemos Malignant como clase positiva para poder evaluar la curva
ROC
y_verdad = np.array([1 if v == 'Malignant' else 0 for v in
actual_val.values()])
y_predic = label_binarize([1 if v == 'Malignant' else 0 for v in
prediccion_f.values()], classes=[0, 1])

# Calcula la curva ROC y el área bajo la curva (AUC)
fpr, tpr, _ = roc_curve(y_verdad, y_predic)
roc_auc = auc(fpr, tpr)

# Plot

```

```

plt.figure(figsize=(8, 8))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area
= {:.2f})'.format(roc_auc))
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
roc_curve(y_verdad, y_predic)
plt.show()

```

## Código de generación de la AppWeb app.py y del tuner ngrok

```

%%writefile app.py
import streamlit as st
from PIL import Image
import numpy as np
from tensorflow.keras.models import load_model

# Función para realizar la predicción
def predecir_imagen(imagen):
    # Cargar el modelo
    model =
load_model('/content/drive/MyDrive/TFM/modelo_FINAL.h5')

    # Preprocesar la imagen
    img = imagen.resize((216, 216))
    img_array = np.array(img)
    img_array = img_array / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    # Realizar la predicción
    test_predict = model.predict(img_array)
    predicho = np.argmax(test_predict, axis=1)
    probabil=test_predict
    cell_dict_or = {0: "Benigno", 1: "Maligno", 2: "Normal"}
    predicho = cell_dict_or[predicho[0]]

    return predicho, probabil

# Configuración de la página

st.title("Predicción de imagen de CT")

```

```

st.header("Con esta app se pretende predecir una imagen de CT de
pulmón, con el fin de predecir si es un caso maligno o benigno")
st.subheader(" Solo tienes que cargar o arrastrar una imagen y
pinchar en el botón de predecir que aparecerá abajo")


# Subir una imagen
imagen = st.file_uploader("Subir una imagen", type=["jpg", "jpeg",
"png"])

if imagen is not None:
    # Mostrar la imagen
    st.image(imagen, caption="Imagen subida",
use_column_width=True)

    # Realizar la predicción cuando se presiona el botón
    if st.button("Predecir"):
        # Convertir la imagen de BytesIO a PIL Image
        imagen_pil = Image.open(imagen)

        # Realizar la predicción
        resultado,probabil = predecir_imagen(imagen_pil)

        # Mostrar el resultado
        st.header("Resultado de la Predicción:")
        st.write(f"Clasificación: {resultado}")
        st.write(f"Probabilidad estimada:
{round(100*np.max(probabil), 2)}%")
        st.markdown("---")

# Descargas necesarias
!pip install streamlit -q
!pip install streamlit -q
!pip install pyngrok

#from pyngrok import ngrok

# Se ejecuta Streamlit en segundo plano en el puerto local 80
!nohup streamlit run app.py --server.port 80 &

#ngrok.set_auth_token("su token")
# Conecta ngrok al puerto 80 (el mismo que Streamlit)
ngrok_tunnel = ngrok.connect()
url=ngrok_tunnel.public_url
url

```