

Detección de neumonía a partir de radiografías de tórax (CXR) mediante redes neuronales convolucionales

Mario Ubierna San Mamés

Máster universitario en Ciencia de Datos - Universitat Oberta de Catalunya
Àrea de la medicina

Jordi de la Torre Gallart

06/2022



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>Detección de neumonía a partir de radiografías de tórax (CXR)</i>
Nombre del autor:	<i>Mario Ubierna San Mamés</i>
Nombre del consultor/a:	<i>Jordi de la Torre Gallart</i>
Nombre del PRA:	<i>Jordi de la Torre Gallart</i>
Fecha de entrega (mm/aaaa):	06/2022
Titulación:	<i>Máster universitario en Ciencia de Datos</i>
Área del Trabajo Final:	<i>Área Medicina (TFM-Med)</i>
Idioma del trabajo:	<i>Castellano</i>
Palabras clave	<i>Redes neuronales convolucionales, radiografía de tórax, neumonía.</i>
<p>Resumen del Trabajo (máximo 250 palabras): <i>Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.</i></p> <p>La finalidad de este trabajo es la construcción de un modelo de <i>deep learning</i>, que nos permita determinar si a partir de radiografías de tórax (CXR) un paciente presenta una neumonía.</p> <p>Aunque las redes neuronales artificiales aparecen por primera vez en 1958, cuando apareció el perceptrón, se han ido mejorando las diferentes propuestas hasta tal punto que desde hace 10 años hasta día de hoy son imprescindibles en nuestra sociedad gracias al elevado rendimiento que proporcionan. Dentro de las redes neuronales, las redes neuronales convolucionales (CNN) han adquirido una gran importancia en los últimos años, debido al alto nivel de precisión que son capaces de alcanzar en tareas de clasificación de imágenes.</p> <p>Para poder lograr el objetivo del proyecto, se ha hecho uso de un elevado número de radiografías de tórax, éstas se han preprocesado y posteriormente se han introducido a la red neuronal convolucional con el objetivo de clasificar las mismas.</p> <p>Gracias a las más de 25000 radiografías y el modelo generado se ha conseguido determinar si un paciente padece una neumonía con una elevada precisión, permitiendo así dar un pequeño paso en el avance biotecnológico.</p>	

Abstract (in English, 250 words or less):

The purpose of this work is the construction of a deep learning model, which allows us to determine if a patient has pneumonia from chest radiographs (CXR).

Although artificial neural networks appear for the first time in 1958, when the perceptron appeared, the different proposals have been improving to such an extent that from 10 years ago to the present day they are essential in our society thanks to the high performance they provide. Within neural networks, convolutional neural networks (CNN) have gained great importance in recent years, due to the high level of precision they are capable of achieving in image classification tasks.

To achieve the objective of the project, a large number of chest radiographs have been used, these have been preprocessed and subsequently entered into the convolutional neural network to be able to classify them.

Thanks to the more than 25,000 chest radiographs and the generated model, it has been possible to determine with great precision if a patient suffers from pneumonia, thus allowing a small step in biotechnological progress to be taken.

Índice

1. Introducción.....	1
1.1 Contexto y justificación del trabajo	1
1.2 Marco teórico sobre la neumonía	2
1.3 Resumen de la problemática.....	6
1.3.1 Definición del problema.....	6
1.3.2 Solución	7
1.4 Motivación personal.....	7
1.5 Objetivos del trabajo	8
1.6 Enfoque y método seguido	8
1.7 Planificación del trabajo.....	10
1.8 Breve resumen de productos obtenidos	12
1.9 Breve descripción de los otros capítulos de la memoria.....	12
2. Estado del arte	14
2.1 Estudios similares	14
2.2 Tecnologías usadas	16
2.2.1 Entornos de trabajo.....	16
2.2.2 Lenguaje de programación	16
2.2.3 Entornos de programación.....	16
2.2.4 Librerías utilizadas	17
3. Marco teórico sobre las técnicas utilizadas	18
3.1 Aprendizaje automático	18
3.2 Aprendizaje profundo	19
3.3 Redes neuronales convolucionales	22
3.4 Redes neuronales convolucionales pre-entrenadas <i>EfficientNet</i>	23
4. Desarrollo	27
4.1 Extracción y comprensión de los datos	27
4.1.1 Dónde y cómo se obtuvieron los datos	27
4.1.2 Estructura del repositorio	29
4.1.3 Estructura de los datos	31
4.2 Preprocesado de los datos.....	33
4.2.1 Lectura y filtrado de los datos	33
4.2.2 Creación de los conjuntos de entrenamiento, validación y test.....	34
4.2.3 Transformación de las imágenes	35
4.2.4 Balanceo de los datos.....	36
4.2.5 Generación de los ficheros de salida	39
4.3 Creación de los modelos	40
4.3.1 Data Generator	40
4.3.2 Carga de las imágenes preprocesadas	40
4.3.3 Transfer learning.....	41
4.3.4 Fine-tuning	43
4.3.5 Detalles de implementación	43
4.4 Evaluación de los modelos.....	45
4.4.1 Matriz de confusión	45
4.4.2 Exactitud	46
4.4.3 Precisión	47
4.4.4 Recall o sensibilidad	47
4.4.5 Especificidad.....	47

4.4.6 F1-score.....	47
4.4.7 Curva ROC (Receiver Operating Characteristic)	48
4.4.8 AUC (Area Under Curve)	48
5. Experimentos y resultados	49
5.1 EfficientNetB0 (base line)	49
5.2 EfficientNetB3.....	50
5.3 EfficientNetB4.....	51
5.4 EfficientNetB5.....	53
5.5 Selección del modelo final.....	54
6. Conclusiones.....	56
7. Líneas de trabajo futuro	58
8. Glosario	59
9. Bibliografía	60

Índice de figuras

Ilustración 1 - (a) radiografía paciente normal, (b) radiografía paciente con neumonía [6].	2
Ilustración 2 - Cross Industry Standard Process for Data Mining (CRISP-DM) [19].	9
Ilustración 3 - Planificación del proyecto.	11
Ilustración 4 - Número de publicaciones en PubMed con la consulta " <i>classification deep convolutional neural networks</i> ".	14
Ilustración 5 - Neurona biológica [43].	19
Ilustración 6 - Neurona artificial [44].	20
Ilustración 7 - Red neuronal artificial [45].	21
Ilustración 8 - Marco teórico sobre la red neuronal convolucional (RNC).	22
Ilustración 9 - Componentes generales de un modelo EfficientNet [49].	24
Ilustración 10 - Precisión <i>EfficientNet</i> vs otros modelos [49].	26
Ilustración 11 - " <i>RSNA Pneumonia Detection Challenge</i> " repositorio de <i>Kaggle</i> .	27
Ilustración 12 - Comando de descarga del repositorio.	28
Ilustración 13 - Descarga manual de los ficheros del repositorio (uno a uno).	28
Ilustración 14 - Descarga manual de los ficheros del repositorio (todos a la vez).	28
Ilustración 15 - <i>Dashboard</i> repositorio de <i>Kaggle</i> .	29
Ilustración 16 - Estructura del repositorio de <i>Kaggle</i> .	29
Ilustración 17 - Estructura del repositorio de <i>GitHub</i> .	30
Ilustración 18 - Estructura de " <i>stage_2_detailed_class_info.csv</i> ".	31
Ilustración 19 - Estructura de " <i>stage_2_train_labels.csv</i> ".	32
Ilustración 20 - Lectura de los datos.	33
Ilustración 21 - Número de registros iniciales vs actuales vs duplicados.	34
Ilustración 22 - Número de registros para el entrenamiento vs validación vs test.	35
Ilustración 23 - Número de imágenes sin neumonía vs con neumonía.	36
Ilustración 24 - Número de imágenes balanceadas sin neumonía vs con neumonía.	38
Ilustración 25 - Tamaño del <i>dataset</i> balanceado antes vs ahora.	38
Ilustración 26 - Imagen original vs imagen aumentada.	39
Ilustración 27 - Curva ROC.	48

Índice de tablas

Tabla 1 - Hitos del proyecto.....	12
Tabla 2 - Librerías utilizadas <i>Anaconda</i> vs <i>Google Colab</i>	17
Tabla 3 - Tipos de aprendizaje automático.....	18
Tabla 4 - Tamaños de las imágenes en <i>EfficientNet</i>	25
Tabla 5 - Número de parámetros en <i>EfficientNet</i>	25
Tabla 6 - Filtrado de las columnas.	34
Tabla 7 - Detalles de implementación del <i>ImageDataGenerator</i>	37
Tabla 8 - Comparativa del nombres original vs aumentada.	39
Tabla 9 - Parámetros <i>data augmentation</i> de los modelos.	44
Tabla 10 - Hiperparámetros de los modelos <i>transfer learning</i>	44
Tabla 11 - Hiperparámetros de los modelos <i>fine-tuning</i> (modelos finales).....	45
Tabla 12 - Matriz de confusión	46
Tabla 13 - Evolución de la exactitud y pérdida de <i>EfficientNetB0</i>	49
Tabla 14 - Resultados de <i>EfficientNetB0</i>	50
Tabla 15 - Evolución de la exactitud y pérdida de <i>EfficientNetB3</i>	51
Tabla 16 - Resultados de <i>EfficientNetB3</i>	51
Tabla 17 - Evolución de la exactitud y pérdida de <i>EfficientNetB4</i>	52
Tabla 18 - Resultados de <i>EfficientNetB4</i>	52
Tabla 19 - Evolución de la exactitud y pérdida de <i>EfficientNetB5</i>	53
Tabla 20 - Resultados de <i>EfficientNetB5</i>	53
Tabla 21 - Comparativa de modelos.	54

1. Introducción

1.1 Contexto y justificación del trabajo

La neumonía [1] es una infección del sistema respiratorio que afecta a los pulmones, es decir, puede generar una inflamación en los bronquiolos, bronquios y alvéolos pulmonares debido a microorganismos como hongos, bacterias y virus [2].

Aunque el término de neumonía está al día entre nosotros y sobre todo después del SARS-CoV2 [3], no somos conscientes de la importancia de esta enfermedad. Según Rosario Menéndez, neumóloga de la Sociedad Española de Neumología y Cirugía Torácica (SEPAR) [4], cito textualmente: *“La neumonía es la primera causa de muerte por infección, y no está reconocida como la enfermedad peligrosa que es ni por la comorbilidad que supone”*.

Un claro ejemplo de la importancia de las enfermedades respiratorias es como bien se ha mencionado el SARS-CoV2, o como comúnmente se conoce Covid-19.

Dicho virus ha generado una crisis global, tanto a nivel sanitario como a nivel económico. El principal peligro de esta enfermedad, u otras enfermedades respiratorias, es la facilidad con la que diferentes microorganismos se pueden trasladar de un ser vivo a otro, a partir del aire, del contacto con superficies infectadas...

Indagando aún más en la problemática sobre la neumonía, la Organización Mundial de la Salud (OMS) informa que el 15% de las defunciones entre los niños menores de 5 años están causadas por esta enfermedad [5], por lo tanto, este grupo de edad junto con las personas mayores de 60 años son los más afectados, debido al bajo sistema inmune que se presenta a esas edades.

A nivel nacional la problemática continúa, ya que solo en el año 2017 se ingresaron a 120000 personas dejando a su paso más de 10000 muertes [4], entrando España en el top de los países de la Unión Europea con más fallecidos por neumonía.

Entendiendo cómo de importante es esta enfermedad y las consecuencias de la misma, se ve de forma clara que es de vital importancia poder detectar y diagnosticar cuanto antes la neumonía, con el objetivo de salvar vidas.

Hasta el día de hoy, el diagnóstico de enfermedades pulmonares se puede hacer a partir de radiografías, resonancias magnéticas, broncoscopias, pruebas de esfuerzo... Todas ellas se complementan, pero por norma general se suele hacer uso de radiografías del tórax, ya que éstas junto con la opinión de un

experto es suficiente para diagnosticar si existe la enfermedad y en qué grado. En la siguiente imagen se puede observar la diferencia entre un paciente sano y uno con neumonía a partir de una radiografía de tórax:

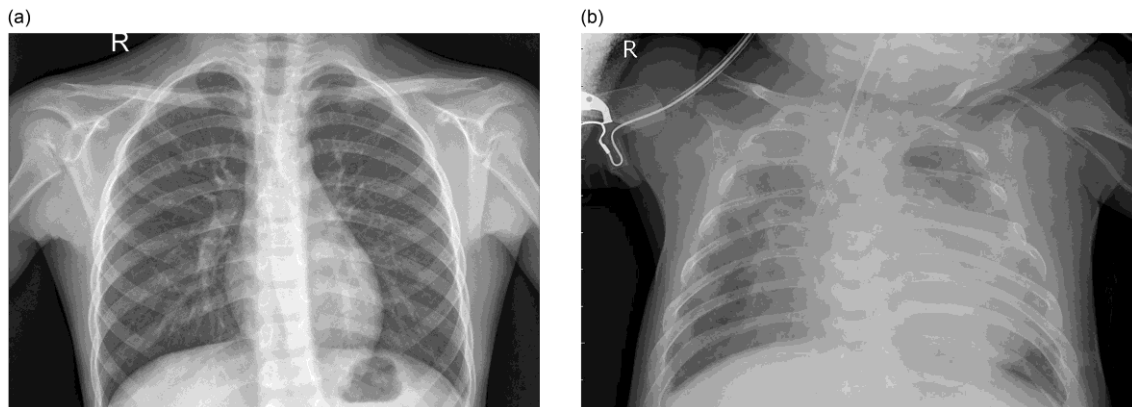


Ilustración 1 - (a) radiografía paciente normal, (b) radiografía paciente con neumonía [6].

El inconveniente en el diagnóstico de la neumonía es que se realiza de forma manual, en otras palabras, se necesita que un médico analice la radiografía para poder determinar si una persona padece de dicha enfermedad o no.

Por otro lado, hay que tener en cuenta el coste económico que supone, según la Organización Mundial de la Salud (OMS) el coste de antibióticos y pruebas necesarias para el diagnóstico de la neumonía solamente en niños supondría invertir 109 millones de dólares al año [5].

Ante este contexto surge la idea de este proyecto, el hacer uso del *deep learning*, y más concretamente de redes neuronales convolucionales (CNN), para poder diagnosticar si una persona padece neumonía o no sin la necesidad de la intervención médica, consiguiendo así reducir la saturación del sistema sanitario, además de reducir los costes al ser capaces de identificar la neumonía con antelación.

1.2 Marco teórico sobre la neumonía

Gracias a los diferentes avances que se han realizado en el ámbito de la salud y en el tecnológico, permiten mejorar tanto la calidad como la esperanza de vida de las personas. Aunque esto supone un gran cambio en la sociedad, la realidad es que seguimos teniendo muchas enfermedades en el día a día. Debido a esto se tiene que seguir mejorando las diferentes técnicas para poder detectar patologías y poder tratar las mismas de una forma adecuada, con el fin de mejorar tanto la esperanza como la calidad de vida.

Una de las enfermedades más comunes es la neumonía, ésta presenta una alta incidencia tanto en los países desarrollados como en los que no lo están, dejando a su paso miles de víctimas año tras año. Además de la mortalidad de la misma, ésta genera un gran impacto en la sociedad debido a la frecuencia con la que es diagnosticada, y el coste económico que supone detectar y tratar esta patología.

Por otro lado, hacer una correcta clasificación de esta enfermedad es de gran dificultad debido a que hay otras enfermedades pulmonares que presentan un cuadro similar, pero gracias a la tecnología y al personal médico se consigue identificar y tratar la misma de forma eficaz.

La neumonía es una lesión inflamatoria pulmonar como consecuencia de la llegada de microorganismos a la vía aérea distal y parénquima [7]. Para hacer un buen diagnóstico y tratamiento sobre ella, hay que tener presente el cómo evoluciona y cuáles son los síntomas que ésta presenta.

Aunque hay enfermedades pulmonares con un cuadro clínico similar, la neumonía se caracteriza porque suele presentar todos o algunos de los siguientes síntomas: tos, expectoración purulenta o herrumbrosa, disnea, dolor pleurítico y fiebre [7].

Viendo los síntomas anteriores vemos que éstos no son solo característicos de la neumonía, es más, según un estudio por tres médicos que desconocían el diagnóstico de 52 pacientes (24 con neumonía), manifestó que entre el 47% y el 69% de los casos con dicha patología fueron diagnosticados [7]. Como podemos apreciar el porcentaje de sensibilidad no es muy alto, y es aquí donde entra en juego el objetivo de este proyecto, hacer que a través del *deep learning* podamos diagnosticar más fácilmente esta enfermedad.

Por otro lado, tenemos que hablar de las diferentes neumonías que existen, y cómo éstas se clasifican dependiendo de diferentes factores.

Según el tipo de agente causal, entendiendo por el mismo cualquier sustancia viva o no, cuya presencia o ausencia determina la causa de una patología [8], presentamos la siguiente clasificación [7] [9]:

- Neumonía neumocócica: se caracteriza porque la neumonía tiene origen a partir de una bacteria denominada *Streptococo pneumoniae*, conocida comúnmente como neumococo [10].
- Neumonía estafilocócica: tiene como origen una bacteria denominada *Staphylococcus aureus*, o mayoritariamente conocido como estafilococo [11]. Es relativamente raro padecer este tipo de neumonía, aunque en época de epidemia suele aumentar la importancia.
- Neumonía por *Klebsiella*: se produce a partir de la bacteria *Klebsiella*, la cual tiene consecuencias graves sobre todo entre pacientes diabéticos y alcohólicos [12].
- Neumonía por *Legionella*: es originada también por una bacteria, en este caso la bacteria *Legionella pneumophila*, un elevado porcentaje de casos con este tipo de neumonía necesitan ser tratados en unidades de cuidados intensivos, dando lugar a una elevada mortalidad [13].

Otro tipo de clasificación es dependiendo de la afectación anatomopatológica, en este caso tenemos los siguientes tipos de neumonía [7] [9]:

- Neumonía alveolar: la zona afectada mayoritariamente son los alvéolos ya que la zona de los bronquiolos no suele presentar grandes daños. Esta neumonía es la manifestación de la neumonía neumocócica.
- Neumonía multifocal: afecta tanto a los alvéolos como a los bronquiolos, pero no suele afectar a todo un lóbulo como sucedía con la anterior neumonía. Este tipo de neumonía es el claro ejemplo de neumonía estafilocócica.
- Neumonía intersticial: en este caso los alvéolos y bronquiolos no se ven afectados, sino que la zona dañada es el intersticio.
- Neumonía necrotizante: este tipo de neumonía es una complicación grave debido a que produce necrosis, muerte de un conjunto de células en el tejido pulmonar, haciendo que la mortalidad en los niños que padecen este tipo de neumonía sea elevada. La neumonía necrotizante está asociada tanto a neumonías neumocócicas como estafilocócicas.

En cuanto a la reacción de la persona que padece de dicha enfermedad, se presenta la siguiente clasificación [9]:

- Neumonía supurada: aquella neumonía que tras su recuperación no deja secuelas.
- Neumonía fibrinosa: neumonía en la que no se recupera el paciente al cien por cien, ya que tras su “recuperación” le queda un exceso en el tejido pulmonar, o también conocido como fibrosis.

Dependiendo del tipo de huésped, la neumonía puede ser [7] [9]:

- Neumonía en paciente inmunocompetente: si es capaz de dar una respuesta inmunitaria de una forma normal.
- Neumonía en paciente inmunodeprimido: si no es capaz de dar una respuesta inmunitaria de forma normal, sino que necesita de ayuda médica para hacer frente a una enfermedad.

Por último, dependiendo de dónde se ha adquirido la bacteria que origina una neumonía [7] [9]:

- Neumonías extrahospitalarias: si la patología proviene de un microorganismo ajeno al ámbito hospitalario.
- Neumonías hospitalarias: si la neumonía proviene a partir de un microorganismo del ámbito hospitalario.

Aunque las anteriores clasificaciones son usadas, mayoritariamente se emplea una simplificación, haciendo una división de la neumonía en tres posibles casos [14]:

- Bronconeumonía: caracterizada inicialmente por la inflamación de los bronquios, posteriormente se extiende al resto del pulmón.
- Neumonía lobular: la inflamación se origina en la zona lobular, afectando después a todo el sistema respiratorio.
- Neumonía intersticial: la neumonía es causada por fibrosis pulmonar.

Como hemos podido apreciar en las anteriores clasificaciones hay muchas formas de categorizar la neumonía, en este proyecto nos vamos a abstraer de la neumonía como concepto científico, pero es importante tener en mente que a partir de diferentes factores podemos tener diferentes tipos de neumonía, no es un problema trivial.

Uno de los mayores problemas de esta enfermedad es la rápida evolución, haciendo que aumente el riesgo de mortalidad en aquellas personas más vulnerables.

Las fases que presenta la neumonía varían dependiendo de qué tipo es, es decir, si es bronconeumonía, neumonía lobular o neumonía intersticial. Tanto para la bronconeumonía como neumonía intersticial es difícil identificar de forma clara las fases de cómo evoluciona, sin embargo, no sucede lo mismo con la neumonía lobular.

En la neumonía lobular encontramos las siguientes cuatro fases [14]:

- Fase de congestión: es visible en las primeras horas tras la infección. Se caracteriza porque el tejido pulmonar presenta un aspecto pesado, además de haber poco glóbulos rojos y neutrófilos.
- Fase de hepatización roja: se hace presente entre dos y cuatro días después de la primera fase. En esta fase se enrojecen los pulmones de color rojo, debido al aumento de glóbulos rojos y neutrófilos.
- Fase de hepatización gris: se manifiesta entre dos y cuatro días de la fase anterior. En este caso, los glóbulos rojos se descomponen haciendo que el pulmón adquiera un color gris.
- Fase de resolución: después de entre cuatro y ocho días predominan los glóbulos blancos sobre las zonas afectadas, poniendo fin a la neumonía.

Tal y como se puede apreciar en la Ilustración 1 hay una clara evidencia entre un paciente con y sin neumonía, para ser más exactos, si hay presencia de una opacidad pulmonar (la nube de color blanco que cubre la cavidad pulmonar) estamos en el caso de un paciente que padece dicha enfermedad.

Aunque esta metodología puede ser usada para identificar la neumonía a partir de una radiografía, la realidad es otra.

Una opacidad pulmonar no indica que sea exclusivamente una neumonía, puede haber una pérdida de sangre, y que esto genere dicha opacidad. También puede ser causada por un cáncer de pulmón, lesiones pulmonares...

Como podemos observar para una misma entrada, una radiografía de tórax, podemos tener diferentes salidas, entendiendo por salidas todas las posibles enfermedades que están originando dicha opacidad pulmonar.

La identificación de la neumonía a partir exclusivamente de una radiografía no es una tarea sencilla, es más, hay casos en los que la radiografía más la ayuda de un médico no resuelve la problemática y se necesitan más pruebas, como por ejemplo:

- Prueba de función pulmonar.
- Análisis de sangre.
- Biopsia.
- Broncoscopía.
- Tomografía.
- Electrocardiografía.
- Ecocardiografía.
- Entre otras...

Por lo tanto, en el siguiente apartado se va a explicar de forma resumida cuál es la problemática a la que nos enfrentamos y cómo se va a resolver.

1.3 Resumen de la problemática

1.3.1 Definición del problema

La neumonía es una de las enfermedades que más está presente en nuestro día, dando lugar a miles de muertes año tras año. Además, el proceso de identificar la misma no es una tarea fácil como ya se ha visto, por lo tanto el primer paso que se realiza en el ámbito sanitario es hacer una radiografía del tórax del paciente.

Gracias a los grandes avances que se han realizado en el ámbito de la salud y en el tecnológico, se puede mejorar la esperanza de vida de las personas que padecen de neumonía, haciendo un rápido diagnóstico que nos permita determinar si el paciente presenta síntomas de dicha patología.

Sabiendo de la importancia de esta enfermedad, y que las radiografías suelen ser el primer método usado para identificar la misma; nos vemos en la necesidad de crear un modelo capaz de solventar la identificación de neumonías a partir de imágenes (radiografías), con el fin de mejorar el diagnóstico de dicha patología.

1.3.2 Solución

Para la solución de este proyecto se ha hecho uso de uno de los principales algoritmos de aprendizaje profundo, las redes neuronales convolucionales. El aprendizaje profundo son redes neuronales artificiales, las cuales presentan muchas capas, haciendo de éstas un sistema complejo que permiten resolver problemas como la clasificación de imágenes, audio, texto...

Las redes neuronales convolucionales se caracterizan porque buscan imitar el ojo humano, haciendo que a partir de imágenes sean capaces de extraer características. Es por ello que se ha elegido este algoritmo de aprendizaje profundo frente a otros (redes neuronales recurrentes, máquina de Boltzmann profunda, redes de memoria a corto plazo...).

Por lo tanto, se ha definido un modelo de red neuronal convolucional para clasificar si una radiografía de tórax presenta neumonía o no, es decir, tenemos dos clases. Los datos necesarios para entrenar la red se encuentran en el repositorio de *Kaggle* [15].

Buscando la continua mejora del modelo se han realizado diferentes técnicas, como: aumento de datos, *transfer learning* y *fine-tuning* [16].

Para la realización de *transfer learning* se ha hecho uso de la redes neuronales convoluciones pre-entrenadas *EfficientNet*, pertenecientes a *Google*.

1.4 Motivación personal

El motivo principal por el que escogí este trabajo es básicamente por el interés personal en el ámbito sanitario. Considero que a día de hoy, existen muchas enfermedades tanto en países desarrollados como no desarrollados, en los que hacer uso de la tecnología permitirá resolver de una forma más eficiente el diagnóstico de enfermedades, consiguiendo así mejorar la calidad de vida de la sociedad.

Debido a ese motivo me parece de vital importancia proyectos como este o similares, en los que se busca investigar cómo afecta una enfermedad y a partir de redes neuronales poder diagnosticar si una persona padece neumonía o no, consiguiendo así reducir tiempos, o dicho en otras palabras, reducir el riesgo de mortalidad que conlleva una enfermedad.

Por otro lado, considero que el conocimiento que voy a obtener desarrollando este proyecto me va a aportar grandes cualidades para el mundo laboral,

principalmente porque el *deep learning* está visible en una gran variedad de proyectos.

1.5 Objetivos del trabajo

La hipótesis u objetivo principal de este trabajo es la generación de un modelo que permita determinar si una persona padece neumonía, a partir de radiografías de tórax.

Con el fin de alcanzar dicho objetivo, se plantean los siguientes objetivos secundarios:

- Entender la problemática de la neumonía, qué la causa, cuáles son las consecuencias, cómo se diagnostica a día de hoy, y el estado del arte en esta área.
- Obtener los datos del repositorio en *Kaggle* [15], entendiendo por datos las imágenes médicas en formato *DCM* [17].
- Realizar un preprocesado de los datos, para poder enviar a la red neuronal la información que necesita y mejorar la precisión de la misma.
- Creación de la red neuronal convolucional, encargada de recibir la imagen como entrada y capaz de clasificar si la persona padece neumonía o no.
- Evaluación del modelo, determinar cómo de bueno es para dar por alcanzado el objetivo principal.
- Búsqueda de los hiperparámetros idóneos, con el fin de mejorar la solución.
- Generar la documentación técnica del proyecto.
- Presentar y defender el proyecto ante un tribunal.

1.6 Enfoque y método seguido

A partir de las radiografías de tórax se quiere obtener un conocimiento para determinar si una persona tiene neumonía, para ello se va a hacer uso de una red neuronal convolucional, que use la imagen como entrada y nos devuelva si hay presencia de dicha enfermedad.

Tal y como se puede apreciar en el objetivo del proyecto, estamos en un caso de minería de datos. Por lo tanto, se va a hacer uso de la metodología *Cross Industry Standard Process for Data Mining (CRISP-DM)* [18], ya que ésta es una metodología estándar y es utilizada en un gran número de proyectos de minería de datos.

Cuando hablamos de metodología nos referimos a un conjunto de procedimientos para alcanzar un determinado objetivo, el hacer uso de una metodología no garantiza el éxito pero sí disminuye las probabilidades de fracaso.

CRISP-DM se caracteriza porque sigue el ciclo de vida de un proyecto de minería de datos, permitiendo que podamos retroceder o avanzar de una fase a otra sin complicaciones, lo contrario a lo que sucede en un modelo de cascada. Además, se parte de la premisa que el proyecto no acaba cuando se encuentra un modelo idóneo, sino que éste hay que mantenerlo, documentarlo y mejorarlo ya que puede ser usado en proyectos futuros.

En esta metodología encontramos seis fases [19], las cuales se detallan a continuación:

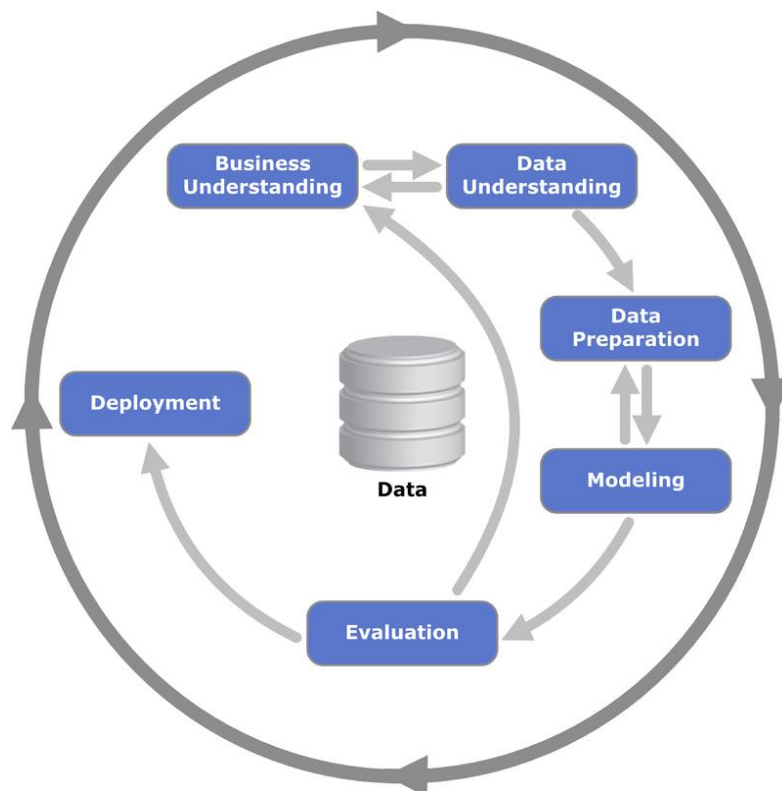


Ilustración 2 - Cross Industry Standard Process for Data Mining (CRISP-DM) [19].

1. Comprensión del negocio, en esta primera fase se busca definir los objetivos del proyecto. Para ello hay que analizar la situación en la que nos encontramos, lo que se busca resolver, qué tareas se van a realizar... Todo ello quedará definido en el apartado 1.5 y 1.7 de este documento.
2. Comprensión de los datos, posterior a la comprensión del negocio tenemos que hacer frente a qué datos tenemos, cómo son los mismos... En nuestro caso, principalmente vamos a trabajar con imágenes, que éstas son las radiografías de tórax.

3. Preparación de los datos, una vez que ya tenemos las radiografías hay que prepararlas para poder introducirlas a la red neuronal, escalando la imagen, realizando un aumento de datos...
4. Modelado, en esta fase nos encargaremos de definir la red neuronal convolucional, la cual tiene que ser capaz de resolver el problema para así poder alcanzar nuestro objetivo, en nuestro caso, diagnosticar si el paciente padece neumonía.
5. Evaluación del modelo, en ella se busca saber cómo de bueno es nuestro modelo a partir de diferentes medidas. Si la calidad del modelo no es elevada, hay que retroceder a pasos anteriores.
6. Despliegue, en esta fase el objetivo es poder transmitir de forma adecuada la conocimiento obtenido, es decir, explicar de forma comprensible cómo de bueno es el modelo generado para diagnosticar una neumonía, el resultado de este punto es el presente documento.

1.7 Planificación del trabajo

En el diagrama de *Gantt* que se presenta en la siguiente página, podemos apreciar el cómo se han definido las diferentes fases del proyecto junto con las fechas límites para entregar cada práctica de evaluación continua.

Cabe destacar que cada fase tiene un color diferente, el color azul representa las tareas definidas en la fase “Definición del proyecto”, el color verde indica las tareas de la fase “Estado del arte”, el color morado hace referencia a la fase “Diseño e implementación del trabajo”, el color naranja nos indica “Redacción de la memoria” y el color granate es la fase de “Defensa”.

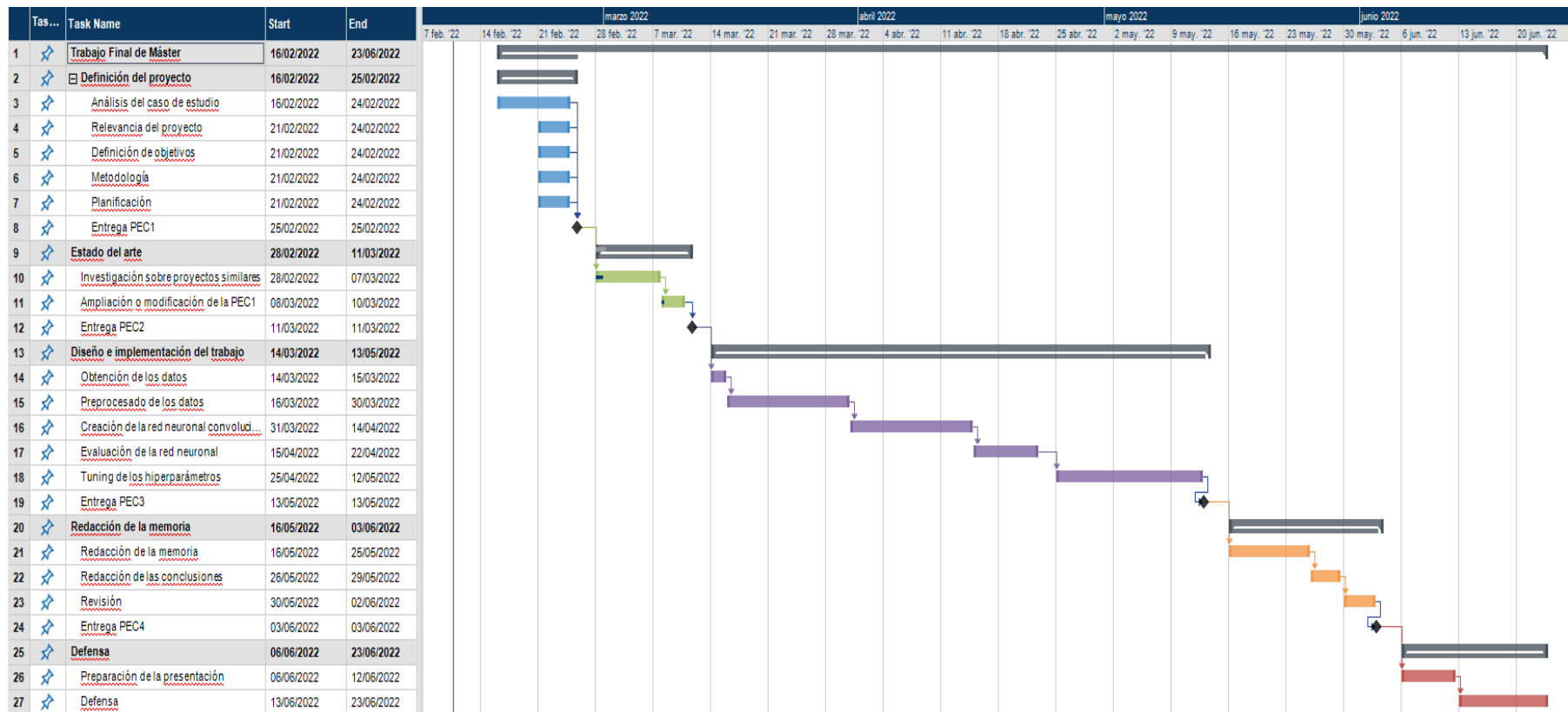


Ilustración 3 - Planificación del proyecto.

Los hitos u objetivos para dar por concluido este proyecto son los siguientes:

	Entregable	Fecha
Hito 1	PEC1	25/02/2022
Hito 2	PEC2	11/03/2022
Hito 3	PEC3	13/05/2022
Hito 4	PEC4	03/06/2022
Hito 5	Defensa	23/06/2022

Tabla 1 - Hitos del proyecto.

1.8 Breve resumen de productos obtenidos

Los productos obtenidos tras la realización del proyecto son:

- La memoria, el presente documento, se refleja toda la información relativa al desarrollo del proyecto, desde el contexto, objetivos, metodología utilizada, estado del arte, resultados obtenidos, conclusiones y líneas futuras.
- El repositorio de *GitHub* [20], en él se encuentra el código desarrollado para alcanzar los objetivos del proyecto, la documentación e imágenes usadas.

1.9 Breve descripción de los otros capítulos de la memoria

En los próximos capítulos de la memoria encontraremos:

- Estado del arte, en él se va a realizar una breve explicación sobre diferentes artículos/proyectos que presentan objetivos similares al nuestro. Además, se va a detallar las diferentes tecnologías utilizadas para el desarrollo.
- Marco teórico, en este apartado se explicará de forma general el método utilizado para alcanzar el objetivo, es decir, la técnica de *machine learning* usada. Además, se hará una breve referencia a las diferentes redes usadas de *EfficientNet*.
- Desarrollo, aquí se detallará cada una de las fases necesarias sobre cómo se ha desarrollado el proyecto, qué decisiones se han ido tomando y por qué se tomaron.

- Experimentos y resultados, en este capítulo se recogerá de forma resumida qué resultados hemos obtenido en cada uno de los modelos.
- Conclusiones, en base a los resultados obtenidos obtendremos diversas deducciones, las cuales serán usadas para conformar la conclusión del proyecto.
- Líneas de trabajo futuro, se hará mención a los posibles caminos que nos podremos encontrar dentro del *machine learning* en el futuro, y su aplicación al ámbito sanitario.

2. Estado del arte

2.1 Estudios similares

El ámbito de la medicina siempre ha sido objeto de estudio, principalmente para mejorar la calidad y la esperanza de vida de las personas. Esto se ha conseguido gracias a los avances tanto en investigaciones medicinales como en la mejoría de la tecnología usada.

En la última década ha aparecido un nuevo factor, el *deep learning*, haciendo que los ordenadores sean capaces de aprender, hasta tal punto que nos permiten determinar si una persona padece una enfermedad, cómo prevenir la misma, qué línea de tratamiento seguir...

Por lo tanto, debido al auge que hay en la actualidad respecto al ámbito medicinal y tecnológico, ha hecho que haya numerosas investigaciones publicadas, en cuanto a clasificación de enfermedades a partir de imágenes con redes neuronales artificiales. En esta sección vamos a analizar algunas de las líneas de investigación en esta área.

Para visualizar cómo de importante está siendo este campo en los últimos años podemos hacer una búsqueda en *PubMed* [21], la cual es una base de datos que contiene millones de referencias bibliográficas sobre investigaciones en la ciencia de la salud, con la consulta "*classification deep convolutional neural networks*" podemos ver el aumento que ha habido en los últimos años, sobre todo en el año 2020 bien conocido como año del coronavirus.

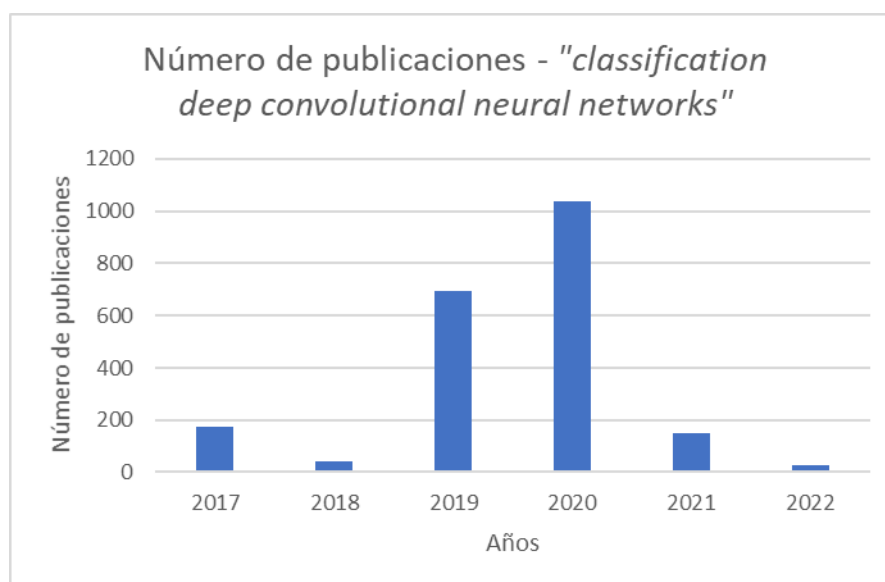


Ilustración 4 - Número de publicaciones en PubMed con la consulta "*classification deep convolutional neural networks*".

Un ejemplo del estado del arte en esta área sería el de *Samir S. Yadav* y colaboradores [22] publicado en 2019. En esta investigación se busca clasificar imágenes médicas a partir de redes neuronales artificiales y de clasificadores de máquinas de soporte, principalmente se hace esto porque se considera que se ha alcanzado el techo con los métodos tradicionales en cuanto a rendimiento. Para poder resolver esta problemática se hace uso de dos redes neuronales convolucionales y de clasificadores de máquinas de soporte, y el objetivo es clasificar si un paciente presenta neumonía o no a partir de radiografías de tórax. El resultado de este estudio muestra que el aumento de datos mejora el rendimiento a la hora de clasificar en cualquier método, pero por otro lado, ofrece una mayor precisión una red neuronal convolucional que un clasificador de máquina de soporte, ya que se alcanza una precisión del 89% y del 77% respectivamente.

Otra de las investigaciones publicadas corresponde a *Rahib H. Abiyev* y colaboradores [23] publicado en 2021. El objetivo de esta investigación es poder diagnosticar si una persona padece *SARS-Covid2* y neumonía a partir de radiografías de tórax. Para resolver esta situación se hace uso de redes neuronales convolucionales, para ser más exactos se hace uso de dos redes neuronales convolucionales entrenadas con diferentes conjuntos de datos. Uno de los modelos fue entrenado con una clasificación binaria (si el paciente presenta *COVID-19* o no), mientras que el otro modelo se “nutría” del conocimiento del primer modelo (transferencia del aprendizaje). El resultado obtenido es que el modelo final consigue una precisión del 98%, haciendo que a partir de una radiografía de tórax se pueda clasificar casi con certeza si un paciente presenta neumonía y por lo tanto tiene el virus *SARS-Covid2*.

Si nos centramos en la clasificación de neumonías a partir de radiografías de tórax, pero sin hacer uso de redes neuronales convolucionales, tenemos la línea de investigación de *Andrés Felipe Romero* y colaboradores [24] publicada en 2020. En este estudio se realiza una segmentación de la imagen de forma manual para extraer características, posteriormente éstas se introducen a un clasificador *k-means* permitiendo así determinar si un paciente padece neumonía o no. Los resultados obtenidos son una especificidad del 76% y una sensibilidad del 83%, se consiguen unos resultados decentes sin necesidad de una red neuronal convolucional.

Otra estudio publicado sobre la clasificación de neumonías es el de *Hicham Moujahid* y colaboradores [25] publicado en 2020. En este estudio el objetivo sigue siendo el mismo, pero para resolver la problemática se hace uso de redes neuronales convolucionales. Cabe destacar que para mejorar la precisión utiliza la técnica de transferencia de aprendizaje de una red neuronal a otra, tal y como se ha indicado en estudios previos. Gracias a las redes neuronales convolucionales y a la técnica de transferencia de aprendizaje se consigue una precisión del 96%.

El último estudio que se presenta es de *Lingzhi Kong* y colaboradores [26] publicado en 2021. En este caso se busca clasificar si un paciente infantil presenta indicios de neumonía. Para ello se hace uso de una red neuronal

convolucional profunda, utilizando *Xception* y una memoria tanto a corto plazo como a largo plazo (*LSTM*). *Xception* es el primer modelo, y éste se encarga de extraer las características de los datos, posteriormente *LSTM* es alimentado a partir de la salida de *Xception*, haciendo que *LSTM* seleccione las características más importantes. La precisión alcanzada con este modelo es del 96%.

En resumen, como podemos apreciar hay numerosas líneas de investigación dentro de la ciencia de la salud, ya sean para diagnosticar cáncer, enfermedades pulmonares, genéticas... Viendo la gran cantidad de estudios que se están realizando, nos podemos hacer una idea de cómo de importante está siendo esta área. Debido a esto, en los siguientes capítulos vamos a hablar de nuestra solución a esta problemática y así aportar nuestro granito de arena.

2.2 Tecnologías usadas

2.2.1 Entornos de trabajo

Para desarrollar el código necesario para la realización de este proyecto se han necesitado de dos entornos de trabajo diferentes.

El primero fue usado para el preprocesado de los datos, éste se realizó de forma local. El ordenador presenta un procesador *Intel i7-8750H* y una memoria *RAM* de 16GB, el sistema operativo es *Windows 10 Pro* de 64bits.

El segundo entorno de trabajo fue usado para la creación de los modelos de *machine learning*, éste se realizó *online* gracias a la plataforma *Google Colab*. Cabe destacar que para obtener mayores recursos de memoria, disco, accesos a *GPUs* más rápidas y tiempos de ejecución más largos, se decidió cambiar del plan “gratis” a la versión “*Colab Pro*”.

2.2.2 Lenguaje de programación

El lenguaje de programación utilizado para todo el desarrollo del proyecto ha sido *Python*, para los dos entornos descritos en el punto anterior.

Python es un lenguaje de programación de alto nivel, el cual no es interpretado como *Java* o *C*, por lo que no es necesario compilar el código [27].

La elección de este lenguaje viene determinada por dos motivos, el primero es que es sencillo escribir e interpretar el código, es decir, tiene una sintaxis simple, legible y fácil de entender. El segundo motivo es porque dentro la comunidad científica tanto *Python* como *R* son los lenguajes estándar, en nuestro caso, *Python* suele ser utilizado cuando se quiere realizar un procesamiento de imágenes.

2.2.3 Entornos de programación

Al igual que tenemos dos entornos de trabajo, sucede lo mismo con los entornos de programación.

Para el entorno local se ha usado *Anaconda* [28], ésta es una distribución de aplicaciones tanto en *Python* como en *R* para facilitar las labores de los científicos de datos. La versión de *Python* usada en *Anaconda* es v3.7.4.

Por otro lado, para el entorno de *Google Colab* no se ha tenido que instalar nada más para hacer uso de *Python*, ya que *Google Colab* es un servicio en la nube, el cual ya viene preinstalado *Python* junto a diferentes librerías. En este caso, la versión usada de *Python* es v3.7.13.

2.2.4 Librerías utilizadas

Librerías	<i>Anaconda</i>	<i>Google Colab</i>
<i>Numpy</i> [29]	✓	✓
<i>Pandas</i> [30]	✓	✓
<i>Os</i> [31]	✓	✓
<i>Pydicom</i> [32]	✓	x
<i>PIL</i> [33]	✓	✓
<i>Random</i> [34]	x	✓
<i>_pickle</i> [35]	x	✓
<i>Skimage</i> [36]	✓	✓
<i>Sklearn</i> [37]	✓	✓
<i>Seaborn</i> [38]	x	✓
<i>Tensorflow</i> [39]	✓	✓
<i>Keras</i> [40]	✓	✓
<i>Matplotlib</i> [41]	✓	✓

Tabla 2 - Librerías utilizadas *Anaconda* vs *Google Colab*.

3. Marco teórico sobre las técnicas utilizadas

3.1 Aprendizaje automático

El aprendizaje automático, o también conocido como *machine learning*, es una rama de la inteligencia artificial en el que se busca que los ordenadores sean capaces de aprender sin necesidad de ser programados, en otras palabras, tienen que ser capaces de aprender por sí mismos [42].

El objetivo es el dotar de conocimiento a los ordenadores a partir de la identificación de patrones, de tal forma que sean capaces de extraer inferencias de nuevos datos con los que no han sido entrenados.

A día de hoy este término está en la boca de todos debido al uso que se empezó a hacer por grandes compañías, las conocidas como *FANG* (*Facebook, Amazon, Netflix y Google*). Pero la realidad es que esta tecnología ya se inventó en el siglo pasado, comenzando a finales de los años 50 con la creación de la red neuronal artificial, el perceptrón.

El *machine learning* presenta tres tipos de aprendizaje: aprendizaje por refuerzo, aprendizaje supervisado y aprendizaje no supervisado.

Tipos de aprendizaje automático

Aprendizaje
supervisado

Aprendizaje
no supervisado

Aprendizaje
por refuerzo

Tabla 3 - Tipos de aprendizaje automático.

Denominamos al aprendizaje automático aprendizaje supervisado cuando los datos de entrenamiento son etiquetados, es decir, para cada observación conocemos la variable objetivo. Un ejemplo sería este proyecto, para cada radiografía de tórax conocemos si hay o no neumonía.

Cuando los datos no están etiquetados nos encontramos en el aprendizaje no supervisado, en este caso se desconoce la variable objetivo, por lo que la máquina tiene que ser capaz de identificar similitudes en los datos. Un ejemplo de este tipo de aprendizaje sería el reconocimiento facial, lo que se busca es identificar similitudes entre una persona y los datos biométricos que hay almacenados de ella, es decir, no se busca unos determinados rasgos.

El último aprendizaje es el aprendizaje por refuerzo, éste es totalmente distinto a los anteriores, en este caso la máquina aprende a partir de prueba y error. En

otras palabras, la máquina (agente) interacciona con el entorno y va obteniendo recompensas, de tal forma que a medida que va obteniendo unas mejores recompensas el agente es capaz de alcanzar el objetivo de una determinada tarea.

En la actualidad se manejan y almacenan una gran multitud de datos, es por ello que estos modelos permiten una mayor flexibilidad y adaptación en el mundo empresarial, haciendo de este campo una piedra angular sobre la evolución del ámbito tecnológico.

3.2 Aprendizaje profundo

El aprendizaje profundo o *deep learning* es un área dentro del *machine learning* caracterizada por hacer uso de redes neuronales artificiales, pero en este caso se usan múltiples capas, de ahí el *deep*.

La idea del *deep learning* viene asociada al funcionamiento del cerebro humano, el cómo las neuronas transmiten impulsos para activarse y así dicho órgano sea capaz de realizar una determinada tarea.

La unidad mínima dentro de una red neuronal biológica es la neurona, ésta presenta la siguiente estructura:

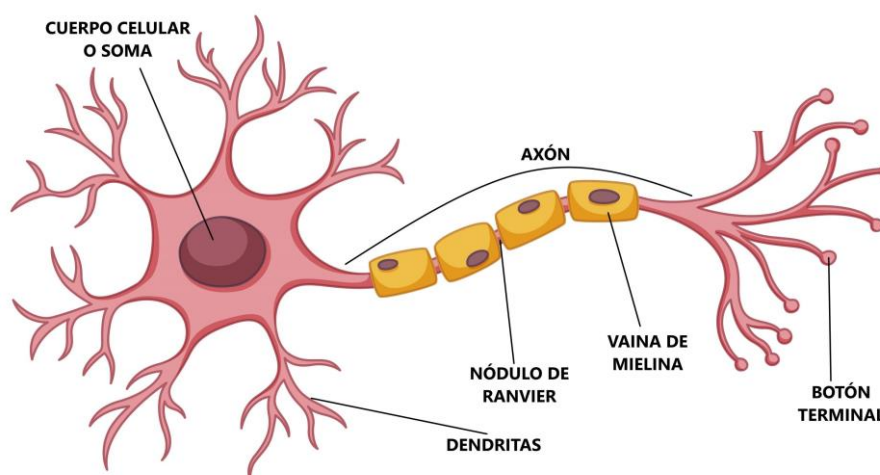


Ilustración 5 - Neurona biológica [43].

Recordando cómo funciona una neurona biológica, básicamente la entrada a la neurona (impulso nervioso que recibe) se produce gracias a las dendritas, una vez recibido dicho impulso en el núcleo de la neurona se computa, es decir, si tiene que activarse la neurona o no. En el caso de activarse se transmite de nuevo un impulso nervioso a las dendritas de la siguiente neurona a través del axón.

El hecho de tener una sola neurona no hace que el cerebro humano sea capaz de realizar tareas, sino que la clave es la red de neuronas. Un conjunto de millones de neuronas permite al ser humano realizar infinidad de tareas.

Volviendo al *deep learning* el objetivo es el mismo, es decir, la unidad mínima de cómputo va a ser la neurona, pero una neurona por sí misma no es de gran utilidad para realizar diversas tareas, es por ello que se definen múltiples neuronas.

La estructura de una neurona artificial es la siguiente:

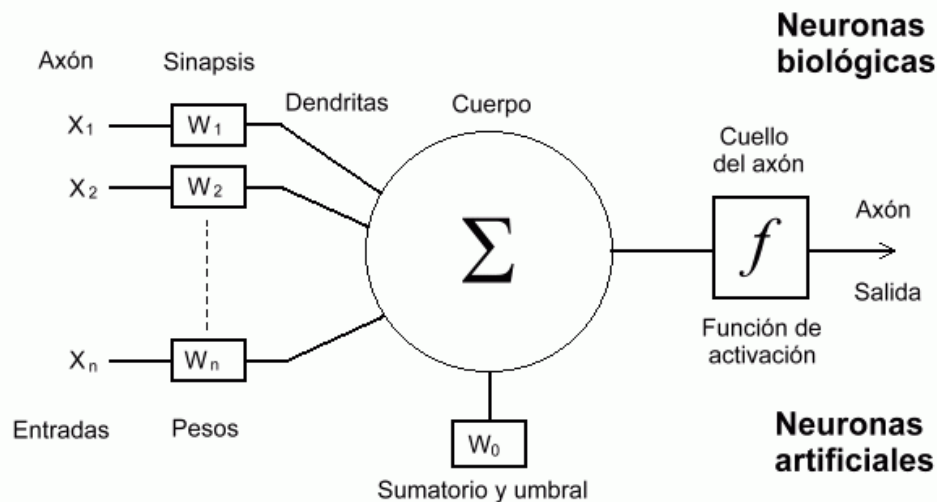


Ilustración 6 - Neurona artificial [44].

Como podemos apreciar hay una gran similitud entre una neurona artificial y biológica.

En este caso, la entrada a la red son un conjunto de valores que aplicando la ponderación de cada uno de los pesos a cada entrada da como resultado la señal de entrada.

Dependiendo de la señal de entrada la neurona artificial se activará o no, pero esto se decide en la función de activación f . Si se activa se emite la señal de salida a través del axón, como si de una neurona biológica se tratara.

Las redes neuronales artificiales presentan tres características fundamentales:

- La función de entrada.
- La función de activación de cada neurona,
- La arquitectura o topología de la red.

Dentro de la función de entrada, la cual es usada para combinar cada una de las entradas con cada uno de los pesos, las más utilizadas son: la función suma ponderada, la función máximo, la función mínimo y la función lógica *and* o *or*.

Por otro lado, la función de activación es la encargada de transformar la señal de entrada en la señal de salida, es decir, dependiendo de si cumple x condición la neurona se activará, generando así la salida correspondiente.

Hay diversas funciones de activación, pero las más usadas son: la función escalón, la función lineal, la función sigmoide, la función hiperbólica y la función rectificadora.

Finalmente se va a detallar la topología de la red neuronal, ésta es fundamental para el aprendizaje de la misma. Dentro de la topología de una red hay muchas características pero las elementales son:

- El número de capas.
- El número de neuronas por capa.
- La propagación hacia atrás de la información.

Definir correctamente esas tres características va a hacer que la red neuronal tenga una gran capacidad de cómputo, haciendo que sea muy buena en aprender por ella misma y poder resolver una tarea de forma adecuada.

Una red neuronal artificial presenta una arquitectura similar a la siguiente ilustración:

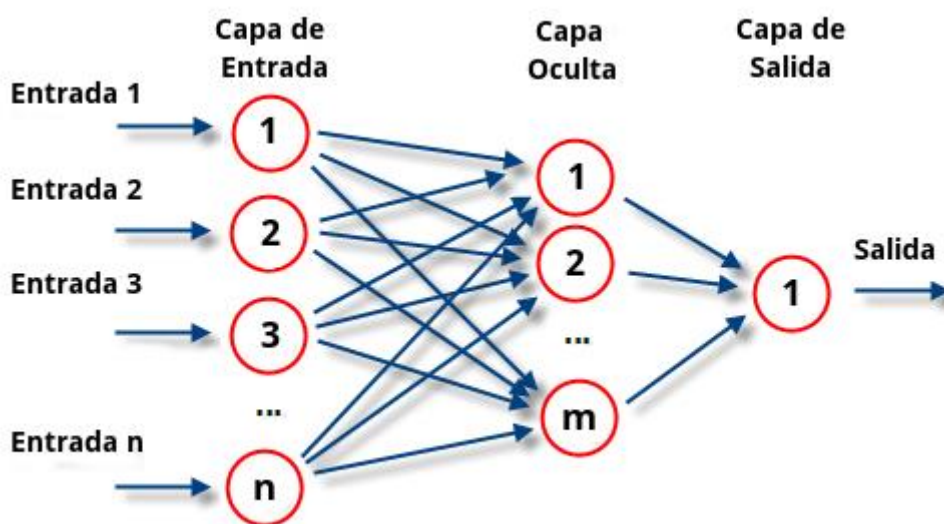


Ilustración 7 - Red neuronal artificial [45].

Cada nodo (círculo rojo) representa una neurona, y cada neurona apilada forma una capa. Dependiendo de si hay solo una capa oculta la red será monocapa, pero si hay varias será multicapa.

Tal y como se puede apreciar en la anterior ilustración hay tres tipos de capas:

- Capa de entrada, formada por las neuronas encargadas de recibir los datos.
- Capa de salida, ésta se encarga de proporcionar la salida generada por la red.

- Capa oculta, formada por muchas neuronas, las cuales son las encargadas de procesar todos los datos de la capa anterior y transmitir los resultados a la capa siguiente.

3.3 Redes neuronales convolucionales

La red neuronal convolucional (RNC) no es más que un tipo de red neuronal artificial, en otras palabras, es una técnica de *deep learning* [46]. Se caracteriza porque la RNC no necesita que se le introduzca datos, tal y como sucede en otras redes para solventar problemas de clasificación o regresión, sino que ésta es capaz de extraer características por sí sola.

Al conseguir extraer características por sí misma, hace que este tipo de red sea muy usada para encontrar patrones en imágenes, es decir, reconocer caras, personas, animales, coches, matrículas, números... Por lo tanto, el uso de este tipo de red neuronal ha aumentado en los últimos años en el ámbito de la inteligencia artificial, haciendo un mayor uso en los campos de la medicina y en la visión artificial.

Cabe destacar que aunque el tratamiento de imágenes es la principal utilidad que se le da a las RNC, éstas también son capaces de clasificar datos de audio, series temporales...

En resumen, las RNC destacan básicamente por dos motivos: el primero es que consiguen extraer características por ellas mismas, y el segundo porque son capaces de reconocer patrones con una alta precisión.

Al igual que sucede con una red neuronal artificial común, las RNC también presentan una capa de entrada y una capa de salida, es decir, el *input* de la red va a ser una imagen y el *output* la clasificación del mismo. Sin embargo, en lo que difieren está en las capas intermedias entre la entrada y la salida de la red.

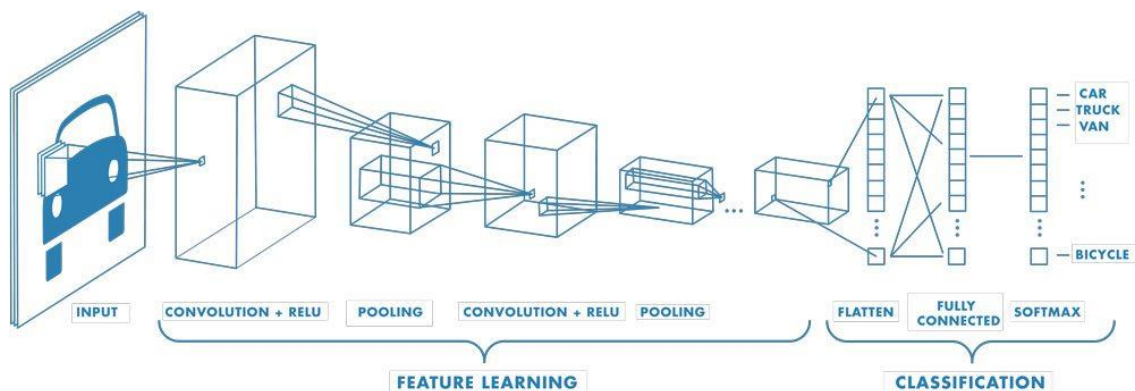


Ilustración 8 - Marco teórico sobre la red neuronal convolucional (RNC).

Como podemos apreciar en la anterior ilustración, una red neuronal convolucional la podemos dividir en la parte de aprendizaje de características y en la de clasificación.

Dentro de la parte de aprendizaje de características se encuentran las siguientes capas:

- Capa de convolución: es la encargada de extraer características a partir de la imagen de entrada, gracias a los filtros convolucionales.
- Capa unidad lineal rectificada (ReLU) o función de activación: esta capa permite que solamente las características activas pasen a la siguiente fase, ya que ReLU transforma los valores negativos a cero y los valores positivos toma el valor máximo entre cero y el valor correspondiente.
- Pooling: simplifica la fase de convolución generalizando los valores, para ello por norma general se coge de un conjunto de píxeles el valor máximo.

Una característica importante de las RNC, es que las capas mencionadas anteriormente se pueden repetir tantas veces como queramos. Suele servir de gran ayuda para mejorar la precisión a la hora de clasificar.

La otra parte de la RNC es la red en sí, es decir, la que nos permite la clasificación. Dentro de ésta tenemos las siguientes capas:

- Flatening: es la capa encargada de transformar la matriz que genera la capa anterior (*pooling*), a un vector que sirva como entrada para la red neuronal.
- Red neuronal (*fully-connected*): se trata de la propia red en sí, como si fuera un red neuronal artificial común, con la característica de que todas las neuronas de una capa tienen que estar conectadas con todas las neuronas de la siguiente capa.

La función de activación: se puede utilizar tanto la función *softmax*, la función sigmoide... Ésta nos permite dar la salida de la red neuronal y completar así la tarea de clasificación.

3.4 Redes neuronales convolucionales pre-entrenadas *EfficientNet*

Uno de los mayores problemas a la hora de entrenar una red neuronal convolucional es el elevado consumo que se necesita para entrenar la misma [47]. Con la idea de solventar este problema aparece el concepto de redes pre-entrenadas, la técnica que hace uso de las redes pre-entrenadas se denomina *transfer learning* [16].

Cuando se realiza el entrenamiento de la red neuronal se calculan los diferentes pesos para realizar la clasificación, el objetivo del entrenamiento es conseguir los pesos óptimos para resolver el problema que se quiere solventar.

Alcanzar los pesos óptimos no es una tarea fácil, y éstos determinan el cómo se comporta la red, es decir, dependiendo de qué pesos se hayan alcanzado la precisión de la red neuronal artificial va a ser mejor o peor.

Las redes pre-entrenadas se caracterizan porque una vez se ha realizado el entrenamiento para un problema se guarda dicha red, en otras palabras, se almacenan los pesos “óptimos” y la arquitectura en sí.

Realizando este proceso se consigue solventar el problema del coste computacional, además de reducir el tiempo necesario para entrenar la red.

Existen numerosas redes pre-entrenadas que nos permiten mejorar la precisión de nuestros modelos de una forma rápida y eficiente. La red pre-entrenada que se ha hecho uso en este proyecto es *EfficientNet* [48], perteneciente a la empresa *Google*.

EfficientNet es el nombre con el que se conoce a un conjunto de redes pre-entrenadas, no solamente a una. Para ser más exactos hay ocho modelos *EfficientNet*: *EfficientNet-B0*, *EfficientNet-B1*, *EfficientNet-B2*, *EfficientNet-B3*, *EfficientNet-B4*, *EfficientNet-B5*, *EfficientNet-B6* y *EfficientNet-B7*.

Todas estas redes neuronales convolucionales pre-entrenadas son similares en cuanto a la estructura de la red, es decir, presentan los mismos componentes generales. Sin embargo, varían en el número de capas dentro de cada componente general, a medida que aumenta el modelo de *EfficientNet* aumenta el número de parámetros a entrenar de la red.

Los componentes que contienen todos los modelos de *EfficientNet* son los que se aprecian en la siguiente ilustración:

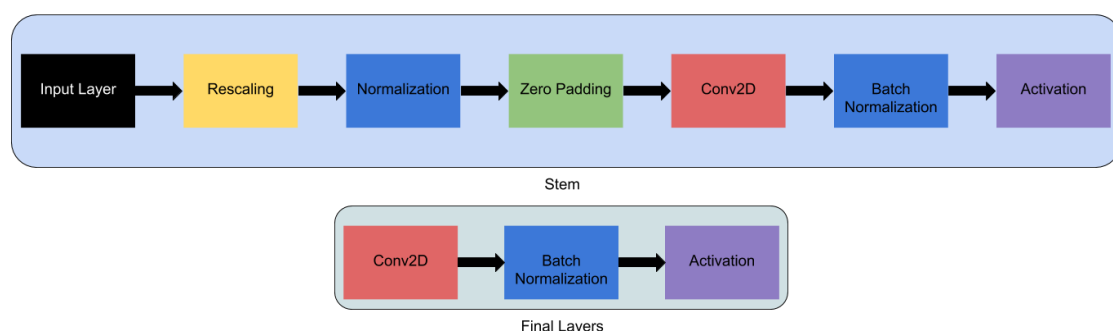


Ilustración 9 – Componentes generales de un modelo EfficientNet [49].

Cabe destacar que para realizar la técnica de *transfer learning*, la cual se explica en el punto 4.3.3, solamente se entrenan los componentes finales (*final layers*). Mientras que cuando se aplica la técnica de *fine-tuning*, explicada en el punto 4.3.4, sí que se entrenan todos los componentes (*stem* y *final layers*).

Otro punto a tener en cuenta en el uso de un determinado modelo de *EfficientNet*, es el tamaño de imagen que necesita la red. Para la realización de este proyecto se han creado modelos para *EfficientNet-B0*, *EfficientNet-B3*,

EfficientNet-B4 y *EfficientNet-B5*, los tamaños de imagen de cada modelo se pueden observar en la siguiente tabla:

Red neuronal	Tamaño de las imágenes
EfficientNetB0	224x224
EfficientNetB3	300x300
EfficientNetB4	380x380
EfficientNetB5	456x456

Tabla 4 - Tamaños de las imágenes en *EfficientNet*.

Además, tal y como se ha comentado con anterioridad a mayor modelo de *EfficientNet* mayor es el número de parámetros necesarios, y por lo tanto más tiempo tarda la red en entrenarse. A continuación se puede observar el número de parámetros entrenables, de parámetros no entrenables y el número total de parámetros para cada modelo:

Red neuronal	Parámetros entrenables	Parámetros no entrenables	Parámetros totales
EfficientNetB0	3.969.373	86.599	4.055.972
EfficientNetB3	10.613.545	177.671	10.791.216
EfficientNetB4	17.428.793	253.991	17.682.784
EfficientNetB5	28.174.193	349.575	28.523.768

Tabla 5 - Número de parámetros en *EfficientNet*.

Finalmente, cabe mencionar que a mayor número de parámetros mayor va a ser la precisión del modelo. En la siguiente ilustración, se puede observar el número de parámetros y la precisión que se alcanza comparando los modelos *EfficientNet* entre sí junto a otros modelos diferentes:

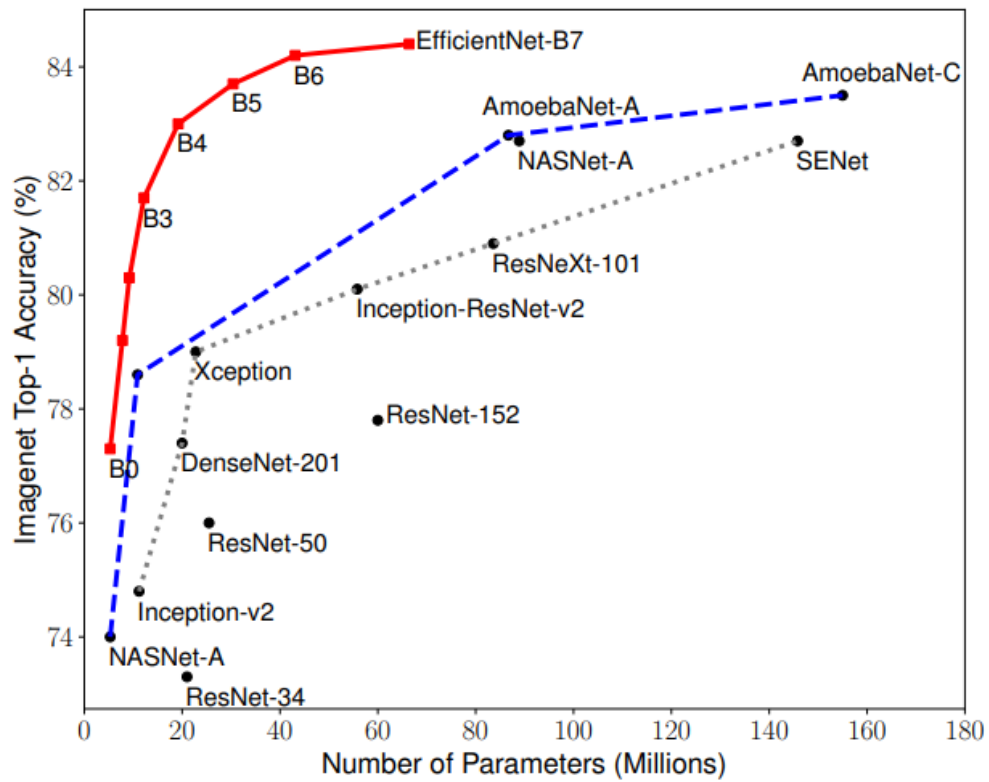


Ilustración 10 - Precisión *EfficientNet* vs otros modelos [49].

4. Desarrollo

Tal y como ya se ha mencionado anteriormente, la metodología usada para el desarrollo de este proyecto fue *Cross Industry Standard Process for Data Mining (CRISP-DM)* [18].

Por lo tanto, las fases que definimos a continuación siguen las pautas definidas por dicha metodología, es decir, el dónde se obtienen los datos, entender los mismos, prepararlos y transformarlos para un uso posterior, crear los diferentes modelos, evaluar dichos modelos, y finalmente, preparar el modelo definitivo para poder utilizarlo en un futuro.

4.1 Extracción y comprensión de los datos

4.1.1 Dónde y cómo se obtuvieron los datos

Todos los datos utilizados en este proyecto han sido obtenidos de un repositorio de *Kaggle*, para ser más exactos del repositorio “*RSNA Pneumonia Detection Challenge*” [15].



Ilustración 11 - "RSNA Pneumonia Detection Challenge" repositorio de Kaggle.

Como podemos apreciar en la anterior ilustración, éste no es un repositorio cualquiera sino que fue una competición creada por *Radiological Society of North America* [50] hace cuatro años.

El objetivo de esta competición era crear un algoritmo capaz de detectar si un paciente presentaba una neumonía a partir de imágenes médicas. Tal y como se ha comentado en la introducción de este proyecto, las neumonías presentan más del 15% de las muertes en niños menores de 5 años a nivel mundial, en el 2015 en Estados Unidos esta enfermedad causo más de 50000 muertes, dando lugar a la neumonía como una de las diez principales causas de muerte en los Estados Unidos [15].

Aunque diagnosticar con precisión la neumonía no es una tarea fácil, principalmente porque se requiere de la revisión de un especialista, el primer paso para detectar dicha enfermedad es hacer una radiografía de tórax. Las radiografías de tórax nos permiten saber si hay neumonía a partir de opacidades en los pulmones. Sin embargo, esto no siempre significa que una opacidad sea una neumonía, ya que dicha opacidad puede ser causada a partir de otras afecciones pulmonares, como por ejemplo: cáncer de pulmón, sobrecarga de líquidos, sangrado... Por lo tanto, saber si un paciente padece una neumonía con solo una radiografía es muy complicado, se necesita más información como los síntomas clínicos y un historial para afirmar el correcto diagnóstico.

Radiological Society of North America [50] lanzó este *challenge* junto a *US National Institutes of Health* [51], *The Society of Thoracic Radiology* [52] y *MD.ai* [53] para desarrollar un buen *dataset* de radiografías de tórax y así desafiar a la comunidad de *machine learning* en *Kaggle*.

Para la obtención de los datos hay que dirigirse al repositorio original en *Kaggle*, “*RSNA Pneumonia Detection Challenge*” [15]. Una vez dentro, tenemos que irnos a la pestaña “*Data*”, y para descargarlos tenemos tres opciones:

- Hacer uso de la *API* de *Kaggle*. En este caso, hay que introducir el siguiente comando en la consola: “*kaggle competitions download -c rsna-pneumonia-detection-challenge*”.

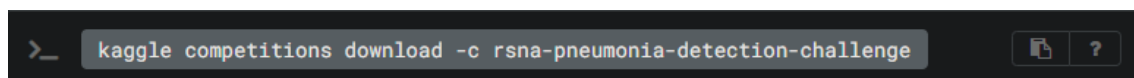



Ilustración 12 - Comando de descarga del repositorio.

- La segunda opción es descargar los archivos de uno a uno de forma manual, para ello hay que seleccionar el archivo a descargar y finalmente pulsar sobre el botón .

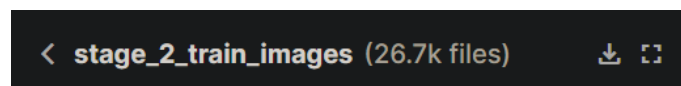


Ilustración 13 - Descarga manual de los ficheros del repositorio (uno a uno).

- La última opción es descargar los ficheros de forma manual, pero en este caso todos a la vez. Para ello hay que pulsar el botón “*Download All*”:

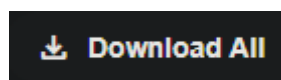


Ilustración 14 - Descarga manual de los ficheros del repositorio (todos a la vez).

Cabe mencionar que para descargar los archivos hay que cumplir dos condiciones, la primera es que debemos de tener una cuenta en *Kaggle* (no tiene ningún coste monetario asociado), y la segunda es firmar un “contrato”. Básicamente lo que se busca con dicho contrato es que se va a hacer un uso

correcto de los datos, asegurando la confidencialidad, que no se va a vender a terceros...

Una vez hecho el paso anterior podremos proceder con la descarga de los archivos y continuar con el proyecto que se desarrolla en el presente documento.

4.1.2 Estructura del repositorio

En el repositorio oficial de *Kaggle* [15] encontramos el siguiente *dashboard*:

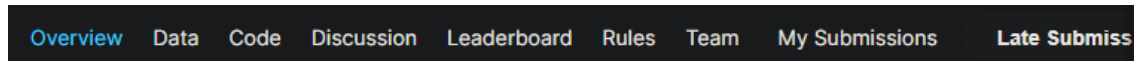


Ilustración 15 - *Dashboard* repositorio de *Kaggle*.

De la anterior imagen nos vamos a centrar en tres puntos:

- *Overview*, en él se plantea la problemática que se quiere resolver con este desafío.
- *Data*, en esta pestaña accedemos a los datos que se usan para el desarrollo de este proyecto.
- *Code*, soluciones de otros usuarios de *Kaggle* para la misma problemática o similares.

Dentro de la pestaña “*Data*” accedemos a los datos, y en ella presentamos la siguiente estructura:

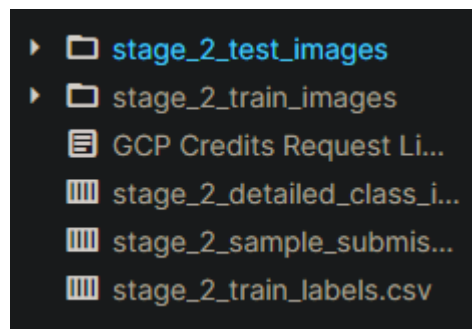


Ilustración 16 - Estructura del repositorio de *Kaggle*.

- “*stage_2_test_images*”, carpeta en la que se encuentran las imágenes para test. Cabe destacar que de estas imágenes se desconoce la variable objetivo, es decir, no sabemos para ninguna de ellas si dicho paciente padece neumonía o no, es por ello que no han sido usadas para evaluar el modelo con el conjunto de test.
- “*stage_2_train_images*”, en ella se encuentran las imágenes para *train*. En este caso sí que conocemos la variable objetivo para cada una de las imágenes, debido a esto usaremos estas imágenes para definir los conjuntos de entrenamiento, validación y test.
- “*GCP Credits Request Link – RSNA.txt*”, es un fichero de texto que incluye una dirección web para solicitar los créditos de GCP.

- “*stage_2_detailed_class_info.csv*”, archivo CSV que recoge información sobre cada uno de los pacientes de forma detallada.
- “*stage_2_sample_submission.csv*”, archivo CSV que muestra el cómo debe subirse el resultado de la predicción para la competición. En nuestro caso, este fichero es innecesario ya que la competición está finalizada.
- “*stage_2_train_labels.csv*”, archivo CSV que contiene la información sobre los pacientes para el entrenamiento del modelo. Este fichero es fundamental para obtener la variable objetivo de cada paciente, es decir, de cada imagen.

Por otro lado, se encuentra el repositorio de *GitHub* en el que se ha desarrollado el proyecto [20]. En él encontramos la siguiente estructura:

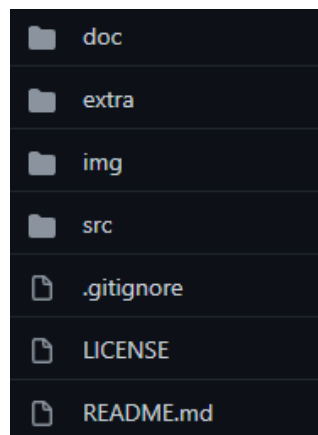


Ilustración 17 - Estructura del repositorio de *GitHub*.

- “*doc*”, carpeta en la que se encuentra la documentación.
- “*extra*”, carpeta en la que se encuentran ficheros usados en el desarrollo de la memoria, como la planificación de la misma...
- “*img*”, carpeta usada para almacenar las imágenes incluidas en esta memoria.
- “*src*”, carpeta en la que se encuentran los *notebooks* usados para el desarrollo de este proyecto. Cabe destacar que hay dos tipos de *notebooks*: “*Preprocessing.ipynb*” (donde se ha realizado todo el preprocesado de los datos de forma local), “*pneumonia_detection_BX*” (cada uno de los *notebooks* para la creación, entrenamiento y evaluación de cada modelo. Destacar que hay un *notebook* por cada una de las redes pre-entrenadas de *EfficientNet*, en este caso ejecutándose en *Google Colab* [54]).
- “*.gitignore*”, archivo en el que indicamos a *GitHub* que no suba todo al repositorio, como por ejemplo archivos temporales. Si se abre este

fichero encontramos que se ignora la subida de la carpeta “data”, en ella se encuentran todos los archivos que se han mencionado del repositorio de *Kaggle*. Se tomó esta decisión ya que la imágenes ocupaban muchos *gigabytes*, y éstas se pueden obtener fácilmente de *kaggle* como bien se mencionó en el punto anterior.

- “*LICENSE*”, archivo en el que se indica la licencia del proyecto, en nuestro caso “*GPL-3.0 license*”. Se usó esta licencia ya que permite a otros usuarios hacer uso del proyecto sin ningún problema favoreciendo a la comunidad de *machine learning*.
- “*README.md*”, archivo de *markdown* en el que definimos la portada de nuestro repositorio.

4.1.3 Estructura de los datos

En este punto se va a comentar la información relativa a los datos en sí del proyecto.

Antes de comenzar a hablar de los ficheros, hay que mencionar dos aspectos importantes sobre las imágenes. La primera es que las imágenes no están en formato “*PNG*” sino que están en “*DICOM*”.

El formato “*DICOM*” es el estándar que se usa para imágenes médicas ya que es un protocolo estándar de comunicación y almacenamiento de imágenes [55]. La clave de este formato es que aparte de la imagen en sí, se puede almacenar información relativa al paciente, como por ejemplo: el sexo, estatura, peso... Esta información adicional no se ha hecho uso de ella ya que no es necesaria para la resolución del problema.

El otro punto importante sobre las imágenes es que en el formato original cada imagen tiene un tamaño de 1024x1024 píxeles, el cual se tiene que transformar para hacer uso de las redes neuronales que definiremos más adelante.

Respecto a los ficheros, tal y como hemos mencionado antes hay dos, “*stage_2_detailed_class_info.csv*” y “*stage_2_train_labels.csv*”.

En el primero de ellos encontramos la siguiente estructura:

patientId	class
26684 unique values	No Lung Opacity / ... 39% Lung Opacity 32% Other (8851) 29%
0004cfab-14fd-4e49-80ba-63a80b6bdd6	No Lung Opacity / Not Normal
00313ee0-9eaa-42f4-b0ab-c148ed3241cd	No Lung Opacity / Not Normal

Ilustración 18 - Estructura de “*stage_2_detailed_class_info.csv*”.

- “*patientId*”: este atributo hace referencia al id del paciente, dentro de la carpeta de imágenes si buscamos dicho atributo más el formato *DICOM* “.DCM” encontramos la imagen correspondiente.
- “*class*”: en este caso el atributo hace referencia a si el paciente pertenece una de las siguientes tres clases: *No Lung Opacity / Not normal*, *Normal*, *Lung Opacity*. La dos primeras clases nos indican que no hay neumonía en la imagen, mientras que la tercera indica lo contrario.

Para resolver el problema que se plantea en este proyecto se podría haber hecho uso del anterior fichero, pero en nuestro caso no ha sido así. Se decidió hacer uso de las clases definidas en el fichero “*stage_2_train_labels.csv*” básicamente porque nos interesaba saber si hay neumonía o no, es decir, si es un 1 o un 0. Dicho fichero presenta los siguientes atributos:

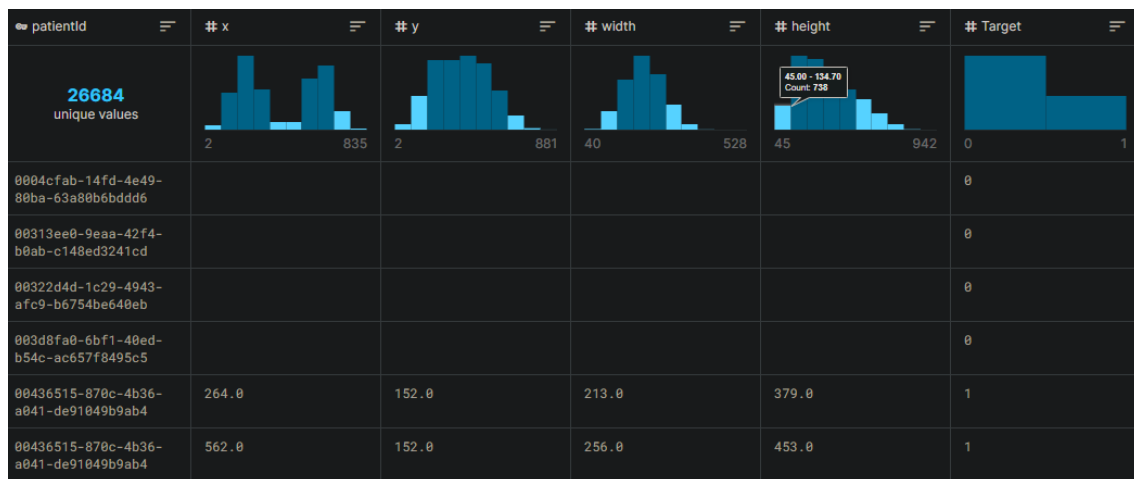


Ilustración 19 - Estructura de “*stage_2_train_labels.csv*”.

- “*patientId*”, al igual que antes indica el id del paciente, necesario para localizar la imagen.
- “*x*”, tanto este atributo como los siguientes tres son usados para identificar el área donde hay una opacidad pulmonar. Éste en concreto indica dónde empieza el área de la opacidad pulmonar respecto al eje de la x.
- “*y*”, indica dónde empieza la opacidad pulmonar respecto al eje de la y.
- “*width*”, anchura de la opacidad pulmonar.
- “*height*”, altura del área de la opacidad pulmonar.
- “*Target*”, variable objetivo. En nuestro caso, si hay un 1 hay opacidad pulmonar (representa a la clase *Lung Opacity*), si hay un 0 no hay neumonía (representa tanto a la clase *No Lung Opacity / Not Normal* como a la clase *Normal*).

4.2 Preprocesado de los datos

4.2.1 Lectura y filtrado de los datos

El primer paso en la preparación de los datos es la lectura de los mismos, para ello se ha hecho uso de la librería de *pandas* tanto para leer el archivo “*stage_2_train_labels.csv*” como para crear el *dataframe* correspondiente y manipularlo.

Una vez hecha la lectura del fichero, obtendremos 30227 registros y 6 dimensiones, las cuales, son los campos definidos en el punto anterior del presente documento (“*patientId*”, “*x*”, “*y*”, “*width*”, “*height*”, “*Target*”). El resultado de esta lectura sería el siguiente:

	<i>patientId</i>	<i>x</i>	<i>y</i>	<i>width</i>	<i>height</i>	<i>Target</i>
0	0004cfab-14fd-4e49-80ba-63a80b6bddd6	NaN	NaN	NaN	NaN	0
1	00313ee0-9eaa-42f4-b0ab-c148ed3241cd	NaN	NaN	NaN	NaN	0
2	00322d4d-1c29-4943-afc9-b6754be640eb	NaN	NaN	NaN	NaN	0
3	003d8fa0-6bf1-40ed-b54c-ac657f8495c5	NaN	NaN	NaN	NaN	0
4	00436515-870c-4b36-a041-de91049b9ab4	264.0	152.0	213.0	379.0	1

Ilustración 20 - Lectura de los datos.

A continuación comprobamos si hay elementos duplicados, en nuestro caso vemos que sí que los hay. Un aspecto a destacar de estos duplicados es que en sí no son duplicados, es decir, en los registros cuyo valor de “*Target*” sea 1, en otras palabras en los que hay neumonía, sí que hay elementos duplicados; básicamente porque se refleja tanto la opacidad pulmonar para el pulmón izquierdo como para el derecho.

En nuestro problema, queremos identificar los casos en los que hay neumonía independientemente dónde se encuentre la misma, es decir, si es en un pulmón o en el otro. Es por ello que eliminamos los duplicados, seleccionando el primer registro y descartando el segundo.

Como resultado de esta operación, pasamos a obtener el siguiente número de registros:

Número de registros

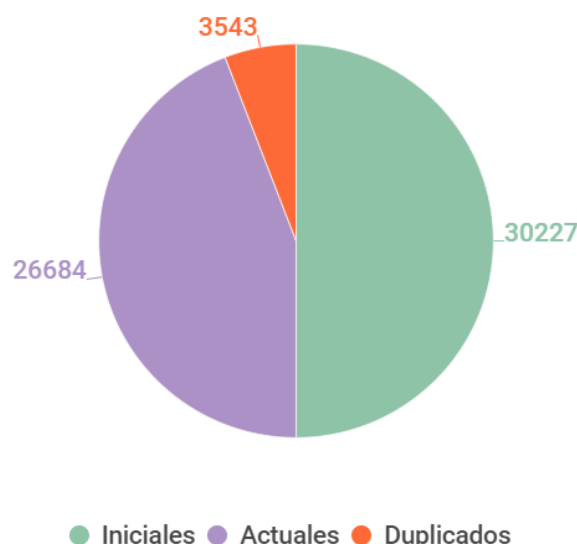


Ilustración 21 - Número de registros iniciales vs actuales vs duplicados.

El último paso es filtrar las columnas, básicamente quedarnos con los atributos que son necesarios y descartar los demás. En rojo podemos observar las variables descartadas y en verde las que finalmente se van a hacer uso en el proyecto:

Atributos finales

<i>patientId</i>	<i>Target</i>	<i>x</i>	<i>y</i>	<i>width</i>	<i>height</i>
------------------	---------------	----------	----------	--------------	---------------

Tabla 6 - Filtrado de las columnas.

4.2.2 Creación de los conjuntos de entrenamiento, validación y test

Una vez que hemos realizado el preprocesado de los datos continuamos con la creación de los conjuntos de entrenamiento, validación y test.

De los registros sin duplicados que tenemos en total, se ha decidido dividir dicho conjunto en tres grupos: uno para test, otro para validación y uno más para el entrenamiento.

En la siguiente ilustración podemos observar el número total de registros que hay para cada grupo:

Número de registros

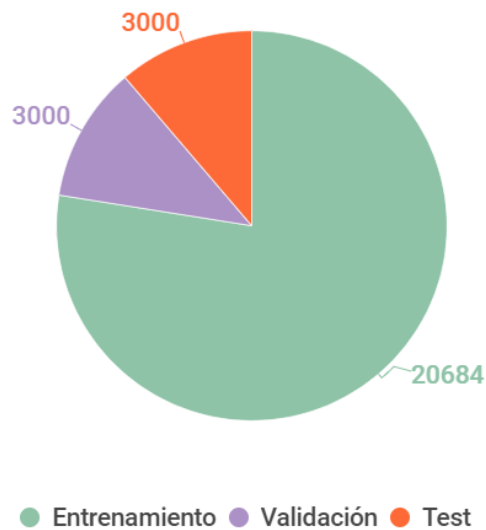


Ilustración 22 - Número de registros para el entrenamiento vs validación vs test.

De forma aproximada para el entrenamiento se hace uso del 80% de los datos, para validación y test un 10% respectivamente.

En cuanto al detalle de implementación sobre la estrategia de partición escogida, se tomó esta decisión porque en la carpeta “*stage_2_test_images*” del repositorio de *Kaggle* [15], se pone a disposición de 3000 imágenes para test. Así que siguiendo la misma pauta se han definido dichos conjuntos.

4.2.3 Transformación de las imágenes

En este apartado se realiza el preprocesado de las imágenes en sí, es decir, realizamos la transformación del formato “*DICOM*” al “*PNG*” y se procede con el redimensionado de las mismas.

Respecto al cambio de formato, se determinó que lo mejor era convertirlas a formato “*PNG*” por dos motivos, el primero con este formato es más fácil trabajar, y en segundo lugar viene determinado por el *data augmentation* que realizaremos en el punto 4.2.4, ya que es más sencillo realizar el guardado de una imagen “*PNG*” que una con el formato “*DICOM*”.

Por otro lado, a la par que se transforma el formato de las imágenes realizamos el redimensionado de las mismas, esto se debe a que el tiempo de entrenamiento de la red neuronal va a ser menor si el *input* que recibe ya está en el tamaño adecuado.

Para saber qué tamaño se corresponde para cada uno de los modelos que se detallarán más adelante, podemos ver dicha información en la tabla 4 del presente documento.

Finalmente, cabe destacar que en todas las tareas de preprocesado se comprueba que todo se realiza de forma correcta. De tal forma que cuando se vayan a usar los datos para el entrenamiento de los modelos esté todo en orden.

4.2.4 Balanceo de los datos

Continuando con el preprocesado de los datos falta comprobar el balanceo de los mismos, es decir, si para el conjunto de entrenamiento, el cual vamos a usar para entrenar la red, tiene el mismo número de casos de pacientes con neumonía que sin neumonía.

Ejecutando los comandos correspondientes vemos que no hay el mismo número de casos, para ser más exactos, nos encontramos en la siguiente situación:

Proporción de imágenes

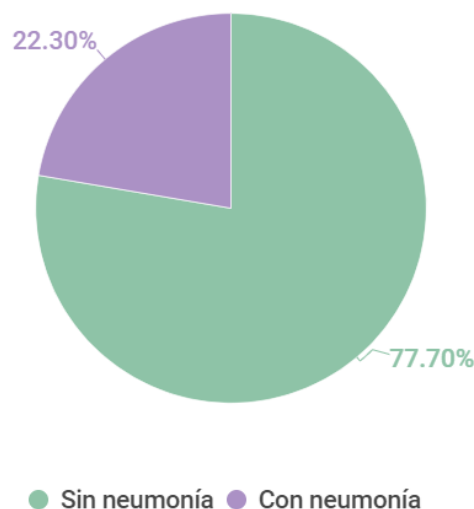


Ilustración 23 - Número de imágenes sin neumonía vs con neumonía.

Hay casi un factor de 3.5 imágenes más de pacientes sin neumonía que con neumonía, por lo que hay un claro desbalanceo de clases.

Solventar el desbalanceo de clases es fundamental para el entrenamiento de la red, ya que de lo contrario podríamos tener una red con pesos sesgados, es decir, que la red se especialice más en los casos sin neumonía que con neumonía.

La solución a este problema viene dada a partir del aumento de datos de la clase minoritaria, en nuestro caso de las imágenes con neumonía.

Para ello, se ha realizado el *data augmentation* con el módulo *ImageDataGenerator* de *Keras*, gracias a él podemos crear imágenes nuevas a

partir de las originales, estableciendo parámetros como: la rotación de la imagen, giro horizontal, zoom, desplazar la misma a lo ancho o vertical...

Este aumento de datos se realiza en dos fases, en la primera se van a crear tantas imágenes aumentadas como sea la diferencia con la proporción, es decir, si la proporción es 3.5 se van a crear “ $3 - 1 = 2$ ” imágenes aumentadas por cada imagen con neumonía. De esta forma nos garantizamos que todos los casos con neumonía al menos van a tener dos imágenes aumentadas.

La segunda fase consiste en crear una imagen aumentada a partir de las imágenes originales con neumonía de forma aleatoria, hasta llegar al punto en el que haya un balanceo de clases.

Respecto a los detalles de implementación sobre *ImageDataGenerator*, se han usado los siguientes parámetros con sus correspondientes valores:

Parámetro	Valor
<i>samplewise_center</i>	False
<i>samplewise_std_normalization</i>	False
<i>horizontal_flip</i>	True
<i>vertical_flip</i>	False
<i>height_shift_range</i>	0.05
<i>width_shift_range</i>	0.02
<i>rotation_range</i>	3
<i>shear_range</i>	0.01
<i>fill_mode</i>	nearest
<i>zoom_range</i>	0.05

Tabla 7 - Detalles de implementación del *ImageDataGenerator*.

En la siguiente ilustración podemos observar el resultado del balanceo de clases:

Proporción de imágenes



Ilustración 24 - Número de imágenes balanceadas sin neumonía vs con neumonía.

Una vez alcanzado el balanceo aumenta el *dataset* de entrenamiento, de esta forma se consigue dotar de mayor calidad el *dataset* original.

A continuación podemos ver la diferencia en el tamaño anterior y actual:

Número de registros

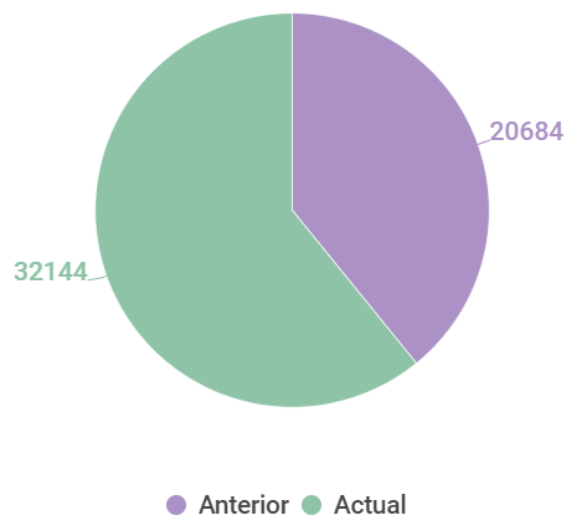


Ilustración 25 - Tamaño del *dataset* balanceado antes vs ahora.

En las siguientes ilustraciones se puede apreciar a modo de ejemplo las imágenes que genera el *data augmentation*:

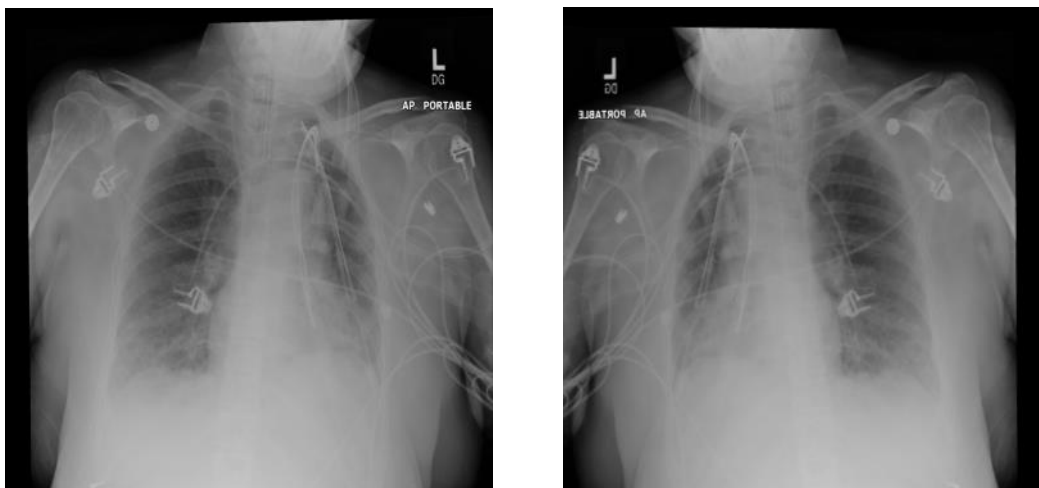


Ilustración 26 - Imagen original vs imagen aumentada.

Finalmente, cabe mencionar el nombre que se le da a las imágenes aumentadas. Para las imágenes que se crean en la fase uno su nombre viene proporcionado por el atributo *"patientId"*, más *"#a"*, más la iteración en la que se crea (0,1). Mientras que para la fase dos, el nombre es el atributo *"patientId"*, más *"#b"*, más la iteración en la que se crea (en nuestro caso 0).

En la siguiente tabla podemos apreciar tanto el nombre original como los nombres de las imágenes aumentadas:

Nombre original	<i>0a8d486f-1aa6-4fcf-b7be-4bf04fc8628b.png</i>
Fase 1 – iteración 1	<i>0a8d486f-1aa6-4fcf-b7be-4bf04fc8628b#a0.png</i>
Fase 1 – iteración 2	<i>0a8d486f-1aa6-4fcf-b7be-4bf04fc8628b#a1.png</i>
Fase 2 – iteración 1	<i>0a8d486f-1aa6-4fcf-b7be-4bf04fc8628b#b0.png</i>

Tabla 8 - Comparativa del nombres original vs aumentada.

4.2.5 Generación de los ficheros de salida

Esta es la última fase de preprocesado, en ella lo que se hace es generar todos los archivos "CSV" necesarios para continuar con el siguiente apartado de creación de los modelos, es decir, se generan los archivos de entrenamiento, validación y test.

Estos ficheros resultantes contienen los siguientes datos:

- *"train_labels.csv"*, en él encontramos al atributo *"PatientId"* y *"Target"* sobre las imágenes de entrenamiento. Este archivo contiene 32144 registros.

- “*validation_labels.csv*”, al igual que el anterior contiene “*PatientId*” y “*Target*” para el conjunto de validación. En este caso hay 3000 registros.
- “*test_labels.csv*”, sigue teniendo los mismos atributos y en este caso hay también 3000 registros.

4.3 Creación de los modelos

4.3.1 Data Generator

Una de las problemáticas de trabajar con imágenes es que a la hora de entrenar los modelos hay dificultades en cargar en memoria todas ellas. El *dataset* de entrenamiento está formado por 32144 imágenes, que dependiendo del modelo que usemos, los cuales serán explicados más adelante, dichas imágenes tendrán diferentes tamaños, haciendo que a mayor tamaño mayor consumo de memoria.

Para solventar esta situación se crea un *data generator*, el cual nos va a permitir ir cargando en memoria pequeños grupos de imágenes, o como se denomina en este campo *batch*, a medida que se vayan necesitando.

La creación de este *data generator* se consigue gracias a la clase “*Sequence*” de la librería *Keras*.

Dentro de esta clase se han definido los siguientes métodos:

- “*__init__(...)*”, este método es el constructor de la clase, el cual nos va a permitir crear cada uno de nuestros objetos de tipo “*Sequence*”.
- “*__train__(...)*”, se encarga de encontrar la imagen y cargarla en memoria, además de encontrar para cada imagen el valor de la variable objetivo. En otras palabras, nos devuelve la imagen y si ésta indica que el paciente padece neumonía o no.
- “*__test__(...)*”, en este caso se carga en memoria y se devuelve la imagen correspondiente, es decir, no hay un retorno de la variable objetivo.
- “*__getitem__(...)*”, método perteneciente a la clase “*Sequence*”, el cual a partir de un índice es capaz de generar el *batch* necesario para el entrenamiento, validación y test.
- “*__on_epoch_end()*”, este método se encarga de mezclar los nombres de los archivos para dotar de mayor aleatoriedad en el entrenamiento.
- “*__len__()*”, devuelve el número de *batches* que se tienen que generar.

4.3.2 Carga de las imágenes preprocesadas

Uno de los principales inconvenientes que se ha tenido a la hora de desarrollar este proyecto ha sido el tiempo de ejecución a la hora de entrenar los modelos.

Al principio se ideó como solución el hacer uso exclusivamente del *data generator* para reducir el consumo de memoria, para ello dentro del propio generador se tenía que buscar la imagen, abrirla y redimensionar al tamaño correspondiente para cada época, esto lo que significa es mucho tiempo de ejecución, para ser más exactos, con la red más sencilla tardaba el modelo dos horas por época en entrenarse.

Para solventar esta situación ya que era inviable que tardase tanto, se decidió sacrificar el consumo de memoria, es decir, en vez de cargar las imágenes poco a poco con el *data generator* se cargan todas a la vez, y se hace uso del generador para introducir a la red cada *batch* como *input*.

Esta carga de las imágenes se realiza a partir de un diccionario. Todo diccionario está formado por una pareja clave-valor, nuestra clave es el atributo "*patientId*" (el nombre de las imágenes sin la extensión "*PNG*") y el valor es la imagen preprocesada con el tamaño correspondiente.

Aunque puede sonar contradictorio el definir el *data generator* para reducir el consumo de la memoria, y luego hacer uso de un diccionario que contiene todas las imágenes, las cuales son cargadas todas en memoria, la realidad es todo lo contrario. En los equipos que tienen una capacidad de cómputo elevada (como en el caso de *Google Colab* [54]) haciendo uso tanto del *data generator* como del diccionario se consigue reducir de forma radical el tiempo de entrenamiento, ya que donde antes tardaba dos horas por época con el modelo más sencillo ahora se tarda un minuto y medio; por otro lado, si un usuario no puede acceder a una capacidad de cómputo elevada, tiene la opción de no usar el diccionario (evitar un mayor consumo de memoria) y hacer uso solamente del *data generator*, consiguiendo poder entrenar la red, pero a cambio de un elevado tiempo de entrenamiento.

4.3.3 Transfer learning

El siguiente paso en el desarrollo del proyecto ha sido crear las diferentes redes neuronales convolucionales para resolver el problema que se ha definido en puntos anteriores.

Recapitulando el apartado 3.3 del presente documento, donde se explicaban las redes neuronales convolucionales, dichas redes se caracterizan por recibir como *input* un vector (la imagen) y proporcionar como *output* la clase a la que pertenece dicha imagen.

Crear una red neuronal convolucional desde cero es un trabajo muy costoso, tanto la creación como el entrenamiento de la misma, y es aquí donde aparece un concepto fundamental, el *transfer learning*.

El *transfer learning* [56] [16] nos va a permitir solucionar la problemática comentada en el párrafo anterior. El objetivo de este método es hacer uso de

redes neuronales pre-entrenadas con otros conjuntos de datos, para así reducir el tiempo de entrenamiento de la red y obtener una buena precisión en los conjuntos de validación y de test.

La mecánica del *transfer learning* consiste en definir un modelo base pre-entrenado, y posterior a éste crear un nuevo modelo basado en el modelo base. De forma resumida lo que se busca es lo siguiente:

1. Inicializar un modelo base con pesos ya pre-entrenados.
2. Congelar los pesos de las capas del modelo base.
3. Crear un nuevo modelo a partir del modelo base.
4. Entrenar el nuevo modelo con el *dataset*.

En otras palabras, lo que se busca es entrenar el nuevo modelo, es decir el clasificador, pero haciendo uso del modelo base.

En nuestro caso se ha hecho uso de *EfficientNet* como modelo base. *EfficientNet* [48] no es más que un conjunto de redes neuronales convolucionales pre-entrenadas proporcionadas por *Google*. Para conocer más detalles sobre estas redes se puede acceder al punto 3.4, en el cual se explica la arquitectura de las mismas, qué tamaño de imagen necesitan de entrada, cuántos parámetros se entrenan, qué precisión se alcanza con cada modelo...

Todos los modelos que se han creado para resolver la problemática de este proyecto presentan la siguiente estructura:

1. Se define el *input* que va a tener la imagen, en nuestro caso depende de qué tipo de *EfficientNet* se use. Si usamos como ejemplo *EfficientNetB0* el tamaño de la imagen va a ser 224x224, y solo un canal ya que las imágenes están en escala de grises.
2. Después, para dotar de mayor variabilidad al modelo se ha creado una capa de *data augmentation*.
3. Un aspecto a tener en cuenta es que las redes *EfficientNet* hacen uso de imágenes en color, es decir, se necesitan tres canales. En nuestro caso las imágenes están en escala de grises por lo que tenemos solamente un canal. Para solventar esta situación, se ha creado una capa "*Concatenate*", concatenado la misma imagen tres veces conseguimos los tres canales.
4. Una vez realizado esto ya podemos aplicar *transfer learning*, para ello se inicializa el modelo de *EfficientNet* que queremos usar, en este proyecto se han creado modelos para *EfficientNetB0*, *EfficientNetB3*, *EfficientNetB4* y *EfficientNetB5*. Un aspecto importante en la inicialización es que las capas finales no estén incluidas en el modelo base.

5. Creado el modelo base pasamos a congelar todas las capas del mismo.
6. Posteriormente, creamos de nuevo las capas finales. Básicamente se crea una capa “*GlobalAveragePooling2D*”, una capa “*BatchNormalization*”, una capa “*Dropout*” con un *rate* de 0.2 y una capa de salida “*Dense*”.
7. En nuestro problema, buscamos saber si un paciente padece una neumonía o no, es decir, la red devuelve una probabilidad entre 0 y 1. Es por ello, que la capa de salida “*Dense*” presenta una activación de tipo “*sigmoid*” y solamente una neurona, si el valor que devuelve la red es inferior a 0.5 se interpreta como un paciente sin neumonía, en caso contrario de ser superior a 0.5 hay síntomas de esta enfermedad en la radiografía.
8. Para terminar, falta compilar el modelo. Como optimizador se ha usado “*Adam*”, como función de pérdida “*binary_crossentropy*” (usada para problemas de clasificación binaria) y como métrica “*accuracy*”.

En la sección 5 del presente documento, se explicará qué experimentos se han realizado, cuáles han sido los resultados y qué modelo se ha comportado mejor, pero para terminar con esta sección 3 nos falta hablar de *fine-tuning*.

4.3.4 Fine-tuning

Aplicado el *transfer learning* al modelo, tenemos que mejorar la precisión. Para ello, se hace uso de otra técnica conocida como *fine-tuning*.

Fine-tuning [16] tiene como objetivo mejorar la precisión que se obtiene al aplicar *transfer learning*, esto se consigue gracias a descongelar parte del modelo base o todo el modelo, entrenar de nuevo la red y usar un *learning rate* bajo.

Cuando se habla de “descongelar”, a lo que se refiere es a que los pesos de las capas del modelo base ahora sí que se van a modificar cuando se entrene la red, es decir, con *transfer learning* se entrena solo el clasificador final (pesos del clasificador final) y con *fine-tuning* los pesos de toda la red.

Para el desarrollo de este proyecto, a la hora de aplicar *fine-tuning* se ha decidido descongelar todo el modelo base, con la excepción de las capas “*BatchNormalization*”, ya que según establece *Keras* en su web oficial, descongelar estas capas reducen la precisión de la primera época en el *training* del modelo [57].

4.3.5 Detalles de implementación

En este apartado se van a detallar los parámetros con los que se han creado los diferentes modelos.

En cuanto a los modelos en sí, cabe destacar los parámetros con los que se ha definido la capa de *data augmentation* tanto para *EfficientNetB0*, como para el resto de modelos (*EfficientNetB3*, *EfficientNetB4*, *EfficientNetB5*) son los siguientes (el guion indica el mismo dato que en el modelo anterior):

Layer	EfficientNetB0	EfficientNetB3	EfficientNetB4	EfficientNetB5
<i>RandomRotation</i>	0.05	-	-	-
<i>RandomTranslation (height)</i>	0.05	-	-	-
<i>RandomTranslation (width)</i>	0.02	-	-	-
<i>RandomFlip</i>	<i>horizontal</i>	-	-	-
<i>RandomContrast</i>	0.05	-	-	-

Tabla 9 - Parámetros *data augmentation* de los modelos.

Respecto a los hiperparámetros de cada modelo entrenado para el *transfer learning*, es decir, los modelos en los que solo se entrena el clasificador final, se presenta la siguiente información:

Hiperparámetros	EfficientNetB0	EfficientNetB3	EfficientNetB4	EfficientNetB5
<i>IMG_SIZE</i>	224	300	380	456
<i>BATCH_SIZE</i>	32	-	-	-
<i>EPOCHS</i>	20	-	-	-
<i>LEARNING_RATE</i>	1e-4	-	-	-
<i>OPTIMIZER</i>	<i>adam</i>	-	-	-
<i>LOSS FUNCTION</i>	<i>binary_crossentropy</i>	-	-	-

Tabla 10 - Hiperparámetros de los modelos *transfer learning*.

Una vez entrenado el clasificador final de los modelos de *transfer learning*, se vuelve a entrenar toda la red (*fine-tuning*), siendo éstos los modelos finales obtenidos en el proyecto. Los hiperparámetros usados para cada modelo se han establecido con el fin de mejorar los resultados, éstos son los siguientes:

Hiperparámetros	<i>EfficientNetB0</i>	<i>EfficientNetB3</i>	<i>EfficientNetB4</i>	<i>EfficientNetB5</i>
<i>IMG_SIZE</i>	224	300	380	456
<i>BATCH_SIZE</i>	32	32	16	8
<i>EPOCHS_FT</i>	5	5	7	7
<i>LEARNING_RATE</i>	1e-5	-	-	-
<i>OPTIMIZER</i>	<i>adam</i>	-	-	-
<i>LOSS FUNCTION</i>	<i>binary_crossentropy</i>	-	-	-

Tabla 11 - Hiperparámetros de los modelos *fine-tuning* (modelos finales).

4.4 Evaluación de los modelos

Para concluir con el presente capítulo, se van a detallar las diferentes funciones de evaluación usadas para valorar la calidad de cada modelo.

En este proyecto se han tenido en cuenta: la matriz de confusión, la exactitud, el *recall*, el *f1-score*, curva *ROC* y el *AUC*.

4.4.1 Matriz de confusión

La matriz de confusión es una técnica que nos permite saber cómo de bueno es un modelo en problemas de aprendizaje supervisado, más concreto en tareas de clasificación [58].

Las columnas de una matriz de confusión representan las clases predichas por el modelo, mientras que las filas son las clases reales. Esta técnica nos permite saber qué tipos de aciertos hay en el modelo, y qué errores se están cometiendo.

En la siguiente tabla podemos ver cómo es una matriz de confusión:

		VALORES PREDICHOS	
VALORES REALES			
		Verdaderos Negativos (VN)	Falsos Positivos (FP)
		Falsos Negativos (FN)	Verdaderos Positivos (VP)

Tabla 12 - Matriz de confusión

Los cuatro casos que se encuentran dentro de una matriz de confusión son:

- Verdadero negativo, el valor real es negativo y el valor predicho es negativo. En nuestro caso, si el valor real y predicho coinciden es que no hay neumonía.
- Falso positivo, el valor real es negativo y el predicho positivo. En la realidad no hay neumonía pero el modelo lo predice como sí.
- Falso negativo, el valor real es positivo y el predicho es negativo. Sí que hay neumonía pero el modelo dice lo contrario.
- Verdadero positivo, el valor real y predicho son positivos. Hay neumonía y el modelo es capaz de detectarlo correctamente.

4.4.2 Exactitud

Primera medida que podemos calcular a partir de la matriz de confusión. Ésta nos indica el sesgo del modelo, es decir, cómo de bien se acerca el resultado obtenido por el modelo al valor real [58].

La exactitud o *accuracy*, se calcula de la siguiente forma:

$$\text{Exactitud} = \frac{VP + VN}{VP + FP + FN + VN}$$

Cuanto más elevado sea el anterior valor más exacto es el modelo en clasificar los datos correctamente.

4.4.3 Precisión

La segunda medida calculada a partir de la matriz de confusión es la precisión o *precision*. Nos permite saber cómo es la dispersión de los valores obtenidos por el modelo, a mayor dispersión menor precisión [58].

La precisión calcula la proporción de verdaderos positivos entre todos los resultados positivos obtenidos por el modelo.

Su fórmula es la siguiente:

$$\text{Precisión} = \frac{VP}{VP+FP}$$

4.4.4 Recall o sensibilidad

La tercera medida que se calcula con la matriz de confusión es el *recall*. Éste nos permite saber la proporción de casos positivos que han sido clasificados correctamente por el modelo [58].

En nuestro problema, cuanto mayor sea este valor mejor, ya que en el ámbito de la salud se busca reducir el número de falsos negativos, en otras palabras, el objetivo es detectar aquellos pacientes que padecen de neumonía (sin olvidar que es importante también identificar correctamente los pacientes que no padecen dicha enfermedad).

La sensibilidad o *recall*, se calcula de la siguiente forma:

$$\text{Recall} = \frac{VP}{VP+FN}$$

4.4.5 Especificidad

La especificidad se calcula también a partir de la matriz de confusión, en este caso, esta métrica nos indica los casos negativos que el modelo es capaz de clasificar correctamente.

En la problemática que se está resolviendo en este proyecto, la especificidad nos indica cuántos de los pacientes sin neumonía de entre todos los pacientes sanos se han clasificado bien.

Se calcula de la siguiente manera:

$$\text{Especificidad} = \frac{VN}{VN+FP}$$

4.4.6 F1-score

La última medida que calculamos directamente con la matriz de confusión es el *f1-score*. Esta medida junta tanto la precisión como la sensibilidad en una, la

cual es muy útil cuando la proporción de casos positivos y negativos no es la misma [58].

El conjunto de test no presenta el mismo número de casos positivos que negativos, es por ello que para la elección del mejor modelo se ha tenido más en cuenta esta métrica.

Al igual que sucedía con las métricas anteriores, a mayor valor mejor rendimiento del modelo.

El *f1-score* se calcula así:

$$F1\text{-score} = \frac{2 * \text{Precisión} * \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}}$$

4.4.7 Curva ROC (Receiver Operating Characteristic)

La curva ROC es una representación gráfica de la sensibilidad de un modelo frente a la especificidad (medidas explicadas en los puntos 4.4.4 y 4.4.5) según se varía el umbral de discriminación. Otra forma de entender la curva ROC es la proporción de falsos positivos frente a la proporción de verdaderos positivos [59].

Un ejemplo de curva ROC es el siguiente:

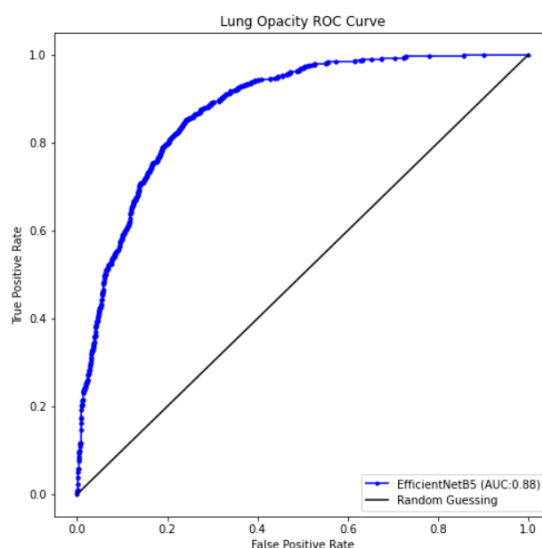


Ilustración 27 - Curva ROC.

Cuanto más cercana sea la curva al 1 mejor es el rendimiento del modelo.

4.4.8 AUC (Area Under Curve)

El AUC como su propio nombre indica es el área bajo la curva, es decir, el área que se genera al calcular la curva ROC. Cuanto más próximo esté el área al valor 1 mejor se comporta el modelo, en caso de ser 0.5 sucede lo contrario.

5. Experimentos y resultados

En este apartado se van a detallar las diferentes métricas alcanzadas para cada uno de los modelos. Todos los hiperparámetros usados para el entrenamiento de cada una de las redes han quedado detallados en el punto 4 del presente documento.

5.1 EfficientNetB0 (base line)

A continuación se detalla la exactitud y la pérdida tanto para la etapa previa (*transfer learning*) como para la versión final del modelo (*fine-tuning*):

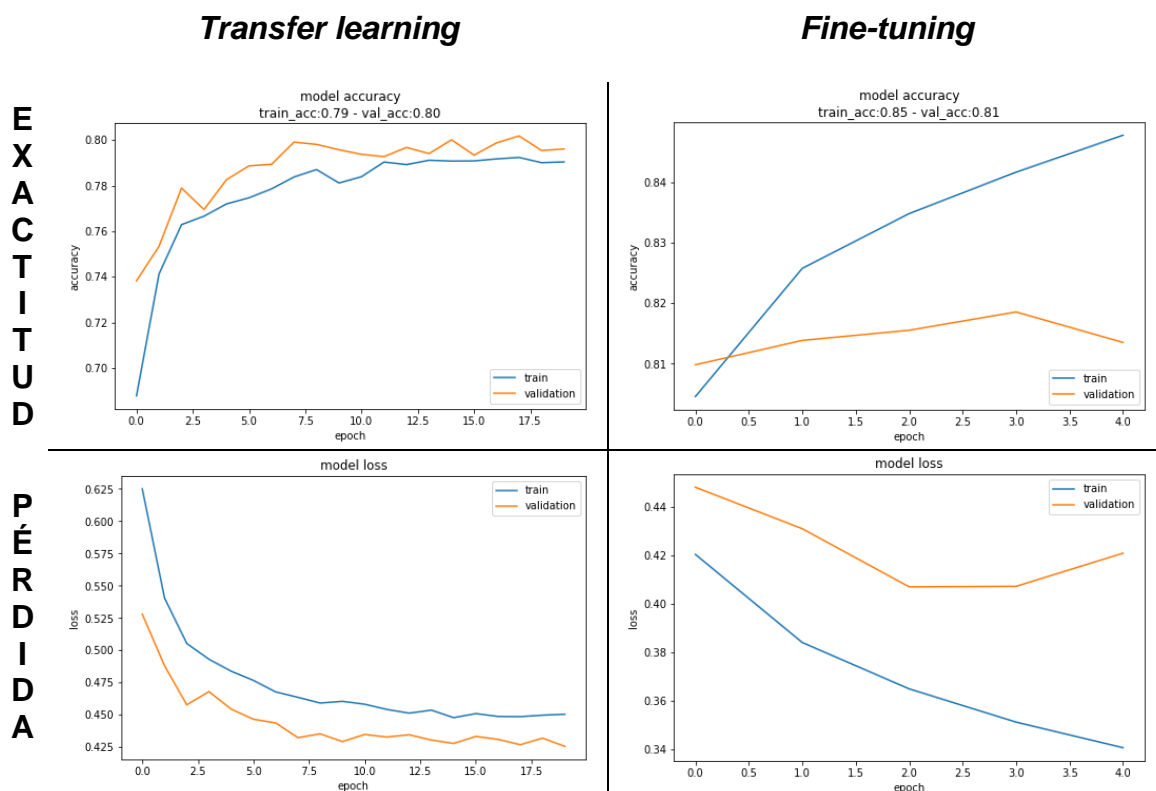


Tabla 13 - Evolución de la exactitud y pérdida de *EfficientNetB0*.

Las métricas obtenidas para el modelo final (*fine-tuning*) son:

Exactitud	Precisión	Recall	Especifici.	F1-score	AUC
0.81	0.67	0.36	0.95	0.47	0.84

Tabla 14 - Resultados de *EfficientNetB0*.

Como podemos apreciar en las anteriores ilustraciones, cuando entrenamos solamente el clasificador final se consigue un aprendizaje estable en términos de *accuracy* y *loss*. Sin embargo, cuando se realiza el entrenamiento de toda la red, es decir, tanto del modelo base como de los *layers* finales se consigue una mayor exactitud, a cambio de un ligero sobreajuste en el entrenamiento (esta evidencia se puede apreciar en el *accuracy* y en la pérdida que se produce en el *fine-tuning*).

Respecto al sobreajuste que se produce cuando se entrena toda la red cabe mencionar que es normal, al permitir cambiar los pesos del modelo base éste va aprendiendo más el conjunto con el que se entrena y ajusta los pesos a la problemática que se busca resolver.

Las métricas obtenidas para el modelo final (*fine-tuning*) no son las mejores tal y como podremos ver más adelante. Se consigue una buena exactitud, pero el *recall* o sensibilidad no es elevado. En un problema del ámbito sanitario se busca obtener el mayor *recall* posible, básicamente para clasificar de forma correcta los casos positivos y reducir los falsos negativos. Un aspecto a tener en cuenta de este modelo es que es bueno clasificando los casos negativos, esto se puede ver en la elevada especificidad. Aún así estos parámetros nos dan una idea general, pero los importantes son el *f1-score* y el AUC, y tal y como veremos en los próximos modelos no son los mejores encontrados.

5.2 EfficientNetB3

Se detalla la exactitud y la pérdida para este modelo:

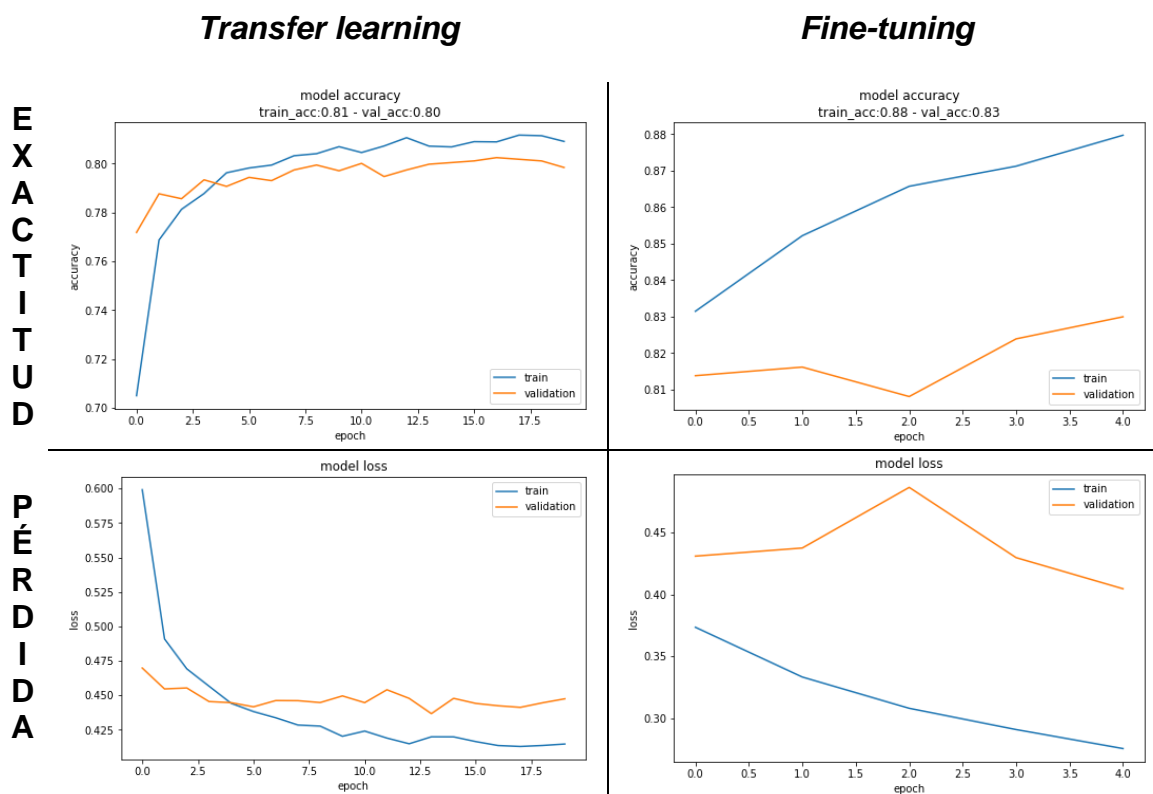


Tabla 15 - Evolución de la exactitud y pérdida de *EfficientNetB3*.

Las métricas obtenidas para el modelo final son:

Exactitud	Precisión	Recall	Especifici.	F1-score	AUC
0.82	0.67	0.46	0.93	0.55	0.86

Tabla 16 - Resultados de *EfficientNetB3*.

Respecto al entrenamiento exclusivo del clasificador final, vemos que con un número pequeño de épocas el modelo consigue alcanzar la exactitud máxima. Además, se produce un sobreajuste muy pequeño en el entrenamiento, ya que la exactitud y pérdida alcanzada para el conjunto de entrenamiento es ligeramente mejor que para el de validación.

En este caso al hacer *fine-tuning* el modelo sí que presenta una gran mejora, se produce un sobreajuste como es normal pero aumenta la exactitud de un 80% a un 83% en el conjunto de validación. Por otro lado aunque el *recall* sigue siendo bajo, se ha conseguido aumentar respecto al modelo anterior, es más todas las métricas con excepción de la especificidad son mejores al anterior modelo. En otras palabras, al aumentar la complejidad de la red y el tamaño de las imágenes se consigue mejorar el resultado.

5.3 EfficientNetB4

Se detalla la exactitud y la pérdida para este modelo:

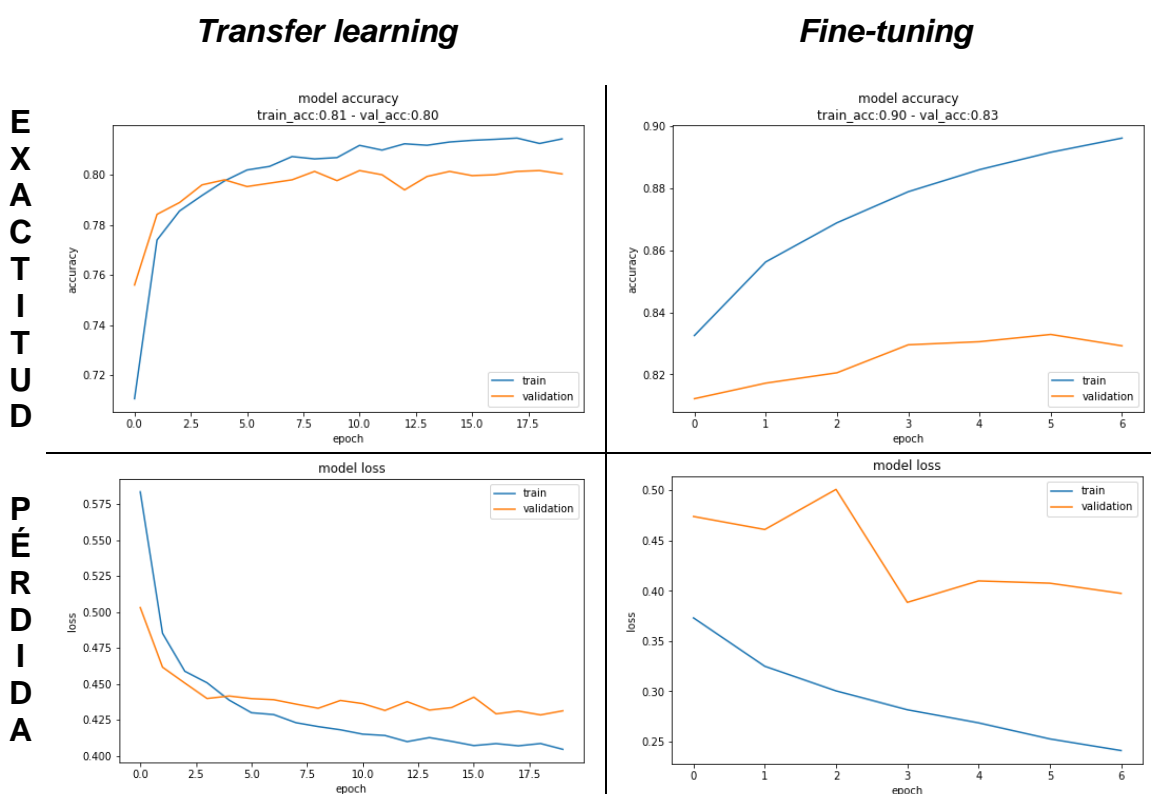


Tabla 17 - Evolución de la exactitud y pérdida de *EfficientNetB4*.

Las métricas obtenidas para el modelo final son:

Exactitud	Precisión	Recall	Especifici.	F1-score	AUC
0.83	0.64	0.57	0.90	0.60	0.87

Tabla 18 - Resultados de *EfficientNetB4*.

En cuanto al entrenamiento del clasificador final vemos que se realiza de forma estable, es decir, a medida que aumenta la época va mejorando la exactitud y la pérdida. Al igual que sucedía en el modelo anterior, se consigue una mayor exactitud al hacer *fine-tuning*, alcanzando del 80% al 83% entre entrenar solo el clasificador final o toda la red.

Respecto las métricas obtenidas para el modelo final cabe destacar que son mejores que los dos modelos anteriores, se consigue una mayor exactitud, un mejor *recall*, un mayor *f1-score* y un AUC más elevado, es decir, sacrificando un poco la precisión y la especificidad este modelo es capaz de identificar mejor los casos positivos (aspecto importante al reducir los falsos negativos, clave en un problema del ámbito sanitario).

Se detalla la exactitud y la pérdida para este modelo:



Las métricas obtenidas para el modelo final son:

Tabla 20 - Resultados de *EfficientNetB5*.

En este caso vemos que entrenando solamente el clasificador final desde un principio se consigue la máxima exactitud. Además se produce sobreajuste, entrenando solamente al clasificador final se obtiene una mayor pérdida que en el modelo anterior.

Al aplicar *fine-tuning* y conseguir el modelo final sí que se hay una mejora en los resultados. Aumenta la exactitud conseguida y la pérdida es ligeramente mejor que el modelo de *EfficientNetB4*.

En cuanto a las métricas obtenidas para el modelo final se consigue la misma exactitud que en el modelo anterior, pero aumenta el *recall* y mejora tanto el *f1-score* como el AUC, es decir, se clasifica mejor los casos positivos (los casos

en los que hay neumonía). Para ello se sacrifica un poco la precisión y la especificidad comparada a modelos anteriores.

5.5 Selección del modelo final

Modelo	Exactitud	Precisión	Recall	Especifici.	F1-score	AUC
<i>EfficientNetB0</i>	0.81	0.67	0.36	0.95	0.47	0.84
<i>EfficientNetB3</i>	0.82	0.67	0.46	0.93	0.55	0.86
<i>EfficientNetB4</i>	0.83	0.64	0.57	0.90	0.60	0.87
<i>EfficientNetB5</i>	0.83	0.63	0.60	0.89	0.61	0.88

Tabla 21 - Comparativa de modelos.

Tal y como podemos apreciar en la anterior tabla, al aumentar la complejidad de la red y el tamaño de las imágenes se consiguen mejores resultados.

Respecto a la exactitud obtenida en cada modelo, vemos que va mejorando poco a poco. Sin embargo, estamos hablando de una exactitud del 83% en el mejor de los casos, es decir, se sigue cometiendo pequeños errores.

Cuando estamos en un problema del ámbito sanitario prima más la sensibilidad del modelo que la precisión del mismo, es decir, se busca clasificar el mayor número de casos positivos y primar reducir el número de falsos negativos antes que los falsos positivos. Es por ello, que a medida que aumenta la complejidad de la red se va reduciendo la precisión y aumentando el *recall* consiguiendo mejores resultados.

A parte de que el modelo final tiene que ser bueno identificando los casos positivos no hay que olvidar los casos negativos, es decir, aquellos en los que no hay neumonía. Por lo tanto, un buen modelo tiene que seguir identificando de forma correcta los casos en los que no hay neumonía. Para ver en este campo qué modelo se comporta mejor nos guiamos del atributo especificidad, y vemos que a medida que aumenta la complejidad de la red se va reduciendo ligeramente este atributo, haciendo a priori un modelo peor identificando los casos negativos. Esta caída ligera compensa frente a la gran mejora que se produce en la sensibilidad, es por ello que no es del todo un problema.

Por último, guiarnos sobre los atributos anteriores nos permite hacernos una idea de qué modelo puede funcionar mejor. Sin embargo, la realidad es que un buen modelo viene denotado tanto por el *f1-score* como el AUC, ya que al tener un conjunto de test desbalanceado las métricas mencionadas en los párrafos anteriores pueden estar ligeramente sesgadas.

Por lo tanto, el mejor modelo que se ha conseguido obtener para resolver la problemática que se plantea en este proyecto es el modelo *EfficienNetB5*.

6. Conclusiones

En el desarrollo de este trabajo, se han realizado un conjunto de tareas de análisis y predicciones con el fin de mejorar la prevención y la identificación de patologías relacionadas con la enfermedad de la neumonía. En líneas generales se ha obtenido un buen resultado, debido al cumplimiento de todos los objetivos planteados al inicio del proyecto.

En lo que respecta a la neumonía en sí, al comienzo del presente documento se hizo un análisis sobre qué es la misma, qué factores influyen, qué fases de desarrollo presenta, cómo se detecta... Es conveniente resaltar la importancia que tiene dicha enfermedad, realizando tareas de investigación se descubrió que hay una elevada mortalidad a nivel mundial causada por la neumonía, de lo que podemos concluir la importancia de la misma.

A día de hoy, la forma más barata y sencilla de descubrir esta enfermedad es a partir de radiografías de tórax, pero una lección aprendida en este apartado es que no siempre se usa esta metodología porque no es del todo fiable, se necesita tener más conocimiento sobre el paciente realizando otras pruebas y haciendo uso del historial médico.

Con lo mencionado en el párrafo anterior, se llega a una conclusión parcial de que los resultados obtenidos no van ser los mejores, es decir, no se va a tener una precisión del 100%, ya que diagnosticar una neumonía a partir de una sola radiografía es una tarea complicada, se necesita más información.

En la segunda parte del presente documento se realizó el desarrollo del mismo, diseñando diferentes experimentos y obteniendo resultados diversos. El mejor de los modelos obtenidos para resolver la problemática que se plantea, fue el modelo *EfficientNetB5*, consiguiendo un *f1-score* igual a 0.61 y un AUC igual a 0.88.

Aunque el AUC sí que es elevado y el *f1-score* es algo superior al 50%, se podría mejorar estos resultados, es verdad que a la hora de diagnosticar los casos positivos no se ha obtenido la mejor de las precisiones, ya que se consigue una sensibilidad del 60%. Tal y como comentábamos identificar la neumonía no es tan fácil a partir de una radiografía, ya que en caso contrario este porcentaje sería mayor.

Sin embargo el modelo seleccionado como mejor, sí que es capaz de identificar de forma correcta cuando una radiografía no presenta síntomas de esta patología, se consigue una especificidad del 89%.

En resumen, los resultados que hemos logrado son en líneas generales buenos, pero a la hora de diagnosticar casos positivos de neumonía se obtiene

un error elevado. El porqué de ese elevado error se desconoce, pero podemos intuir que se debe a la dificultad del problema en sí.

Cabe mencionar que en el repositorio oficial de *Kaggle* [15] hay diversas soluciones implementadas, tanto para realizar tareas de segmentación como para clasificación. Los resultados obtenidos para la tarea que nos interesa, la clasificación, son similares a los logrados en este proyecto, es decir, no presentan una elevada tasa de éxito en el diagnóstico de la neumonía a partir de una radiografía, por lo que llegamos a la conclusión que los resultados conseguidos son el límite para nuestro problema.

De todas formas, aunque no se logra una elevada tasa de éxito en la identificación de la neumonía, estos datos son útiles para complementarlos con otras pruebas ya mencionadas en el primer apartado del presente documento.

Esta complementación entre otro tipo de pruebas y resultados obtenidos de nuestro modelo va a conseguir simplificar el diagnóstico de esta enfermedad, ya que por ejemplo con una aplicación informática podríamos saber de una forma rápida si un paciente padece neumonía o no, solamente necesitando subir la imagen de la radiografía. Consiguiendo así descongestionar el sistema sanitario que tal y como se pudo ver en el año del coronavirus es de vital importancia.

Debido a todo esto, podemos llegar a la conclusión de que el campo de la biología computacional cada día va adquiriendo un mayor protagonismo, pero los algoritmos que se hacen uso para resolver este tipo de problemas no están del todo avanzados para determinadas tareas. Es por ello, que es necesaria la continua investigación tanto del *hardware* como *software*, permitiendo así grandes capacidades de cómputo y nuevos algoritmos para conseguir resolver la problemática que se plantea en este proyecto con una muy alta precisión.

En lo que respecta al seguimiento del plan del proyecto, se ha cumplido de forma casi idónea sobre la planificación que se realizó al comienzo del semestre. Cabe destacar que por motivos de desconocimiento en este campo y otros motivos personales, la fase de desarrollo de la red neuronal (*transfer learning* y *fine-tuning*) se empezó a implementar dos semanas más tarde de la planificación inicial, haciendo que se llegara de forma apresurada a la entrega de la PEC3, por todo lo demás se siguió y se cumplió la planificación gracias también a la metodología usada, *Cross Industry Standard Process for Data Mining (CRISP-DM)* [18].

Para concluir, me gustaría hacer mención a la importancia que tiene el uso de redes neuronales artificiales en el ámbito sanitario, ya que redes que proporcionan grandes resultados van a permitir avanzar mucho a la sociedad, solventando problemas que a día de hoy son de gran dificultad. Es por ello, que se debe continuar con la investigación y el desarrollo de algoritmos relativos al aprendizaje profundo con el fin de mejorar los resultados obtenidos, además de poder aplicarlo a otros ámbitos.

7. Líneas de trabajo futuro

En cuanto a las líneas de trabajo futuras que se pueden alcanzar tanto en este proyecto como en el ámbito del *deep learning* encontramos:

- Probar otros modelos para realizar el *transfer learning*, es decir, hacer uso de *EfficientNetB6* y *EfficientNetB7* que por motivos de tiempo no se han podido realizar. También se podría estudiar el probar otros modelos pre-entrenados como *VGG16*, *RESNET52*, *XCEPTION*, *DENSENET169*, *DENSENET121*...
- Hacer uso de otras tecnologías para mejorar los resultados obtenidos. En el entrenamiento del modelo se usó *Google Colab*, pero puede ser que otras plataformas como *Kaggle* permitan un uso de *GPU* o *TPU* superior.
- Aunque para el entrenamiento de la red se hace uso de un gran número de imágenes, y además se realiza un aumento de datos para dotar de mayor variabilidad al modelo, se puede investigar si haciendo uso de otros parámetros o creando más capas de aumento de datos se consigue una mayor precisión.
- Crear una aplicación web o móvil que permita el diagnóstico de la neumonía, es decir, que de una forma fácil se pueda explotar el modelo creado con *EfficientNetB5*, haciendo que con solamente subir la radiografía en formato “*PNG*” se pueda indicar a un usuario si padece neumonía, evitando así colapsar el sistema sanitario de urgencias.
- Debido a la gran importancia que está adquiriendo el *deep learning* y sobre todo en la aplicación al ámbito sanitario, una buena línea de investigación sería seguir desarrollando algoritmos para mejorar la predicción. Muchos de los algoritmos usados hoy en día son de la década de los 70 y 80 debido a que ahora se puede hacer uso de esa capacidad de cómputo necesaria, pero si seguimos mejorando tanto en el desarrollo de algoritmos como en el *hardware* dentro de otros 50 años los resultados obtenidos pueden ser muy precisos.

8. Glosario

CXR: Radiografías de tórax

OMS: Organización Mundial de la Salud.

SEPAR: Sociedad Española de la Neumología y Cirugía torácica.

CNN: *Convolutional Neural Network*.

RNC: Red Neuronal Convolucional.

CRISP-DM: *Cross Industry Standard Process for Data Mining*.

FANG: *Facebook, Amazon, Netflix y Google*.

AUC: *Area Under Curve*

9. Bibliografía

- [1] «Neumonía». <https://medlineplus.gov/spanish/pneumonia.html> (accedido 22 de febrero de 2022).
- [2] «Neumonía», *Wikipedia, la enciclopedia libre*. 22 de febrero de 2022. Accedido: 22 de febrero de 2022. [En línea]. Disponible en: <https://es.wikipedia.org/w/index.php?title=Neumon%C3%ADa&oldid=141845195>
- [3] «SARS-CoV-2», *Wikipedia, la enciclopedia libre*. 16 de febrero de 2022. Accedido: 23 de febrero de 2022. [En línea]. Disponible en: <https://es.wikipedia.org/w/index.php?title=SARS-CoV-2&oldid=141723154>
- [4] «La neumonía causa 10.000 muertes anuales, muchas de ellas prevenibles», *El Independiente*, 6 de noviembre de 2019. <https://www.elindependiente.com/vida-sana/salud/2019/11/07/la-neumonia-causa-10-000-muertes-anuales-muchas-de-ellas-prevenibles/> (accedido 22 de febrero de 2022).
- [5] «Neumonía». <https://www.who.int/es/news-room/fact-sheets/detail/pneumonia> (accedido 22 de febrero de 2022).
- [6] L. Kong y J. Cheng, «Based on improved deep convolutional neural network model pneumonia image classification», *PLOS ONE*, vol. 16, n.º 11, p. e0258804, nov. 2021, doi: 10.1371/journal.pone.0258804.
- [7] J. L. García Satué y J. Aspa Marco, *Neumonías*. Majadahonda (Madrid: Ergon, 2006).
- [8] «Agente (medicina)», *Wikipedia, la enciclopedia libre*. 17 de febrero de 2022. Accedido: 6 de marzo de 2022. [En línea]. Disponible en: [https://es.wikipedia.org/w/index.php?title=Agente_\(medicina\)&oldid=141730538](https://es.wikipedia.org/w/index.php?title=Agente_(medicina)&oldid=141730538)
- [9] «Neumonía», *Wikipedia, la enciclopedia libre*. 27 de febrero de 2022. Accedido: 6 de marzo de 2022. [En línea]. Disponible en: <https://es.wikipedia.org/w/index.php?title=Neumon%C3%ADa&oldid=141952610>
- [10] «Enfermedad Neumocócica (Incluye la neumonía neumocócica, la meningitis neumocócica y la bacteremia neumocócica)». https://www.health.ny.gov/es/diseases/communicable/pneumococcal/fact_sheet.htm (accedido 6 de marzo de 2022).
- [11] N. estafilocócica N. causada por un estafilococos A. transmisor: S. aureus, «Neumonía Estafilocócica - EcuRed». https://www.ecured.cu/Neumon%C3%ADa_Estafiloc%C3%B3cica (accedido 6 de marzo de 2022).
- [12] «Infecciones por Klebsiella, Enterobacter y Serratia - Enfermedades infecciosas», *Manual Merck versión para profesionales*. <https://www.merckmanuals.com/es-us/professional/enfermedades-infecciosas/bacilos-gramnegativos/infecciones-por-y> (accedido 6 de marzo de 2022).
- [13] J. I. Sánchez Olmedo, V. Jorge Amigo, M. Pérez Alé, J. Garnacho Montero, F. J. Jiménez Jiménez, y C. Ortiz Leyba, «Neumonía comunitaria grave por Legionella pneumophila», *Med. Intensiva*, vol. 26, n.º 10, pp. 504-507, dic. 2002.

- [14] V. Jain, R. Vashisht, G. Yilmaz, y A. Bhardwaj, «Pneumonia Pathology», en *StatPearls*, Treasure Island (FL): StatPearls Publishing, 2022. Accedido: 18 de mayo de 2022. [En línea]. Disponible en: <http://www.ncbi.nlm.nih.gov/books/NBK526116/>
- [15] «RSNA Pneumonia Detection Challenge». <https://kaggle.com/c/rsna-pneumonia-detection-challenge> (accedido 23 de febrero de 2022).
- [16] R. Alberto, «Redes Neuronales: ¿Qué es Transfer Learning y Fine Tuning?», *Medium*, 4 de julio de 2021. <https://rubialesalberto.medium.com/redes-neuronales-qu%C3%A9-es-transfer-learning-y-fine-tuning-8259a81cfdbc> (accedido 13 de mayo de 2022).
- [17] «DICOM», *Wikipedia, la enciclopedia libre*. 27 de enero de 2022. Accedido: 23 de febrero de 2022. [En línea]. Disponible en: <https://es.wikipedia.org/w/index.php?title=DICOM&oldid=141264476>
- [18] «CRISP-DM: La metodología para poner orden en los proyectos», *Sngular*, 2 de agosto de 2016. <https://www.sngular.com/es/data-science-crisp-dm-metodologia/> (accedido 24 de febrero de 2022).
- [19] J. F. V. Rueda, «CRISP-DM: una metodología para minería de datos en salud», *healthdataminer.com*, 4 de noviembre de 2019. <https://healthdataminer.com/data-mining/crisp-dm-una-metodologia-para-mineria-de-datos-en-salud/> (accedido 24 de febrero de 2022).
- [20] M. Ubierna San Mamés, *Pneumonia-Detection*. 2022. Accedido: 13 de mayo de 2022. [En línea]. Disponible en: <https://github.com/mariouism/Pneumonia-Detection>
- [21] «PubMed», *PubMed*. <https://pubmed.ncbi.nlm.nih.gov/> (accedido 7 de marzo de 2022).
- [22] S. S. Yadav y S. M. Jadhav, «Deep convolutional neural network based medical image classification for disease diagnosis», *J. Big Data*, vol. 6, n.º 1, p. 113, dic. 2019, doi: 10.1186/s40537-019-0276-2.
- [23] R. H. Abiyev y A. Ismail, «COVID-19 and Pneumonia Diagnosis in X-Ray Images Using Convolutional Neural Networks», *Math. Probl. Eng.*, vol. 2021, p. e3281135, nov. 2021, doi: 10.1155/2021/3281135.
- [24] A. F. R. Gómez, C. A. C. Guarnizo, A. M. G. Pomarico, y S. L. C. Suárez, «ALGORITMO PARA LA CARACTERIZACIÓN DE NEUMONÍA EN RADIOGRAFÍA DE TÓRAX», *Encuentro Int. Educ. En Ing.*, ago. 2020, Accedido: 3 de marzo de 2022. [En línea]. Disponible en: <https://acofipapers.org/index.php/eiei/article/view/794>
- [25] «Convolutional Neural Network Based Classification of Patients with Pneumonia using X-ray Lung Images», *Journal*. <https://astesj.com/v05/i05/p22/> (accedido 3 de marzo de 2022).
- [26] L. Kong y J. Cheng, «Based on improved deep convolutional neural network model pneumonia image classification», *PLOS ONE*, vol. 16, n.º 11, p. e0258804, nov. 2021, doi: 10.1371/journal.pone.0258804.
- [27] «Python: qué es y por qué deberías aprender a utilizarlo». <https://www.becas-santander.com/es/blog/python-que-es.html> (accedido 18 de mayo de 2022).
- [28] «Anaconda | The World's Most Popular Data Science Platform», *Anaconda*. <https://www.anaconda.com/> (accedido 18 de mayo de 2022).
- [29] «NumPy». <https://numpy.org/> (accedido 18 de mayo de 2022).

- [30] «pandas - Python Data Analysis Library». <https://pandas.pydata.org/> (accedido 18 de mayo de 2022).
- [31] «os — Miscellaneous operating system interfaces — Python 3.10.4 documentation». <https://docs.python.org/3/library/os.html> (accedido 18 de mayo de 2022).
- [32] «Pydicom |». <https://pydicom.github.io/> (accedido 18 de mayo de 2022).
- [33] «Pillow». <https://pillow.readthedocs.io/en/stable/index.html> (accedido 18 de mayo de 2022).
- [34] «random — Generate pseudo-random numbers — Python 3.10.4 documentation». <https://docs.python.org/3/library/random.html> (accedido 18 de mayo de 2022).
- [35] «pickle — Python object serialization — Python 3.10.4 documentation». <https://docs.python.org/3/library/pickle.html> (accedido 18 de mayo de 2022).
- [36] «scikit-image 0.19.2 docs — skimage v0.19.2 docs». <https://scikit-image.org/docs/stable/> (accedido 18 de mayo de 2022).
- [37] «scikit-learn: machine learning in Python — scikit-learn 1.1.0 documentation». <https://scikit-learn.org/stable/> (accedido 18 de mayo de 2022).
- [38] M. Waskom, «seaborn: statistical data visualization», *J. Open Source Softw.*, vol. 6, n.º 60, p. 3021, abr. 2021, doi: 10.21105/joss.03021.
- [39] «TensorFlow», *TensorFlow*. <https://www.tensorflow.org/?hl=es-419> (accedido 18 de mayo de 2022).
- [40] «Keras: the Python deep learning API». <https://keras.io/> (accedido 18 de mayo de 2022).
- [41] «Matplotlib — Visualization with Python». <https://matplotlib.org/> (accedido 18 de mayo de 2022).
- [42] BBVA, «Te contamos qué es el “machine learning” y cómo funciona», *BBVA NOTICIAS*, 8 de noviembre de 2019. <https://www.bbva.com/es/machine-learning-que-es-y-como-funciona/> (accedido 19 de mayo de 2022).
- [43] «Diferencia entre neurona y neuroglía - Centros EQ & Psicolab, centro de Psicología, Neuropsicología, Logopedia, Pedagogía en Benalmádena y Málaga». <https://www.psicolab.com/diferencia-entre-neurona-y-neuroglia/> (accedido 19 de mayo de 2022).
- [44] «A. Requena y col.: “Nuevas Tecnologías y Contaminación de Atmósferas para PYMES”». <https://www.um.es/LEQ/Atmosferas/Ch-VI-3/F63s4p3.htm> (accedido 19 de mayo de 2022).
- [45] «Qué son las redes neuronales y sus funciones», *ATRIA Innovation*, 22 de octubre de 2019. <https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/> (accedido 19 de mayo de 2022).
- [46] «Redes neuronales convolucionales». <https://es.mathworks.com/discovery/convolutional-neural-network-matlab.html> (accedido 6 de marzo de 2022).
- [47] D. Lerch, «Deep Learning con redes pre-entrenadas en ImageNet», *Neuron4*, 28 de noviembre de 2021. <https://medium.com/neuron4/redes-pre-entrenadas-en-imagenet-30d858c37b1f> (accedido 19 de mayo de 2022).
- [48] M. Tan y Q. V. Le, «EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks». arXiv, 11 de septiembre de 2020.

- Accedido: 13 de mayo de 2022. [En línea]. Disponible en: <http://arxiv.org/abs/1905.11946>
- [49] «Social Network for Programmers and Developers». <https://morioh.com> (accedido 13 de mayo de 2022).
 - [50] «Radiological Society of North America». <https://www.rsna.org/> (accedido 13 de mayo de 2022).
 - [51] «National Institutes of Health (NIH)», *National Institutes of Health (NIH)*. <https://www.nih.gov/> (accedido 13 de mayo de 2022).
 - [52] «Society of Thoracic Radiology». <https://thoracicrad.org/> (accedido 13 de mayo de 2022).
 - [53] «MD.ai». <https://www.md.ai/> (accedido 13 de mayo de 2022).
 - [54] «Google Colaboratory». <https://colab.research.google.com/?hl=es> (accedido 13 de mayo de 2022).
 - [55] «¿Qué es el formato DICOM? Las claves del estándar en imágenes médicas», *Clinic Cloud*, 22 de julio de 2021. <https://clinic-cloud.com/blog/formato-dicom-que-es-estandar-imagenes-medicas/> (accedido 13 de mayo de 2022).
 - [56] K. Team, «Keras documentation: Transfer learning & fine-tuning». https://keras.io/guides/transfer_learning/ (accedido 13 de mayo de 2022).
 - [57] K. Team, «Keras documentation: Image classification via fine-tuning with EfficientNet». https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/#tips-for-fine-tuning-efficientnet (accedido 13 de mayo de 2022).
 - [58] «La matriz de confusión y sus métricas – Inteligencia Artificial –», *Juan Barrios*, 26 de julio de 2019. <https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/> (accedido 22 de mayo de 2022).
 - [59] «Curva ROC», *Wikipedia, la enciclopedia libre*. 3 de abril de 2022. Accedido: 22 de mayo de 2022. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Curva_ROC&oldid=142674318