Universitat Oberta de Catalunya

uoc.edu

Aprendizaje profundo y neumonía: modelo de clasificación de imágenes de rayos-X para una detección más rápida

Raquel Sauras Salas
Máster Universitario Bioinformática y Bioestadística UOC-UB
Área del trabajo final 4

Romina Rebrij
Antoni Perez Navarro

12/2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Aprendizaje profundo y neumonía: modelo de clasificación de imágenes de rayos-X para una detección más rápida.*
Nombre del autor:	*Raquel Sauras Salas*
Nombre del consultor/a:	*Romina Rebrij*
Nombre del PRA:	*Antoni Pérez Navarro*
Fecha de entrega (mm/aaaa):	12/2021
Titulación:	*Máster Universitario Bioinformática y Bioestadística UOC.UB*
Área del Trabajo Final:	*TFM- Bioinformática y Bioestadística Área 4*
Idioma del trabajo:	Castellano
Número de créditos:	15
Palabras clave	*Neumonía, radiografías torácicas, aprendizaje profundo.*

Resumen del Trabajo

Actualmente, la neumonía es una enfermedad pulmonar prevalente y una de las principales causas mortales en el mundo. Sin ir más lejos, la actual pandemia provocada por COVID-19 y sus variantes nos ha hecho enfrentarnos a la realidad de la importancia de una detección rápida para poder realizar una tratamiento rápido y eficaz.

La herramienta de diagnóstico para la neumonía suelen ser las radiografías torácicas, debido a su rapidez y a su coste. Ofrecen menos exposición ionizante pero también menos claridad en la resolución de la imagen ya que aparecen opacidades o grises difusos.

En este Trabajo de final de Máster se ha propuesto diseñar un modelo de aprendizaje profundo, mediante el lenguaje Python y usando librerías de TensorFlow y Keras a través de Google Colab. El modelo se ha generado usando una red neuronal convolucional (CNN) y la propuesta de mejora de precisión se ha propuesto mediante el aumento de datos.

La base de datos utilizada para este proyecto ha sido extraída del repositorio de Kaggle y consta de 5856 radiografías torácicas clasificadas según si no presentan neumonía (normal) o presentan neumonía (pneumonia virus, pneumonia bacteria).

Finalmente, se ha creado una aplicación web desarrollada en NextJS y publicada en AWS para facilitar la clasificación de imágenes al usuario.

Abstract

Currently, pneumonia is a prevalent lung disease and one of the leading causes of death in the world. Without going any further, the current pandemic caused by COVID-19 and its variants has made us face the reality of the importance of a early detection to carry out early and effective treatment.

The diagnostic tool for pneumonia is usually chest X-rays, due to its easy and quick usage and cost. They offer less ionizing exposure but also less clarity in image resolution as diffuse grays or opacities appear.

The aim of this master's Thesis has been to design a deep learning model, using the Python language and using TensorFlow and Keras libraries through Google Colab. The model has been generated using a convolutional neural network (CNN) and the precision improvement proposal has been suggested by augmentation data.

The database used for this project has been extracted from the Kaggle repository and consists of 5856 chest X-rays classified according to whether they do not present pneumonia (normal) or present pneumonia (pneumonia virus, pneumonia bacteria).

Finally, a web application has been developed in NextJS and published on AWS to bring to the users an easy image classification.

Índice

Lista de figuras

Lista de tablas

1. Resumen

Antecedentes (Background).

La neumonía es una enfermedad pulmonar prevalente, principales causas mortales en todo el mundo, que afecta sobretodo a niños siendo causante del 18% de las muertes en niños menores de 5 años, gente mayor y pacientes con otras patologías. [6, 7]

Actualmente existen diferentes técnicas para el diagnóstico para la imagen, y aunque muchas de ellas son cada vez más precisas, por un motivo de coste y rapidez se suelen usar las radiografías torácicas como herramienta principal para la detección de infecciones pulmonares.

Método.

Se ha diseñado un modelo de red neuronal convolucional (CNN) capaz de clasificar radiografías torácicas en formato JPEG en 3 clases: normal, neumonía bacteriana y neumonía vírica. El conjunto de datos usados para entrenar el modelo pertenece a una base de datos abierta de Kaggle. Para la optimización del modelo se ha usado la técnica de aumento de datos. Además, se ha diseñado una aplicación web creada con NextJS y publicada en AWS.

Resultados.

El modelo optimizado con la técnica de aumento de datos ha ofrecido una precisión ligeramente superior al modelo inicial. Este modelo ha sido utilizado para generar una aplicación web capaz de procesar una imagen y ofrecer al usuario una predicción.

Conclusiones.

Se ha conseguido un modelo de clasificación que genera una predicción con una precisión del 78%. Esta precisión podría mejorarse con el aumento de los epoch, entre otras.

Aportación.

Este proyecto pretende mostrar al público en general que se pueden crear modelos de Deep-learning con el objetivo de ayudar a los profesionales de la salud a detectar precozmente neumonías, gracias a un soporte de inteligencia artificial.

2. Introducción

2.1. Contexto y justificación del Trabajo.

La neumonía es una infección respiratoria aguda que se caracteriza por tratarse de una infección grave que afecta a nivel pulmonar. Ésta suele deberse a una inflamación en los sacos pulmonares causada principalmente por dos agentes patógenos; bacterias o virus. [1]

Se trata de una enfermedad pulmonar prevalente, principales causas mortales en todo el mundo, que afecta sobretodo a niños - causante del 18% de las muertes en niños menores de 5 años-, gente mayor y pacientes con otras patologías [2,3].

Aunque la sintomatología inicial de la neumonía bacteriana o de la vírica no difieran demasiado entre si, en cambio, su tratamiento es muy distinto [4].

Actualmente existen diferentes técnicas para el diagnóstico para la imagen, y aunque muchas de ellas son cada vez más precisas, también tienen un coste muy elevado y no muy accesible para la mayoría de las poblaciones o regiones [2]

Además, Elshennawy, N. M. y Ibrahim, D. M. (2020) ponen de manifiesto la "escasez de expertos en radiología en países con bajos recursos o regiones rurales" o las listas de espera interminables para los diagnósticos en ciertas zonas y, eso, conlleva al aumento de la gravedad de la enfermedad y, también, a su tasa de mortalidad. [2]

Y, aunque el diagnóstico por radiografía sea la técnica de diagnóstico más económica, menos ionizante y de aplicación más rápida que otras técnicas, puede llevar a confusiones a la hora de realizar un diagnóstico, ya que las opacidades que se pueden visualizar pueden llevar al analista a malinterpretar los resultados [5, 2].

Muchos de los estudios consultados en la sección de referencias bibliográficas desarrollan técnicas de aprendizaje automático (machine learning) basados en modelos de aprendizaje profundo (de ahora en adelante, Deep-learning) tales como redes neuronales convolucionales (CNN) pero en técnicas de diagnóstico para la imagen de resolución más precisa como tomografía computarizada (CT).

Teniendo en cuenta que no todos los hospitales o centros de atención de salud tienen acceso a técnicas de diagnóstico más preciso, ya sea por presupuesto

o por arquitectura, y que existen diferencias entre distintas regiones de un mismo país, y con el objetivo de llegar con más rapidez y a más población, la intención es desarrollar un modelo de aprendizaje automático que ayude a minimizar el tiempo de diagnóstico y a mejorar la precisión del diagnóstico con el uso de radiografías.

2.2. Objetivos del Trabajo.

2.2.1. Objetivo general:

Desarrollar un modelo de aprendizaje automático para clasificar imágenes de rayos-x torácico para determinar si hay presencia o no de neumonía.

2.2.2. Objetivos específicos:

1. Identificar las características principales de la neumonía en el diagnóstico para la imagen mediante un modelo de aprendizaje profundo (Deep-learning).
2. Optimizar el modelo para que logre una precisión mínima del 70%.
3. Desarrollar una aplicación web que permita importar imágenes y mostrar resultados.

2.3. Enfoque y método seguido.

Clasificación de datos (imágenes) de la que va a constar este proyecto corresponden a los datos del realizado por Kermany, D. S. et al. (2018). Ésta consta de 5856 radiografías torácicas -obtenido desde las plataformas de bases de datos de Kaggle y Mendeley- en formato JPEG y realizadas a niños de entre 1 a 5 años de la región de Canton (Guangzhou). Esta base de datos contiene radiografías con presencia de neumonía y sin presencia (normal). [6]

A partir de la información anterior, y teniendo en cuenta el estudio realizado por Kermany, D. S. et al. (2018), este proyecto va a constar de las siguientes fases: [6]

1. Fase de análisis de detalles médicos: se estudiarán las características radiológicas pulmonares más relevantes para la detección de neumonía y, también, se analizarán las diferencias entre la infección vírica y bacteriana.

2. Fase de etiquetaje de imágenes: clasificación de datos con la que se va a trabajar ya cuenta con la validación de 3 técnicos expertos en diagnóstico por la imagen. Así que, en esta fase se validará la información obtenida en la fase 1.

3. Fase de procesamiento: en esta fase se van a estandarizar el tamaño de las imágenes y posible tratamiento de la escala de grises de éstas. Hay distintos estudios que han usado el modelo de aprendizaje CNN. Además, para pre-entrenar el modelo, se pueden usar librerías de inteligencia artificial (IA) de libre acceso desarrolladas con Python, tales como Tensorflow (Kermany, D. S. et al., 2018). Se estudiará la posibilidad de transferencia de aprendizaje y de librerías de IA de acceso libre. [6]

4. Fase de entrenamiento y testeo: en esta fase se va a entrenar el modelo de Deep-learning escogido. Se evaluará su rendimiento y se validará con las comparaciones de las clasificaciones realizadas previamente por los expertos del estudio de Kermany, D. S. et al. (2018). [6]

5. Fase de análisis estadístico: se analizarán los resultados obtenidos para determinar la precisión de la clasificación. Esto me ayudará a decidir si es necesario una mejora del modelo.

6. Fase visualización información: la idea es acercar el modelo de clasificación al usuario, por ese motivo se va a trabajar para generar una aplicación web que acepte imágenes de radiografías torácicas en formato JPEG y dé un resultado de fácil comprensión para el usuario.

El método escogido a seguir viene influenciado por los estudios de Kermany, D. S. et al. (2018), Elshennawy, N. M., y Ibrahim, D. M. (2020) y Hashmi, M. F. et al. (2020). También, han influenciado las estrategias adquiridas con relación al análisis y resolución de problemas personales. Aunque pueden existir otros métodos como los explicados por Wang, S. et al. (2020), entre otros, se ha optado por seguir un modelo combinado relacionado con radiografías que creo que puede cumplir con las expectativas. [2,4,6,7]

2.4. Planificación del Trabajo.

2.4.1. Tareas.

Descripción de las tareas propuestas influyendo sus fechas de inicio, fin y entrega en cada fase.

Tarea	Fecha Inicio	Fecha Fin	Hito
Definición del proyecto	15/09	22/09	PAC 0
Plan del proyecto	23/09	4/10	PAC 1
Fase 1 del desarrollo del proyecto	5/10	8/11	PAC 2
Estudio de las características radiológicas pulmonares más relevantes para la detección de neumonía.	5/10	13/10	
Exploración de paquetes para el análisis de imágenes.	26/10	6/11	
Documentar la memoria.	19/10	8/11	
Fase 2 del desarrollo del proyecto	9/11	9/12	PAC 3
Prueba de distintos modelos de Deep-learning.	8/11	11/11	
Entrenamiento del modelo escogido.	11/11	14/11	
Evaluar el rendimiento del modelo.	15/11	17/11	
Validación del modelo Deep-learning.	17/11	21/11	
Análisis de los resultados de clasificación obtenidos.	22/11	24/11	
Mejora del modelo.	24/11	23/12	
Desarrollo de la aplicación web.	29/11	06/12	
Testeo de la aplicación.	7/12	9/12	
Documentar la memoria.	11/11	23/12	
Cierre de la Memoria	10/12	24/12	PAC 4
Revisar el formato y desarrollo de apéndices que requiera la memoria.	10/12	24/12	
Elaboración de la presentación	27/12	3/01	PAC 5a
Defensa Pública	13/01	21/01	PAC 5b

Tabla 1. Tareas del proyecto

2.4.1. Calendario.

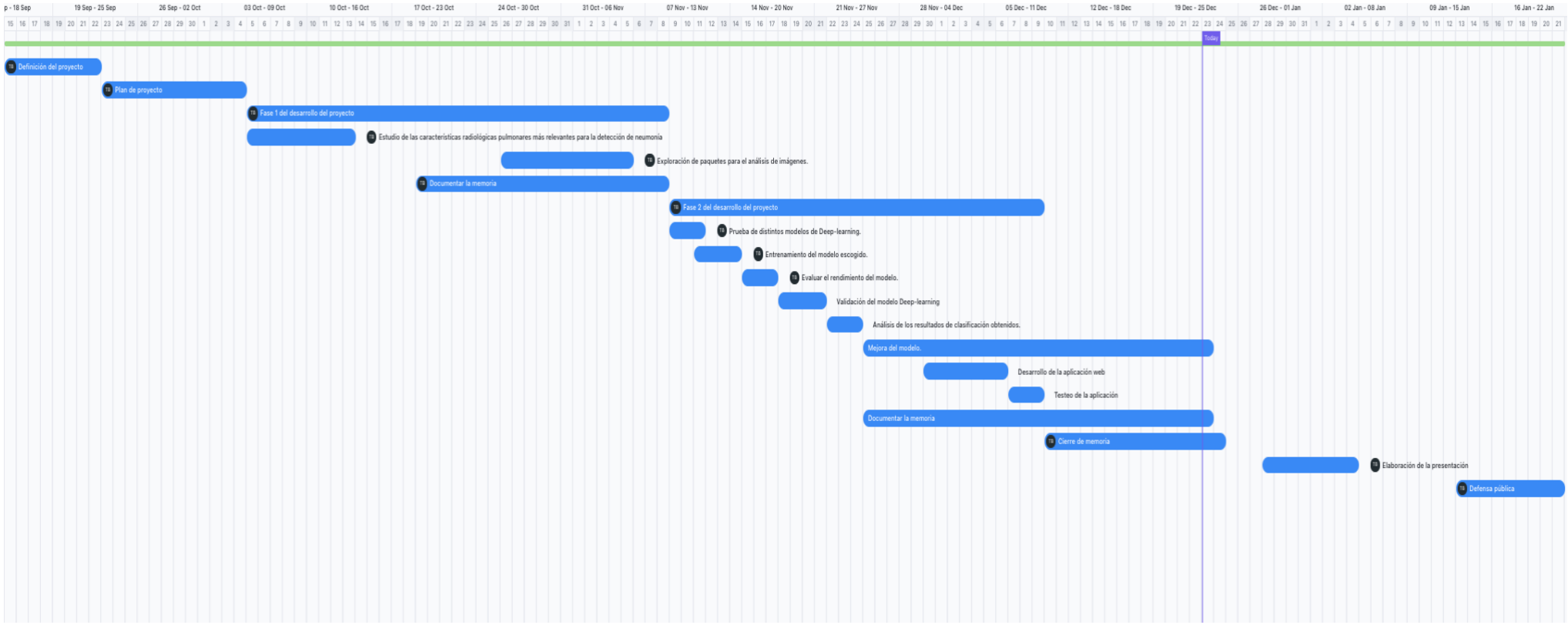

Figura 1. Diagrama de Gantt del proyecto

2.4.2. Hitos.

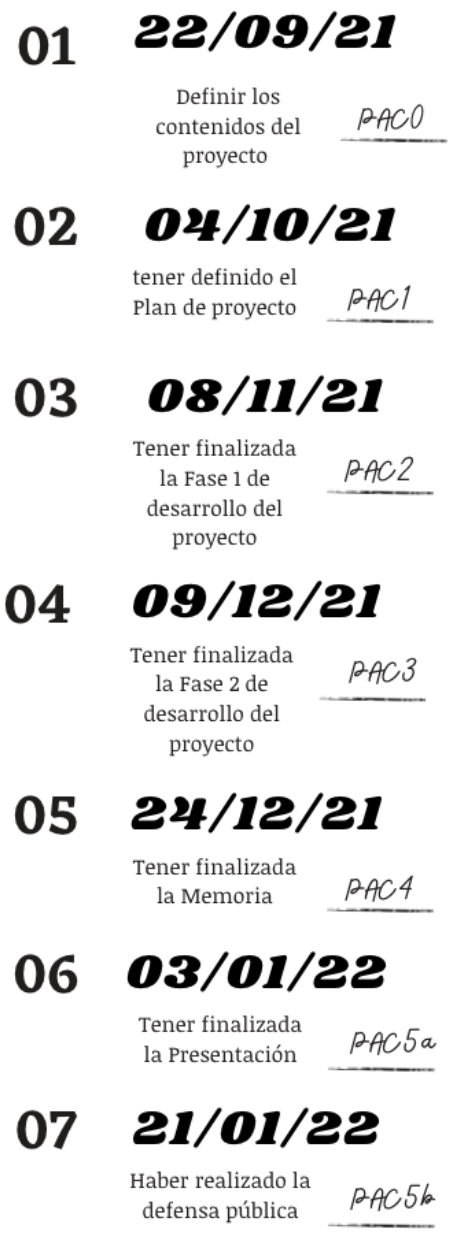

Figura 2. Hitos del proyecto

2.4.3. Análisis de riesgos.

- No poder cumplir con las fechas propuestas para desarrollar las tareas de cada fase:
  Severidad: Moderada
  Probabilidad: Moderada
  Solución: Se pueden acabar de desarrollar en fases posteriores. Se establecerá un periodo entre tareas amplio para mitigar este

posible riesgo.

- No poder encontrar una librería de imágenes que funcione.
  Severidad: Alta
  Probabilidad: Baja
  Solución: Investigar librerías alternativas, pedir ayuda a compañeros o buscar código de funciones en GitHub.

- No conseguir un rendimiento del modelo correspondiente al 70% preestablecido.
  Severidad: Alta
  Probabilidad: Alta
  Solución: Iterar y mejorar el modelo. También se valorará cambiar el modelo.

- No poder elaborar la aplicación web.
  Severidad: Alta
  Probabilidad: Baja
  Solución: Generar documentación donde se explique como se elaboraría.

2.5. Breve sumario de contribuciones y productos obtenidos

2.5.1. Plan de trabajo.

Documento para concretar las tareas y su duración.

2.5.2. Memoria.

Documento desarrollado en formato Word y entregado en PDF donde se detallarán cada fase y con los scripts más relevantes programados en Python.

2.5.3. Código del proyecto.

Documento donde se puede consultar el código (script) del proyecto.

2.5.4. Producto.

Aplicación web donde el usuario podrá importar una radiografía en formato JPEG y podrá saber si el paciente tiene neumonía (vírica o bacteriana) o no.

2.5.5. Presentación virtual.

Presentación realizada con Genially (entrega el 3/01/22).

2.6. Breve descripción de los otros capítulos de la memoria

Capítulo 3. Estado del arte. En este capítulo se realiza una exposición breve sobre los conceptos teóricos y claves para la comprensión de este proyecto.
Capítulo 4. Metodología. Se explica el desarrollo este proyecto de manera detallada.
Capítulo 5. Resultados. Se muestran los resultados obtenidos tanto en el rendimiento del modelo como de la aplicación web.
Capítulo 6. Discusión. Se comentan los resultados obtenidos y se relacionan con el marco teórico.
Capítulo 7. Conclusiones. Se exponen las conclusiones del proyecto y se plantean futuras líneas de trabajo.

3. Estado del arte

3.1. Estructura fisiológica pulmonar.

El pulmón es el órgano terminal del aparto respiratorio. Se encuentra ubicado dentro de la caja torácica formada pos las costillas y la columna vertebral y por encima del diafragma. A partir de la bifurcación de la tráquea, nos encontramos con las estructuras anatómicas de los bronquios (principal, secundario y terciario), bronquiolos (principal y terminal) y alveolos. Como vemos en la figura 3, la red de bronquiolos tiene una semejanza a un árbol donde sus ramas se van bifurcando hasta generar ramas más pequeñas y terminar en “hojas” o usando la metáfora correcta, racimos de uva que simbolizarían a los alveolos.

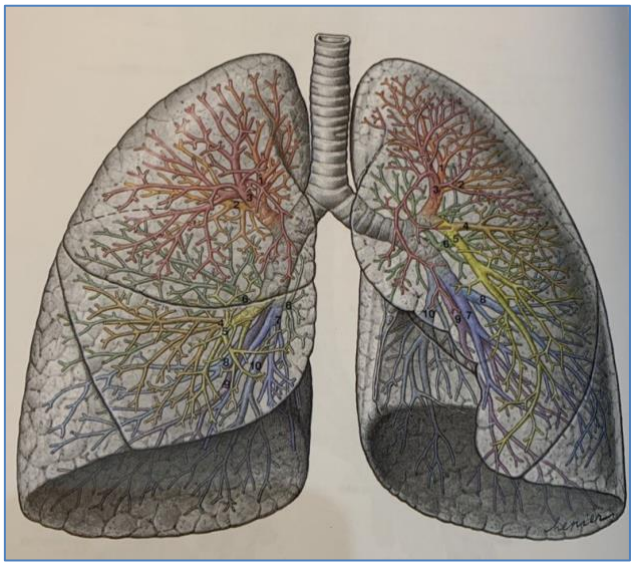

Figura 3. Estructura anatómica de los pulmones. [8]

3.2. Inflamación aguda pulmonar.

Se entiende por inflamación aguda pulmonar aquella que se produce de manera rápida para combatir a un microorganismo patológico, ya sea una bacteria o un virus. La respuesta consiste en la liberación de agentes inmunológicos de defensa tales como leucocitos o neutrófilos encargados de eliminar a los agentes invasores e iniciar el proceso de digestión y deshecho de los tejidos dañados. [9]

En el tejido dañado, existe fuga de proteínas plasmáticas que provocan un edema, tal y como se puede observar en la figura 4 estadio 2.

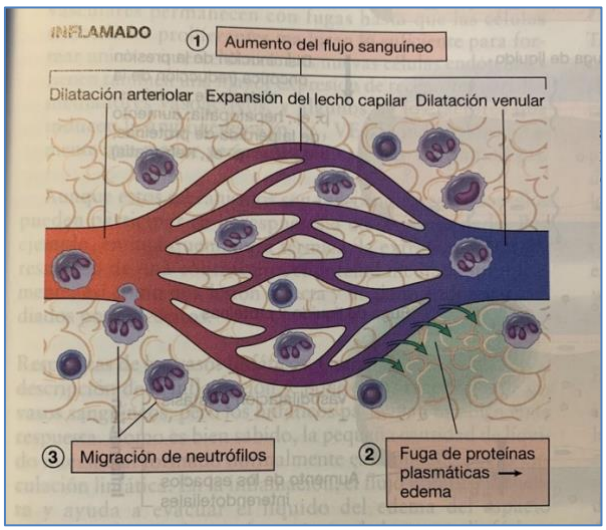

Figura 4. Principales manifestaciones locales de la inflamación aguda. [9]

Las superficies epiteliales -en concreto, el parénquima pulmonar- de los pulmones se encuentran en constante exposición a volúmenes de aire contaminante diariamente, también se desprende y aspira parte de la flora nasofaríngea mientras dormimos; por eso se entiende como neumonía a cualquier infección que se produce en el pulmón. [9, 12]

Existe diversidad de tipologías neumónicas según el agente patógeno que las causa y la altura a la que afecta en el pulmón. Por ejemplo, los causantes más comunes de neumonía aguda suelen ser *S. pneumoniae* (neumococo con inflamación lobular), *H. Influenza* o *M. catarrhalis* (asociados a irritaciones agudas de la enfermedad pulmonar obstructiva crónica o EPOC), *S. aureus* (infección secundaria a infección vírica), *M. pneumoniae*, Virus respiratorios (virus gripales A y B, Adenovirus, Virus Influenza, Virus respiratorio sincitial, rinovirus, etc.) *C. penumoniae* y *C. burnetti* (fiebre Q) que suelen provocar inflamaciones agudas atípicas con inflamación predominante en los tabiques de los alveolos. [9, 10, 11, 12]

3.3. Deep-learning y radiografías torácicas.

La radiografía torácica es una herramienta de diagnóstico que se puede usar para el diagnostico de neumonías pulmonares, ya que se considera una herramienta "rápida" (el paciente tiene que exponerse a la prueba poco tiempo en comparación con otras), menos ionizante que otras y menos cara [5, 2]. Aunque la imagen final que puede ofrecer una radiografía, en este caso torácica, esté llena de opacidades, los grises suelen corresponder a estructuras pulmonar conocidas. Gracias a los estudios de las opacidades, lo radiólogos son capaces de determinar si hay existencia o no de edemas o infecciones. Véase la figura 5 a modo de ejemplo de la identificación de estructuras pulmonares.

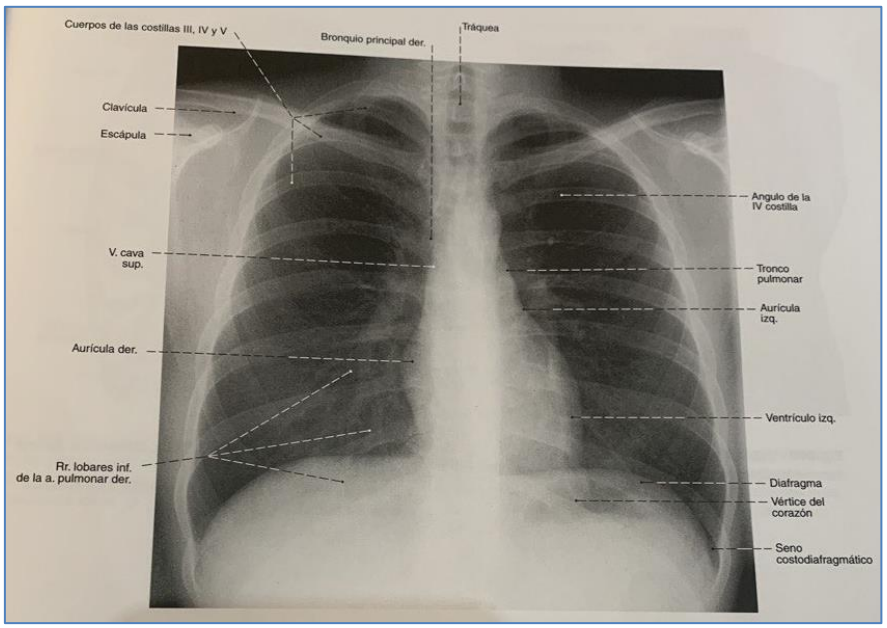

Figura 5. Radiografía torácica con sus partes identificadas [9]

Con la evolución de la medicina y su convergencia con la inteligencia artificial, llega a nuestro conocimiento el Deep-learning, o aprendizaje profundo, que unido al estudio de radiografías torácicas se plantea una herramienta muy potente de soporte diagnostico. [13]

Como se observa en la figura 5, a ojo "desnudo" parece casi imposible ser capaz de diferenciar las diferentes estructuras pulmonares, y se complica más el juicio cuando se exponen las imágenes de la figura 6. Se pueden intuir tonalidades grisáceas entre las imágenes de neumonía vírica y bacteriana en relación con la radiografía normal. Pero se aprecia claramente la dificultad de este diagnóstico sólo a primera vista.

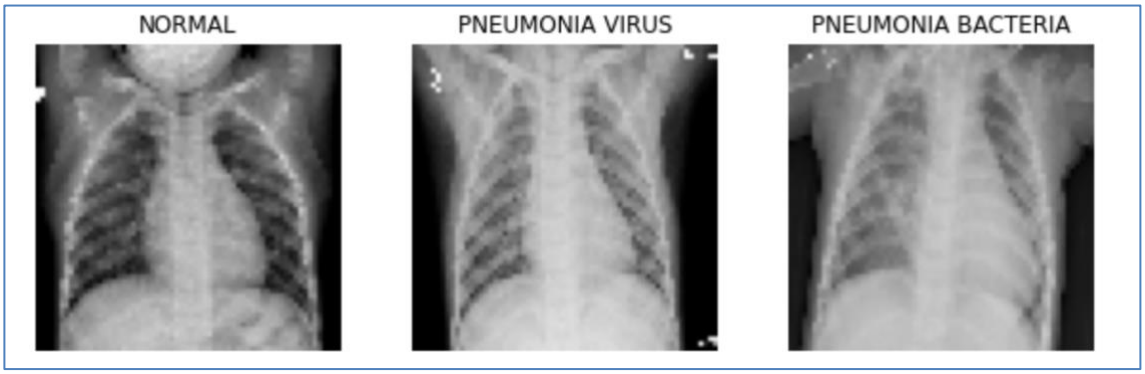

Figura 6. Radiografías torácicas: normal, neumonía vírica y neumonía bacteriana.

Gracias a herramientas de Deep-learning, se pueden plantear modelos de clasificación de imágenes que ayuden a distinguir entre las diferentes opacidades y grises que suelen aparecer en las radiografías torácicas y puedan analizar en un tiempo mínimo todas las peculiaridades de la imagen a partir de un entrenamiento previo con una base de datos que ayude al modelo a poder crear una relación de clasificación o predicción. Algo que se ha estado probando, con resultados prometedores en la bibliografía consultada. [2, 3, 4, 5, 6, 7]

4. Metodología

4.1. Conjunto de datos.

Los datos usados en este proyecto se obtuvieron a través de una plataforma con una gran cantidad de base de datos e imágenes de libre acceso, llamada Kaggle. Se usó Mendeley, también, para acceder a los datos publicados en Kaggle.

Para este proyecto se han usado los datos publicados por Kermany, D. S. et al. (2018) que consta de radiografías torácicas realizadas a niños de entre 1 a 5 años de la región de Canton (Guangzhou). [6]

Los datos han sido publicados en Kaggle por Paul Mooney, su creación esta fechada el 22 de marzo del 2018 y su última actualización (versión 2) fue el 24 de marzo del 2018.

El archivo, de 1Gb, contiene 5856 imágenes en formato JPEG clasificadas en diferentes grupos:

- Imágenes de rayos X torácicas normales.
- Imágenes de rayos X torácicas neumonía bacteriana.
- Imágenes de rayos X torácicas neumonía vírica.

4.2. Entorno.

4.2.1. Entorno de trabajo.

Para realizar este proyecto se ha usado Google Colaboratory (“Colab”), el cual ha permitido ejecutar el código a través del navegador y se han podido evitar los problemas de falta de memoria que ocurrían en el sistema operativo MacOS (M1) usado inicialmente.

4.2.2. Lenguaje de programación.

Se eligió Python como lenguaje de programación para el desarrollo de este proyecto. El motivo por el cual se escogió fue por ser un lenguaje sencillo a la hora de programar y porque tiene la herramienta de TensorFlow.

4.2.3. Librerías usadas

Las librerías usadas en este proyecto han sido:

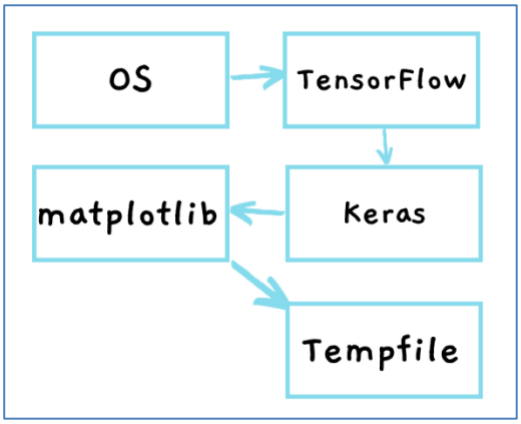

Figura 7. Librerías de programación

4.4. Entorno de los datos.

Los datos se almacenaron en el Drive[1] de la cuenta Google institucional de la UOC. En la carpeta datasets, se encuentran las carpetas Train (entrenamiento), Test (Test) y Val (validación) que contienen las imágenes subagrupadas según el diagnóstico: Normal, Pneumonia Bacteria (neumonía bacteriana) y Pneumonia Virus (neumonía vírica). La ruta general de trabajo fue drive/MyDrive/TFM_Raquel_Sauras/datasets/chest_xray.

[1] Encontraran la información en el siguiente enlace.

4.4.1. Reconfiguración de los datos.

Como se muestra en la figura 8, las imágenes fueron reconfiguradas a un mismo tamaño más pequeño, 64 x 64, y transformado a “rgb” para que cada una de las imágenes tenga el mismo número de canales (3).

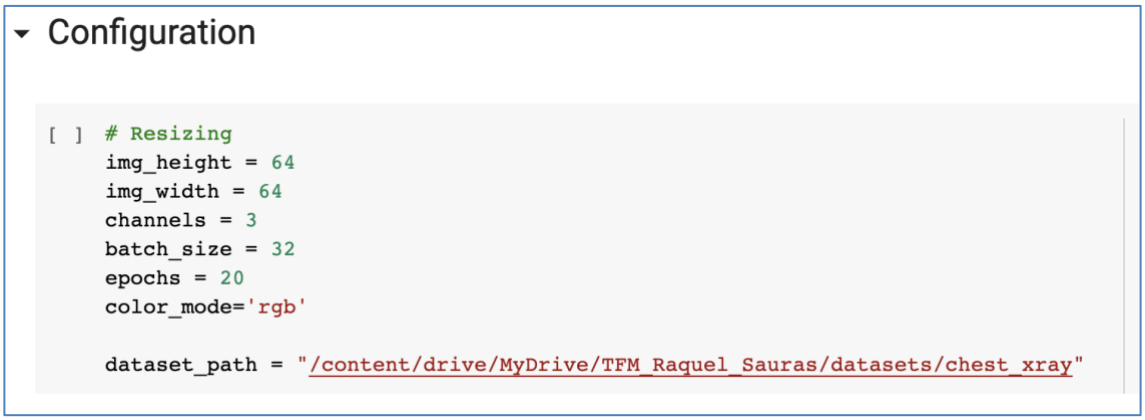

Figura 8. Script sobre la configuración del tamaño de la imagen.

4.4.2. Clasificación de los datos

Los datos importados ya habían sido agrupados inicialmente de la siguiente manera:

- **Train (entrenamiento)**: imágenes clasificadas como entrenamiento. Son las imágenes que inicialmente se usaron para entrenar el modelo de Kermany, D. S. et al. (2018). [6]
- **Test (prueba):** corresponde a las imágenes que se usan para probar el modelo en cuestión.
- **Val (validación)** en esta carpeta se hayan las imágenes para evaluar el modelo.

Aunque inicialmente los datos ya estaban reagrupados; se decidió no contar con la carpeta inicial de validación ya que sólo contenía imágenes de neumonía bacteriana. Para generar los subgrupos de entrenamiento (train) y validación (val) se generó una muestra aleatoria tal y como se puede observar en la figura 9. Con esta decisión, la muestra de imágenes que inicialmente era de 5856 se quedó en 5840, perdiéndose 16 imágenes de la carpeta inicial validación.

Los datos quedaron clasificados de la siguiente manera:

	Normal	Bacteriana	Vírica	Total
Entrenamiento	1073	2024	1076	4173
Prueba	234	242	148	627
Validación	268	506	269	1043
Total	1577	2774	1495	5840

Tabla 2. Clasificación de las imágenes según tipo y grupo.

También se mostró, para comprobar que se habían subclasificado las imágenes según lo parámetros, el número de clases encontradas, figura 10.

Classification datasets

```
# Classifying train dataset in 3 labels

train_ds = tf.keras.utils.image_dataset_from_directory(
  join(dataset_path, "train"),
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)

validation_ds = tf.keras.utils.image_dataset_from_directory(
  join(dataset_path, "train"),
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)


# Classifying test dataset in 3 labels
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    join(dataset_path, "test"),
    color_mode=color_mode,
    image_size=(img_height, img_width),
    batch_size=batch_size)
```

Figura 9. Script sobre la clasificación de las imágenes según grupos.

```
# Showing found labels
class_names = train_ds.class_names
nClasses = len(class_names)
print("Train classes", class_names)
print('Total number of outputs : ', nClasses)

print("Validation classes", validation_ds.class_names)
print('Total number of validation outputs : ', len(validation_ds.class_names))

print("Test classes", test_ds.class_names)
print('Total number of outputs : ', len(test_ds.class_names))

# Showing 10 images from de training set already resized
show_images(class_names, train_ds)
```

Figura10. Script donde se muestra los datos según la subclasificación aleatoria.

4.5. Creación del modelo.

Los diferentes artículos consultados usaban el modelo CNN como modelo de referencia en Deep-learning para clasificar imágenes. Así que usó este modelo para la clasificación de las imágenes de rayos-x torácicas.

Para conseguir entrenar al modelo, se usó la función Sequential () a la que se le añadieron 3 capas convolucionales mediante la clase Conv2D (16, 32 y 64) y al tener los datos normalizados, se usó la función “ReLu” como activadora.

La función MaxPooling2D se encarga de reducir la muestra en ancho y alto (2D) al finalizar cada capa añadida. En última instancia, se añade la capa de compactación de capas o Flatten().

Por último, la función Dense de unión de las capas, con 128 unidades, como capa de salida. [14]

Training the model

```
from tensorflow.keras import datasets, layers, models

num_classes = len(class_names)

model = models.Sequential([
  layers.Rescaling(1./255, input_shape=(img_height, img_width, channels)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

Figura 11. Script modelo CNN usando la función Sequential ().

4.6. Entrenamiento y validación del modelo.

Se generó el modelo CNN, tal y como muestra la figura 11, y se compiló y entrenó usando las funciones model.compile y model.fit, respectivamente.

```
# Compilation of the model
model.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])

# Training the model
training_model = model.fit(train_ds, validation_data=validation_ds, epochs=epochs)
```

Figura 12. Fase de compilación y entrenamiento.

Se estudiaron los parámetros de precisión del modelo y se mostraron para su posterior evaluación. Se usaron las funciones acc y loss para el estudio de la precisión, tal y como se muestra en la figura 13.

```
# Accuracy

acc = training_model.history['accuracy']
val_acc = training_model.history['val_accuracy']

loss = training_model.history['loss']
val_loss = training_model.history['val_loss']
```

Figura 13. Estudio de la precisión del modelo CNN.

4.7. Mejora del modelo.

Para la mejora del modelo, se planteó una función de aumento de información que consiste en ir rotando las imágenes en el eje horizontal para poder sacar la mayor información de cada capa. Con la función “prepare” se aumentarán los datos de las imágenes destinadas al entrenamiento, ya que lo que nos interesa es mejorar el modelo.

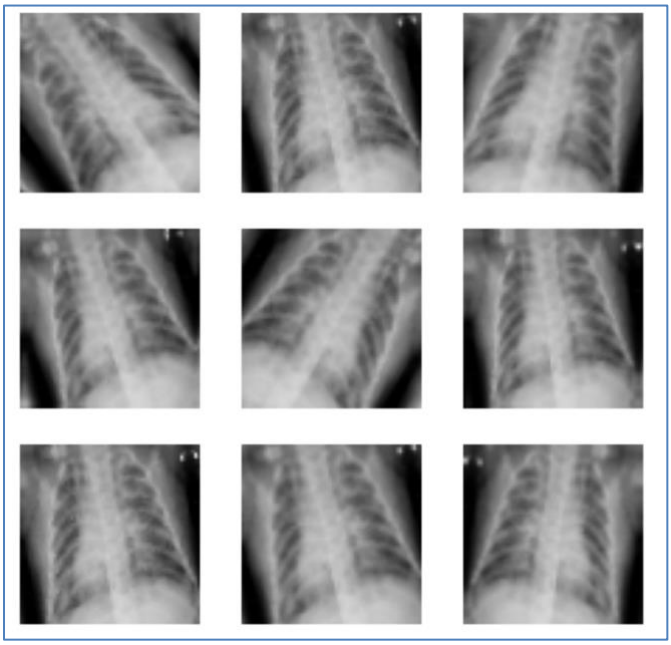

Figura 14. Rotación de imágenes para aumentar la información.

```
# Data augmentation

AUTOTUNE = tf.data.AUTOTUNE

data_augmentation = tf.keras.Sequential(
  [
    layers.RandomFlip("horizontal", input_shape=(img_height, img_width, channels)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
  ]
)

# Visualizing augmented images

plt.figure(figsize=(10, 10))
for images, _ in train_ds.take(1):
  for i in range(9):
    augmented_images = data_augmentation(images)
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(augmented_images[0].numpy().astype("uint8"))
    plt.axis("off")

def prepare(ds, shuffle=False, augment=False):
  # Resize and rescale all datasets.

  if shuffle:
    ds = ds.shuffle(1000)

  # Use data augmentation only on the training set.
  if augment:
    ds = ds.map(lambda x, y: (data_augmentation(x, training=True), y), num_parallel_calls=AUTOTUNE)

  return ds
```

Figura 15. Aumento de la información para mejorar el modelo.

Además, se aumento el número de “epoch” -parámetro que determina el número de veces que el algoritmo trabajará con todos los datos de entrenamiento (véase figura 16)- en relación con el modelo inicial, de 20 se pasó a 30. [15]

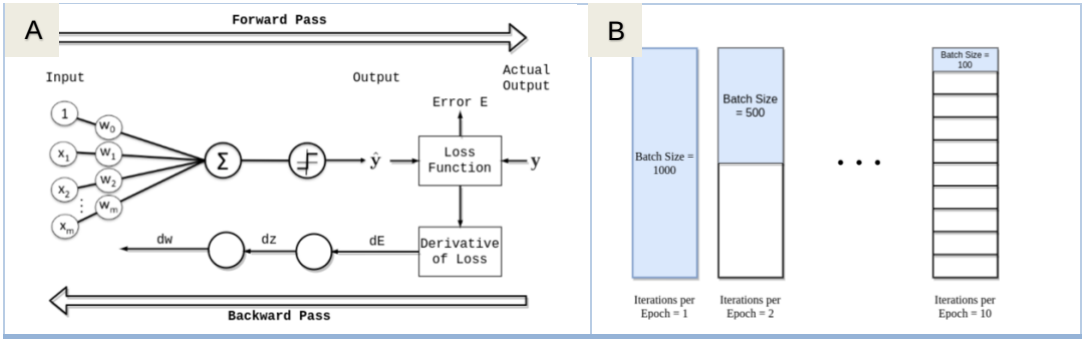

Figura 16. A) Representación gráfica del procese en una red neural. B) Relación entre Bach e Iteraciones, para entender visualmente el concepto de epoch. [15]

4.8. Desarrollo de la aplicación web.

En primer lugar, se guardó el modelo optimizado, tal y como se puede ver en la figura 17. Para el desarrollo de la aplicación se usó NextJS, un framework en JavaScript para generar aplicaciones web fácilmente.

Saving model

```
import tempfile
import os

MODEL_DIR = os.path.join(dataset_path, "model")
version = 2
export_path = os.path.join(MODEL_DIR, str(version))
print(f'export_path = {export_path}')

tf.keras.models.save_model(
    model,
    export_path,
    overwrite=True,
    include_optimizer=True,
    save_format=None,
    signatures=None,
    options=None
)

print('Saved model:')
!ls -l {export_path}
```

Figura 17. Script para guardar el modelo optimizado.

Para usar el modelo, se guardó y se usó TensorFlow serving que genera una API REST para poder interactuar con el modelo. Existe un segundo componente que usa Flask que se encarga de convertir la imagen que el usuario sube al formato que entiende el componente de TensorFlow serving (matriz de píxeles, redimensiona la imagen, etc.).

El proceso de comunicación entre los diferentes componentes se puede visualizar en la figura 18.

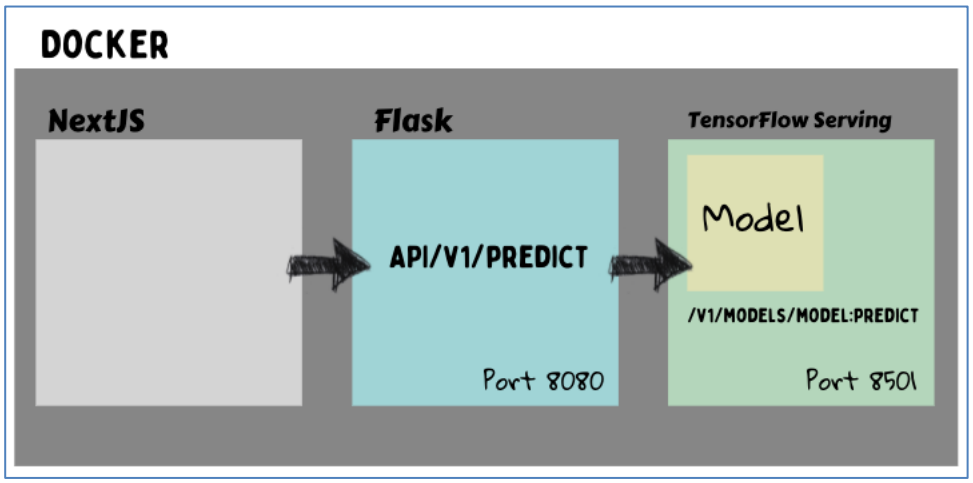

Figura 18. Esquema de comunicación entre los diferentes componentes del desarrollo de la aplicación web.

4.9. Publicación de la aplicación web.

La página web consiste en un formulario que permite al usuario subir una imagen en JPEG y obtener una predicción.

Para la publicación de la aplicación web se usó un dominio propio y se usó un servidor en AWS. Además, para su publicación se usó Docker, se containearizó tanto la web como las dos APIs y se desplegaron los contenedores de Docker en la instancia en AWS usando Docker compose.

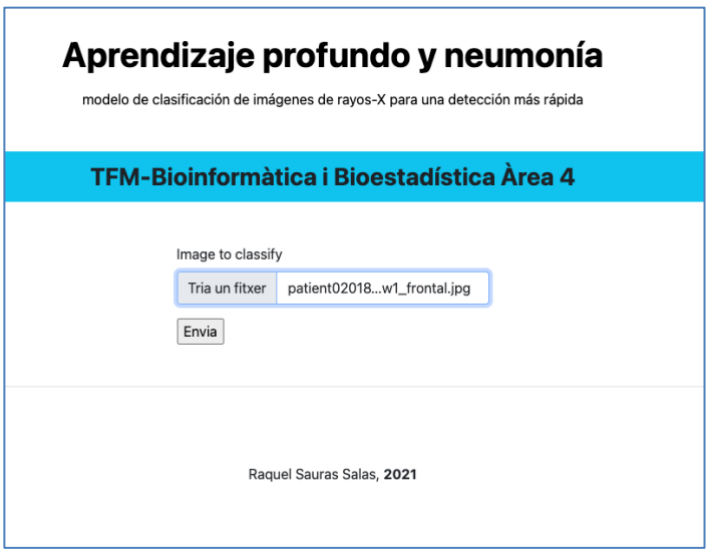

Figura 19. Interfaz de la aplicación web.

5. Resultados

A continuación, se exponen los resultados obtenidos:

5.1. Modelo inicial.

El modelo inicial, modelo 1, contaba con 4173 imágenes (1073 normales, 2024 bacterianas y 1076 víricas, como se puede apreciar en la tabla 2) y generó una precisión del 70% y un porcentaje de pérdida que supera el 80%.

En la figura 20, en muestra la relación entre la precisión y la pérdida según grupos (entrenamiento y validación) según la evolución de los epoch. Si miramos la relación entre la precisión de entrenamiento y la de la validación, éstas muestran la misma tendencia hasta que se llega al 4 epoch, en ese momento las curvas se distancian; para la precisión del entrenamiento tiende a aumentar, pero no sucede lo mismo para la precisión de la validación, que disminuye.
En esta misma figura, la relación de la pérdida entre el entrenamiento y la validación empiezan con una tendencia a la baja similar hasta el epoch 5, cuando la pérdida de validación vuelve a aumentar hasta superar los niveles iniciales y, en cambio, la de entrenamiento sigue con la tendencia a la baja.

Figura 20. Evolución de la precisión y la pérdida frente a los “epoch”.

El porcentaje de precisión del modelo llegó al 70,19% y la pérdida fue del 2,6756, como se puede ver en la figura 21.

```
[51] test_loss, test_acc = model.evaluate(test_ds, verbose=2)

     print(test_acc)

     20/20 - 4s - loss: 2.6756 - accuracy: 0.7019 - 4s/epoch - 201ms/step
     0.7019230723381042
```

Figura 21. Valor de la precisión generada en el modelo inicial.

5.2. Modelo optimizado.

Para el modelo optimizado, observamos que -figura 22- la relación entre precisión de entrenamiento y la de la validación tienden a aumentar a medida que aumenta el número de epochs. Si bien es cierto que no siguen un aumento logarítmico idéntico, ya que la precisión de la validación presenta una tendencia más "irregular" con más aumentos y bajadas dependiendo del epoch.
En esta misma figura, la relación de la pérdida entre el entrenamiento y la validación empiezan con una tendencia a la baja similar que se mantiene hasta el último epoch. Se observa una ligera subida a partir del epoch 20 de la gráfica de la pérdida de la validación.

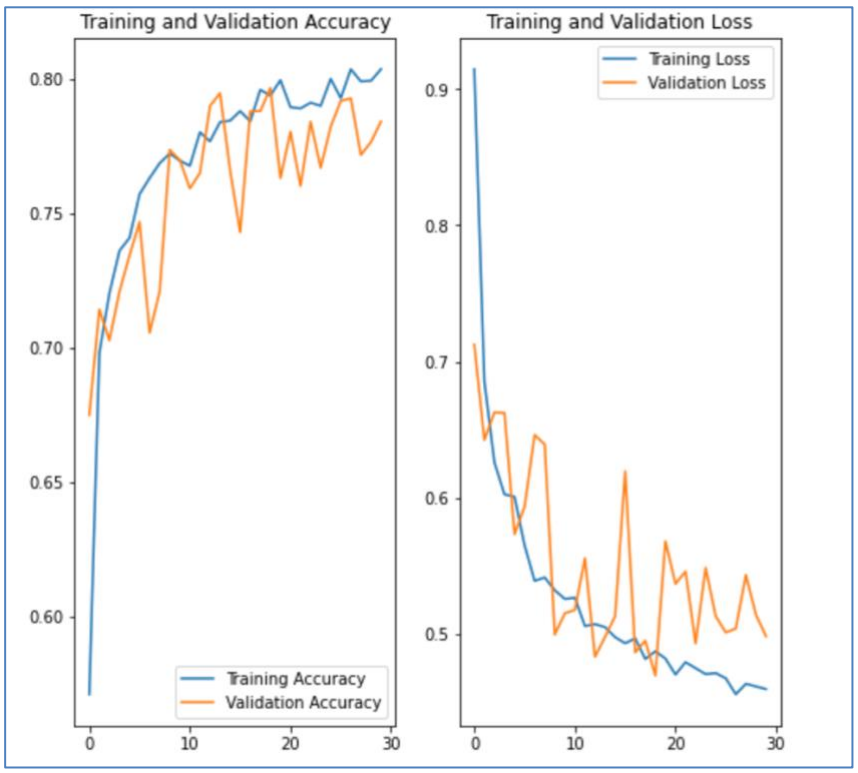

Figura 22. Evolución de la precisión y la pérdida frente a los "epoch".

El porcentaje de precisión del modelo optimizado llegó al 78,37% y la pérdida fue del 0,698, como se puede ver en la figura 23.

```
test_loss, test_acc = model.evaluate(test_ds, verbose=2)

print(test_acc)

20/20 - 5s - loss: 0.6988 - accuracy: 0.7837 - 5s/epoch - 232ms/step
0.7836538553237915
```

Figura 23. Valor de la precisión generada en el modelo optimizado.

5.3. Aplicación web.

La interfaz web que el usuario ve es la que se representa en la figura 15. El usuario, ahora, podrá subir la imagen en formato JPEG si se dirige a “Tria un fitxer” -figura 24- que desee analizar y cuando la tenga seleccionada, podrá apretar a “Envia”.

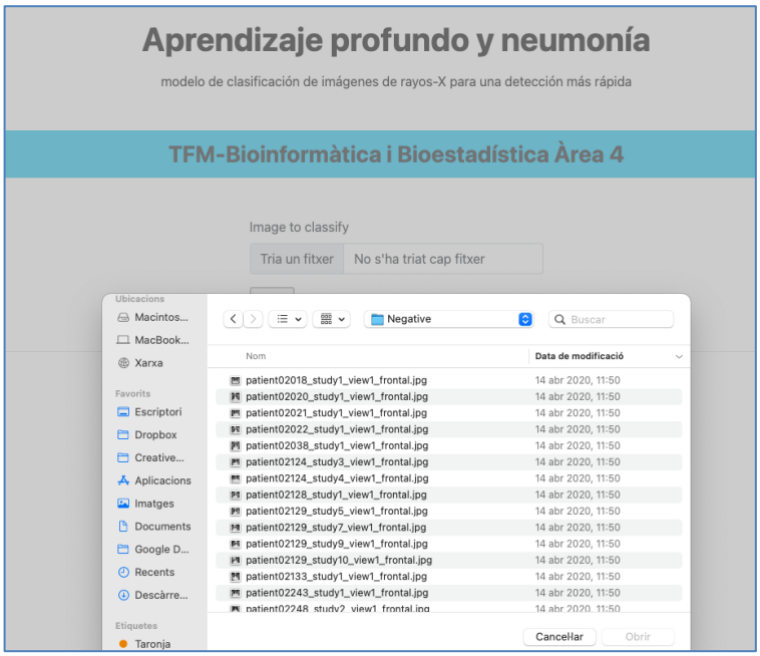

Figura 24. Interfaz web cuando el usuario quiere escoger una imagen a analizar.

Según la imagen escogida por el usuario, la página web devuelve la predicción. A modo de ejemplo, se pude ver en la figura 25 las distintas posibilidades de predicción.

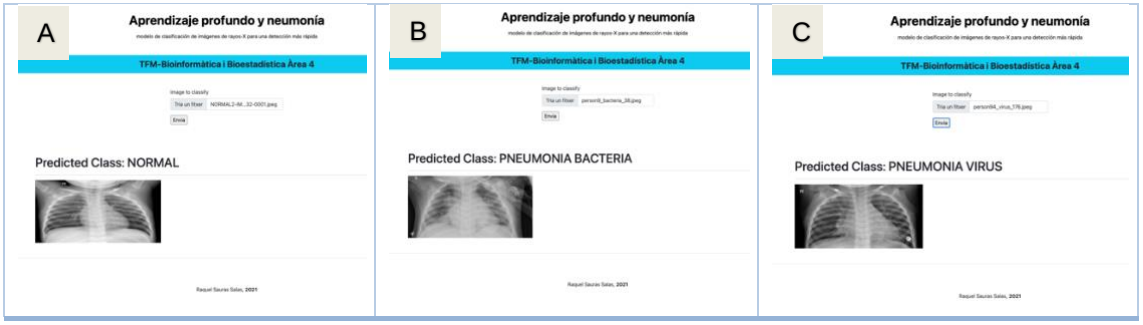

Figura 25. Retorno de las diferentes posibilidades de predicción de la aplicación web. A) Normal B) Neumonía Bacteriana C) Neumonía Vírica.

En el siguiente enlace, se puede visualizar el funcionamiento de la aplicación web y sus predicciones.

6. Discusión

Actualmente, y debido a diferentes factores socioeconómicos, existe una escasez de profesionales en radiología en relación con el volumen de pacientes que llegan a consulta con sintomatología pulmonar. [2]

La inteligencia artificial pretende ayudar a los profesionales a establecer un diagnostico más rápido y fiable y que se relacione o no con la sintomatología presentada.

Estudios como los revisado para realizar este trabajo, han sido capaces de diseñar aplicaciones con precisiones muy elevadas, aunque siempre teniendo en cuenta que un profesional de la radiología tendría que validar la predicción. [6, 4, 2 y 7].

En este proyecto, se han usado métodos de aumento de datos para obtener un modelo de clasificación que ha logrado un porcentaje de precisión del 78,37%.

El modelo de clasificación generado ofrece al usuario una buena clasificación de las imágenes siendo capaz de diferenciar entre radiografía torácica normal y neumonía, pero las opacidades existentes pueden contribuir a generar algunos errores entre la diferenciación de neumonía bacteriana y vírica. Por ese motivo, algunos estudios proponen tomografías computarizadas como método más eficaz para la detección y diferenciación de la neumonía vírica y bacteriana.

Una de las limitaciones que han surgido durante la realización del proyecto fue la creación del entorno de trabajo y la falta de memoria del equipo usado e incompatibilidad de librerías, pero gracias a Google Colaboratory se consiguió

solucionar. Además, nos da la oportunidad de poder compartir el código con otros investigadores.

Otra limitación del proyecto realizado es que se han usado pocas epoch en comparación con los estudios consultados. En nuestro caso solo se han realizado 20 epochs, para el modelo inicial y 30 para el modelo optimizado. Esta diferencia de epochs se debe a la falta de equipo adecuado para realizar un modelo que compile de manera rápida y eficaz, ahorrando tiempo al investigador.

También se debe tener en cuenta que sólo se ha trabajo con un tipo de red neural (CNN) y se ha generado un único modelo el cual se ha mejorado. Hubiese sido interesante poder realizar más modelos y compararlos entre ellos, pues el factor tiempo ha sido también una limitación para este trabajo.

7. Conclusiones

7.1. Conclusiones.

La neumonía es una enfermedad pulmonar prevalente y una de las principales causas mortales en todo el mundo, que a diferentes colectivos. Debido a sus características y su índice de mortalidad, y podemos añadir que a la actual pandemia mundial por SARS-COV-2, actualmente existe mucha bibliografía que es accesible al público.

En este trabajo se ha conseguido desarrollar una aplicación web que permite al usuario poder clasificar la imagen de interés según si existe presencia o no de neumonía y de qué tipología.

Por lo general, se han conseguido lograr todos los objetivos específicos planteados y se superado la precisión mínima establecida en el objetivo específico 2, que era del 70%.

Con la realización de proyecto, y con la gran cantidad de bibliografía existente actualmente en relación con tema escogido, cobra más importancia la implementación de modelos de aprendizaje automático para ayudar a nuestros profesionales a diagnosticar con más rapidez y pocos recursos enfermedades que con un tratamiento adecuada a tiempo, pueda salvar vidas.

7.2. Líneas de futuro.

Las líneas futuras de trabajo que se pueden plantear a raíz de este proyecto podrían ser las siguientes:

- Mejorar el modelo hasta llegar a un porcentaje de precisión del 95%. Es muy importante poder establecer una buena diferenciación entre neumonía vírica y bacteriana, porque como se ha comentado anteriormente, el tratamiento difiere.

- Generar una red de profesionales donde se compartan los modelos desarrollados y se compartan, además, bases de datos para mejorar los modelos existentes. Esto contribuiría al poder generar el mejor modelo posibles y poderlo compartir e implementar en todos los centros de salud a nivel mundial.

- Mejorar la aplicación web. Seria interesante que tanto la apariencia como la funcionalidad de ésta. Ofreciendo al usuario más información de la imagen de interés y dejando un espacio para que el usuario pueda compartir comentarios tales como la sintomatología o la validación (en caso de ser un radiólogo) de la predicción generada. Eso alimentaria y mejoraría el modelo y podría abrir nuevas líneas de investigación.

- Implementar en la mayoría de los centros de salud la inteligencia artificial como apoyo al diagnostico, ofreciendo formación y colaboración con los profesionales de la salud.

7.3. Seguimiento de la planificación

La planificación inicial propuesta para este proyecto ha ido sufriendo modificaciones debido a los imprevistos que han generado las actividades no previstas o los contratiempos personales.

Los problemas que mayor tiempo han consumido y han forzado la modificación del calendario han sido:

- La creación del entorno de trabajo, que como hemos explicado se solucionó gracias a Google Colab.
- La falta de tiempo para seguir mejorando el modelo. Aunque éste ha cumplido con el mínimo deseado, creo que con más tiempo se podría establecer una precisión más buena.
- La realización de tareas que no fueron contempladas inicialmente por la falta de experiencia.

8. Bibliografía

[1] Johns Hopkins Medicine. Pneumonia. Accesible online: https://www.hopkinsmedicine.org/health/ conditions-and-diseases/pneumonia (acceso el 20 de Septiembre 2021).

[2] Elshennawy, N. M., y Ibrahim, D. M. (2020). Deep-Pneumonia Framework Using Deep Learning Models Based on Chest X-Ray Images. *Diagnostics*, *10*(9). https://doi.org/10.3390/diagnostics10090649

[3] Cavallazzi, R., y Ramirez, J. A. (2018). Influenza and Viral Pneumonia. In *Clinics in Chest Medicine* (Vol. 39, Issue 4, pp. 703–721). W.B. Saunders. https://doi.org/10.1016/j.ccm.2018.07.005

[4] Hashmi, M. F., Katiyar, S., Keskar, A. G., Bokde, N. D., y Geem, Z. W. (2020). Efficient pneumonia detection in chest xray images using deep transfer learning. *Diagnostics*, *10*(6). https://doi.org/10.3390/diagnostics10060417

[5] Güneyli, S., Atçeken, Z., Doğan, H., Altınmakas, E., y Atasoy, K. Ç. (2020). Radiological approach to COVID-19 pneumonia with an emphasis on chest CT. In *Diagnostic and Interventional Radiology* (Vol. 26, Issue 4, pp. 323–332). AVES. https://doi.org/10.5152/dir.2020.20260

[6] Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. C. S., Liang, H., Baxter, S. L., McKeown, A., Yang, G., Wu, X., Yan, F., Dong, J., Prasadha, M. K., Pei, J., Ting, M., Zhu, J., Li, C., Hewett, S., Dong, J., Ziyar, I., … Zhang, K. (2018). Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. *Cell*, *172*(5), 1122-1131.e9. https://doi.org/10.1016/j.cell.2018.02.010

[7] Wang, S., Zha, Y., Li, W., Wu, Q., Li, X., Niu, M., Wang, M., Qiu, X., Li, H., Yu, H., Gong, W., Bai, Y., Li, L., Zhu, Y., Wang, L., y Tian, J. (2020). A fully automatic deep learning system for COVID-19 diagnostic and prognostic analysis. *European Respiratory Journal*, *56*(2). https://doi.org/10.1183/13993003.00775-2020

[8] Sobotta, J., Putz, R. y Pabst, R. (2005). Sobotta atlas de anatomía humana. Madrid: editorial medica paramericana.

[9] Kumar, Vinay; Abbas, Abul K.; Aster, Jon C. (2013). Robbins. Patología humana. London: Elsevier.

[10] Prasso, J. E., y Deng, J. C. (2017). Postviral Complications: Bacterial Pneumonia. In *Clinics in Chest Medicine* (Vol. 38, Issue 1, pp. 127–138). W.B. Saunders. https://doi.org/10.1016/j.ccm.2016.11.006

[11] Henig, O., & Kaye, K. S. (2017). Bacterial Pneumonia in Older Adults. In *Infectious Disease Clinics of North America* (Vol. 31, Issue 4, pp. 689–713). W.B. Saunders. https://doi.org/10.1016/j.idc.2017.07.015

[12] Oterino Serrano, C., Alonso, E., Andrés, M., Buitrago, N. M., Pérez Vigara, A., Parrón Pajares, M., Cuesta López, E., Garzón Moll, G., Martin Espin, I., Bueno Barriocanal, M., De Ceano-Vivas la Calle, M., Calvo Rey, C., y Bret-Zurita, M. (2020). Pediatric chest x-ray in covid-19 infection. *European Journal of Radiology*, *131*. https://doi.org/10.1016/j.ejrad.2020.109236

[13] Hurt, B., Kligerman, S., y Hsiao, A. (2020). Deep Learning Localization of Pneumonia: 2019 Coronavirus (COVID-19) Outbreak. In *Journal of Thoracic Imaging* (Vol. 35, Issue 3, pp. W87–W89). Lippincott Williams and Wilkins. https://doi.org/10.1097/RTI.0000000000000512

[14] TensorFlow. (2021). Image classification. Consultada Noviembre 2021, en https://www.tensorflow.org/tutorials/images/classification

[15] Baeldung. (2021). Epoch in Neural Networks. Consultada December 2021, from https://www.baeldung.com/cs/epoch-neural-networks