Universitat Oberta de Catalunya

uoc.edu

Detección precoz de covid-19 a partir de imágenes de radiografías de tórax mediante redes neuronales convolucionales

Blanca Alfonso López
Máster Universitario en Bioinformática y Bioestadística
Área 4

Consultora: Romina Astrid Rebrij
Profesor responsable de la asignatura: Antoni Pérez Navarro

Junio 2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Detección precoz de covid-19 a partir de imágenes de radiografías de tórax mediante redes neuronales convolucionales*
Nombre del autor:	*Blanca Alfonso López*
Nombre del consultor/a:	*Romina Astrid Rebrij*
Nombre del PRA:	*Antoni Pérez Navarro*
Fecha de entrega:	06/2021
Titulación:	*Máster Universitario en Bioinformática y Bioestadística*
Área del Trabajo Final:	*Área 4*
Idioma del trabajo:	Castellano
Número de créditos:	15
Palabras clave	*Covid-19, Aprendizaje profundo, Rayos-X*

Resumen del Trabajo:

En la actualidad el mundo se encuentra enfrentado a una crisis sanitaria global provocada por la enfermedad Covid-19. Desde su aparición en Wuhan (China) el virus ha causado alrededor de cuatro millones de muertes en el mundo provocando una alta presión hospitalaria.

El virus destaca por su alta capacidad de propagación. Por ello, parece indispensable contar con mecanismos de detección precoz de la enfermedad para aumentar la rapidez del tratamiento y reducir la probabilidad de contagio.

Las radiografías de tórax son una prueba muy efectiva para la detección de neumonía a causa de Covid-19. El aprendizaje profundo es capaz de extraer características relacionadas con los resultados clínicos de las radiografías.

En este Trabajo de Fin de Máster se ha propuesto un modelo de red neuronal convolucional y se ha intentado mejorar mediante la utilización de las técnicas de aumento de datos, transferencia de aprendizaje y ajuste fino. El proyecto se ha desarrollado en el lenguaje de programación Python mediante la utilización de las librerías Tensorflow y Keras.

Para el entrenamiento del modelo se ha utilizado una base de datos de imágenes de radiografías de tórax del repositorio Kaggle. Dichas imágenes se encuentran clasificadas en cuatro categorías: normal, opacidad torácica, neumonía viral y neumonía por Covid-19.

Finalmente se ha realizado una comparación de los modelos creados. El considerado mejor modelo ha sido utilizado para mostrar los resultados

mediante una aplicación web desarrollada con Flask y publicada en la plataforma PythonAnywhere.

Abstract:

The world is currently facing a global health crisis caused by the Covid-19 disease. Since its appearance in Wuhan (China), the virus has caused around four million deaths worldwide, causing high hospital pressure.

The virus stands out for its high propagation capacity. For this reason, it seems essential to have mechanisms for early detection of the disease to increase the speed of treatment and reduce the probability of contagion.

Chest X-rays are a very effective test for detecting Covid-19 pneumonia. Deep learning is able to extract characteristics related to clinical results from radiographs.

In this Master's Thesis, a convolutional neural network model has been proposed and an attempt has been made to improve it through the use of data augmentation, learning transfer and fine-tuning techniques. The project has been developed in the Python programming language and using the Tensorflow and Keras libraries.

For the training of the model, a database of images of chest radiographs from the Kaggle repository was used. These images are classified into four categories: normal, thoracic opacity, viral pneumonia and Covid-19 pneumonia.

Finally, a comparison of the created models has been made. The considered best model has been used to show the results through a web application developed with Flask and published on the PythonAnyWhere platform.

Dedicatoria

A todos mis compañeros sanitarios y en especial al servicio de radiología del Hospital Universitario Severo Ochoa de Leganés, por ser un equipo a pesar de los duros y agotadores momentos vividos durante este último año.

INDICE

1. Resumen 9
2. Introducción 11
2.1. Contexto y justificación del trabajo 11
2.2. Objetivos 15
2.3. Enfoque y método seguido 15
2.4. Planificación del trabajo 17
2.5. Breve sumario de contribuciones y productos obtenidos 21
2.6. Breve descripción de los otros capítulos de la memoria 22
3. Estado del arte 22
3.1. Artículos relacionados 22
3.2. Inteligencia artificial 28
3.3. Aprendizaje automático 30
3.4. Aprendizaje profundo 31
3.5. Redes neuronales convolucionales 38
4. Metodología 42
4.1. Conjunto de datos 42
4.2. Entorno de trabajo 42
4.3. Lenguaje de programación 42
4.4. Entorno de programación 43
4.5. Librerías utilizadas 44
4.6. Importación de datos 44
4.7. Análisis exploratorio de los datos 45
4.8. Reducción del conjunto de datos 46
4.9. Preprocesamiento de los datos 47
4.10. Construcción del modelo 49
4.11. Entrenamiento y validacón del modelo 51
4.12. Mejora del modelo 51
4.13. Diseño de la aplicación web 53
4.14. Publicación de la aplicación web 54
5. Resultados 54
5.1. Exposición de los resultados de cada modelo 54
5.2. Comparación de modelos 62
5.3. Elección final de un modelo 62
5.4 Uso de la aplicación web 64
6. Discusión 65
7. Conclusiones 65
8. Glosario 67
9. Bibliografía 68

Lista de ilustraciones

Ilustración 1. Evolución de la incidencia acumulada en España. Fuente: Datos RTVE [6]..... 13
Ilustración 2. Diagrama de Gantt: Calendario del proyecto. Fuente: Elaboración propia 19
Ilustración 3. Hitos del proyecto. Fuente: elaboración propia..... 20
Ilustración 4. Tipos de aprendizaje automático. Fuente: [34] 30
Ilustración 5. Inteligencia artificial, subcampos. Fuente: [41] 31
Ilustración 6. Estructura de una neurona. Fuente: 38..... 32
Ilustración 7. Red neuronal artificial sencilla. Fuente: [41] 33
Ilustración 8. Función de una neurona. Fuente: 42 33
Ilustración 9. Tipos de capas de una red neuronal - Fuente: [44]..... 35
Ilustración 10. Algoritmo entrenamiento redes monocapa. Fuente: [44] 36
Ilustración 11. Convolución imagen a color. Fuente: [47]..... 40
Ilustración 12. Max Pooling. Fuente: 48 41
Ilustración 13. Ejemplo imagen Radiografía de tórax. Fuente: elaboración propia. 45
Ilustración 14. Gráfico circular: imágenes agrupadas por clase. Fuente: elaboración propia. 46
Ilustración 15. Modelo secuencial red neuronal convolucional Fuente: elaboración propia. 50
Ilustración 16. Visualización de la precisión durante el entrenamiento y validación del modelo general. Fuente: elaboración propia. 55
Ilustración 17. Visualización de la función de pérdida durante el entrenamiento y validación del modelo general. Fuente: elaboración propia 55
Ilustración 18. Visualización de la precisión durante el entrenamiento y validación del modelo con aumento de datos. Fuente: elaboración propia. 56
Ilustración 19. Visualización de la función de pérdida durante el entrenamiento y validación del modelo con aumento de datos. Fuente: elaboración propia 57
Ilustración 20. Visualización de la precisión durante el entrenamiento y validación del modelo VGG16 sin AD. Fuente: elaboración propia. 58
Ilustración 21. Visualización de la función de pérdida durante el entrenamiento y validación del modelo VGG16 sin AD Fuente: elaboración propia 58
Ilustración 22. Visualización de la precisión durante el entrenamiento y validación del modelo VGG16 con AD. Fuente: elaboración propia. 59
Ilustración 23. Visualización de la función de pérdida durante el entrenamiento y validación del modelo VGG16 con AD Fuente: elaboración propia. 59
Ilustración 24. Visualización de la precisión durante el entrenamiento y validación del modelo VGG16 con ajuste fino y AD. Fuente: elaboración propia. 60
Ilustración 25. Visualización de la función de pérdida durante el entrenamiento y validación del modelo VGG16 con ajuste fino y AD Fuente: elaboración propia. 61

Ilustración 26. Pantalla principal aplicación web. Fuente: elaboración propia. . 63
Ilustración 27. Seleccionar una imagen en la aplicación web. Fuente: elaboración propia. 63
Ilustración 28. Imagen seleccionada en la aplicación web. Fuente: elaboración propia. 64
Ilustración 29. Resultado predicción aplicación web. Fuente: elaboración propia. 64

Lista de tablas

Tabla 1. Número de muertes a consecuencia de Covid-19 por continente. Fuente: Orus, A[5]---------- 12
Tabla 2. Análisis de riesgos del proyecto. Fuente: elaboración propia. ---------- 21
Tabla 3. Características artículos relacionados. Fuente: Elaboración propia--- 23
Tabla 4. Continuación: Características artículos relacionados. Fuente: Elaboración propia---------- 24
Tabla 5. Continuación: Características artículos relacionados. Fuente: Elaboración propia.---------- 25
Tabla 6. Tipos de función de activación. Fuente: [43]---------- 34
Tabla 7. Métricas modelo general. Fuente: elaboración propia. ---------- 55
Tabla 8. Métrica modelo con aumento de datos. Fuente: elaboración propia. - 57
Tabla 9. Métricas modelo VGG16 sin AD. Fuente: elaboración propia.--------- 58
Tabla 10. Métricas modelo VGG16 con AD. Fuente: elaboración propia. ------- 60
Tabla 11. Métricas modelo VGG16 con ajuste fino y AD. Fuente: elaboración propia.---------- 61
Tabla 12. Comparación de modelos. Fuente: elaboración propia. ---------- 62

1 Resumen

Antecedentes

El Covid-19, que apareció por primera vez a finales de 2019 en Wuhan (China), ha causado más de cuatro millones de muertes en todo el mundo. Caracterizado por tratarse de un virus altamente contagioso, su expansión se produjo de forma rápida por todos los continentes.

La enfermedad causada por el Covid-19 puede cursar desde un cuadro asintomático hasta un síndrome de insuficiencia respiratoria grave originando la muerte. La evolución grave de la enfermedad se caracteriza por la aparición de una neumonía característica, la neumonía Covid-19. Por ello, una de las formas que utilizan los profesionales sanitarios para el diagnóstico de Covid-19 es mediante una radiografía de tórax.

Método

Se ha diseñado un modelo de red neuronal convolucional capaz de clasificar imágenes de radiografías de tórax en cuatro clases: normal, neumonía viral, neumonía Covid-19 y opacidad torácica. Para el entrenamiento del modelo se han utilizado las imágenes de un conjunto de datos abierto disponible en Kaggle. Con el objetivo de mejorar el rendimiento del modelo se han utilizado las técnicas de aumento de datos, aprendizaje por transferencia, utilizando la red pre-entrenada VGG-16, y ajuste fino. El modelo con mejor rendimiento se ha utilizado para la creación de una aplicación web interactiva, creada con Flask y publicada en PythonAnywhere.

Resultados

El modelo que tiene mejor rendimiento es aquel que utiliza la red pre-entrenada VGG-16 junto con aumento de datos.

Esta red se ha utilizado para la creación de la aplicación web. Dicha aplicación es capaz de procesar una imagen subida por el usuario y realizar una predicción.

Conclusiones

Se ha conseguido crear una red neuronal convolucional con un nivel de precisión del 87%. Esta precisión podría ser aumentada implementando los modelos en entornos de trabajo GPU con mayor memoria, en los que se podría modificar parámetros del entrenamiento como el aumento del número de épocas.

El aprendizaje profundo ha demostrado ser de gran utilidad en la detección del Covid-19.

Con este trabajo se pretende mostrar una herramienta de ayuda para los profesionales sanitarios en la detección del Covid-19. La aplicación web tiene un funcionamiento sencillo que podría ser mejorado e implementado en el entorno hospitalario.

2 Introducción

2.1. Contexto y justificación del trabajo

2.1.1. Descripción general

En este trabajo se desarrolla un algoritmo que permite el procesamiento y clasificación de imágenes médicas de radiografías de tórax con el propósito de detectar neumonía causada por Covid-19 y diferenciarla de otras afectaciones pulmonares. Para ello se hace uso del aprendizaje profundo mediante la construcción de modelos de redes neuronales convolucionales.

Para mostrar los resultados se diseña y desarrolla una aplicación web capaz de realizar predicciones sobre una imagen de radiografía de tórax subida por el usuario.

2.1.2. Contexto

En la actualidad el mundo se encuentra enfrentado a una crisis sanitaria global provocada por la enfermedad Covid-19.

Dicha enfermedad ha supuesto un gran desafío en numerosos países debido principalmente a su alta tasa de contagio [1].

El inicio del Covid-19 lo situamos en Wuhan (China), en diciembre de 2019, cuando se reportaron los primeros casos de neumonía de causa desconocida. Dichos casos tenían en común la exposición a un mercado de productos del mar de Huanan que comerciaba con animales vivos [2]. Así, el primer indicio de enfermedad ocurre el 7 de diciembre de 2019 cuando aparecen los síntomas del primer caso.

China notificó el brote el 31 de diciembre de 2019 a la OMS y el 7 de enero de 2020 el virus fue identificado como un coronavirus con más del 70% de similitud con el SARS-CoV y más del 95% de homología con el coronavirus de murciélago [2].

A partir de entonces empezaron a surgir los primeros contagios en personas que no habían acudido a aquel mercado, suponiendo así que se estaba produciendo la transmisión de persona a persona.

Así la propagación del virus comenzó en China, agravándose con la *"migración masiva de chinos durante el Año Nuevo"* [2]. El 23 de enero de 2020 se cerró la ciudad de Wuhan. La infección siguió transmitiéndose por Asia hasta llegar a Irán e incluso Italia, donde se produjeron los primeros grandes brotes [3].

El 11 de marzo de 2020 la OMS declaró pandemia global [4]. Desde entonces el virus ha ido expandiéndose de forma sin precedentes por todo el mundo, aumentando los casos de manera exponencial y llegando a un *"tiempo de duplicación de la epidemia de 1.8 días"* [2].

Actualmente, el virus ha causado alrededor de 4 millones de muertes en todo el mundo. La cifra de muertes según el continente se puede ver en la siguiente tabla:

Continente	Número de muertes
América	1.808.880
Europa	1.143.108
Asia	652.295
África	132.347
Oceanía	1.394

Tabla 1. Número de muertes a consecuencia de Covid-19 por continente. Fuente: Orus, A[5]

En la Tabla 1 podemos observar cómo a pesar de que el virus se originó en el continente asiático, el número de muertes de Europa duplica a las ocurridas en el continente de origen y América se acerca a los dos millones de fallecidos.

Estados Unidos se sitúa como el país con más muertes por coronavirus mientras que Perú es el país con más muertes por coronavirus por cada 100.000 habitantes. [6]

En España, el impacto de la pandemia también ha sido muy importante. Fue en la Comunidad Valenciana donde se confirmó el primer fallecido a causa del Covid-19. Desde aquel marzo de 2020 han fallecido en nuestro país, según cifras oficiales, más de 80.000 personas. [7]

Después del origen del coronavirus en marzo de 2020, España, al igual que la mayoría de países, ha sufrido una serie de recaídas durante la pandemia, denominadas "olas" y caracterizadas por un aumento del número de contagios, de hospitalizaciones y de muertes trayendo como consecuencia un aumento de la presión hospitalaria. En la Ilustración 1 podemos observar la evolución de la incidencia acumulada en España desde la segunda ola hasta la actualidad.

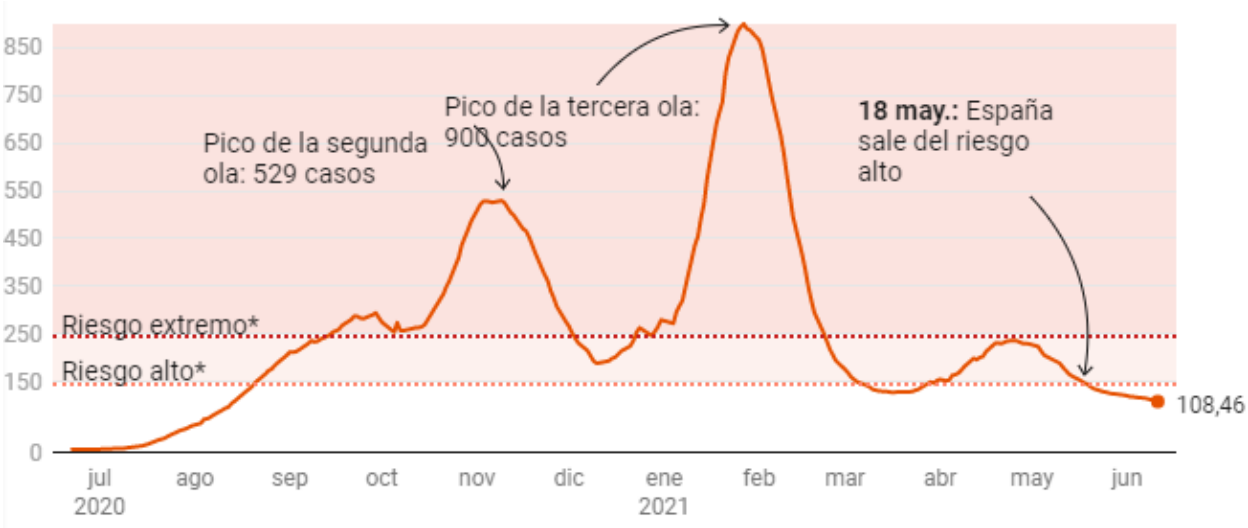

Ilustración 1. Evolución de la incidencia acumulada en España. Fuente: Datos RTVE [6]

Además del impacto en la salud de las personas, la pandemia ha ocasionado una serie de consecuencias sociales y económicas importantes. Disminución del número de personas afiliadas a la Seguridad Social, más de un millón de trabajadores en situación de ERTE, cierre de negocios, separación de familias, aumento de problemas psicológicos… son solo algunos ejemplos del impacto de la pandemia en la vida de las personas.

Actualmente el mundo mira hacia el futuro con ligero optimismo gracias a las vacunas. Desde el comienzo de la vacunación a principios de 2021 el número de contagios y de muertes ha descendido de forma notable. En nuestro país ya hay más de 12 millones de personas que han recibido la pauta completa de vacunación correspondientes a un 25.8% de la población. Las vacunas administradas en España son de tipo Pfizer, Moderna, AstraZeneca y Jansen, aunque se espera que este verano se introduzcan otros tipos de vacunas. [8]

Se ha demostrado que la enfermedad por Covid1-19 puede afectar a cualquier individuo sin importar el género, la edad o la etnia a la que pertenezca.

La enfermedad puede cursar desde un estado asintomático hasta un *"síndrome de dificultad respiratoria y disfunción multiorgánica"* [2] que termina con la muerte.

Los síntomas que aparecen más comúnmente son la tos, dolor de garganta y de cabeza, fatiga, fiebre y dificultad para respirar [2]. La aparición de neumonía se produce normalmente una semana después de la presencia de los primeros síntomas, siendo el periodo de incubación de la enfermedad de 5.1 días. Dicha neumonía puede cursar de forma grave provocando insuficiencia respiratoria y muerte.

La progresión de esta neumonía *"se asocia con un aumento extremo de citocinas inflamatorias, incluidas IL2, IL7, IL10, GCSF, IP10, MCP1, MIP1A y TNFα"* [9].

Desde que una persona enferma por Covid-19 hasta su recuperación el tiempo medio es de 2 semanas cuando ha sido un cuadro leve o de 3 a 6 semanas cuando la persona ha sufrido un cuadro grave.

La gravedad de la enfermedad depende de muchos factores, tanto propios de la persona (edad, estado inmunológico, obesidad…) como ajenos a la misma (atención sanitaria, demografía, efectividad de las vacunas). Los grupos que se consideran con mayor riesgo de sufrir un cuadro grave son los siguientes [10]:

- Edad avanzada.
- Enfermedades cardiovasculares.
- Diabetes.
- EPOC.
- Cáncer.
- Inmunodepresión.
- Embarazo
- Enfermedades crónicas.

2.1.3. Justificación

Las pruebas de imagen han adquirido un papel muy importante en la detección del coronavirus, ayudando al diagnóstico y manejo de los pacientes, determinando la gravedad y guiando su tratamiento [11]. *"La radiografía de tórax es generalmente la primera prueba de imagen en los pacientes con sospecha o confirmación de COVID-19 por su utilidad, disponibilidad y bajo coste"* (Orozco C., et al)[11]. Incluso se acepta la radiografía de tórax como método de triaje en algunos escenarios, destacando así su capacidad para diagnosticar Covid-19 [11].

Debido a la situación actual de pandemia, los servicios de radiodiagnóstico han tenido que enfrentarse a un verdadero reto para poder informar el gran volumen de radiografías de tórax realizadas [11]. Los recursos sanitarios son limitados y surge la necesidad de agilizar el proceso de detección del virus.

El aprendizaje profundo, mediante el procesamiento y clasificación de imágenes médicas, podría ayudar a los sanitarios en el diagnóstico médico además de reducir el tiempo de evaluación. Por todo lo anterior, parece primordial contar con mecanismos de detección rápida del coronavirus, siendo esta la única manera de lograr un control de la propagación del virus [12]. Además los resultados de una radiografía de tórax no solo nos indican si el paciente tiene o no coronavirus, sino que también puede indicarnos la gravedad de la afectación

de tórax que sufre el paciente, orientando así el tratamiento y la prevención del mismo.

El aprendizaje profundo ha demostrado resultados prometedores en el análisis de enfermedades pulmonares [13]. Gracias a la capacidad de aprendizaje de características, el aprendizaje profundo es capaz de extraer las características relacionadas con los resultados clínicos de las radiografías de tórax de manera automática [13]. *"Las características aprendidas por los modelos de aprendizaje profundo pueden reflejar mapeos abstractos de alta dimensión que son difíciles de percibir para los humanos pero que están fuertemente asociados con los resultados clínicos"* (Brownlee, J., 2019)[13].

2.2. Objetivos del trabajo

2.2.1. Objetivo general:

El objetivo general del proyecto es desarrollar un algoritmo capaz de distinguir entre radiografías de pacientes sanos, con neumonía viral, con opacidad torácica y con neumonía a causa de la Covid-19 mediante la aplicación de técnicas de aprendizaje profundo.

2.2.2. Objetivos específicos:

A continuación se detallan los objetivos específicos:

- Crear una red neuronal convolucional (CNN) que identifique patrones característicos de Covid-19 en radiografías de tórax diferenciándola de la neumonía tipo viral y la opacidad torácica.
- Obtener un nivel de precisión superior al 70% en el diagnóstico de neumonía por SARS-CoV-2.
- Desarrollar una aplicación web interactiva que permita importar imágenes, realizar la predicción y mostrar los resultados.

2.3. Enfoque y método seguido

Para el desarrollo del proyecto se ha escogido una base de datos de imágenes de radiografías de tórax del repositorio Kaggle. En dicha base de datos las imágenes se clasifican en cuatro clases: normal, neumonía covid-19, neumonía viral y opacidad torácica.

Los métodos de aprendizaje profundo se basan en la construcción de redes neuronales grandes y complejas, utilizados generalmente cuando se trabaja con conjuntos muy grandes de datos analógicos etiquetados como imágenes, texto, audio o video [14]. El proceso de aprendizaje se llama profundo porque las redes neuronales artificiales están formadas por varias capas de entrada, salida y ocultas. Gracias a esta estructura se permite que la red aprenda a través de su propio procesamiento de datos.

Para realizar el trabajo se ha escogido métodos de aprendizaje profundo debido a que trabajamos con imágenes y estas tienen gran cantidad de características. Por ejemplo, una imagen de 800 x 1000 píxeles en color puede llegar a tener 2,4 millones de características.

Los principales algoritmos de aprendizaje profundo son [14]:

- Red neuronal convolucional (CNN)
- Redes neuronales recurrentes (RNN)
- Redes de memoria a corto plazo (LSTM)
- Codificadores automáticos apilados
- Máquina de Boltzmann profunda (DBM)
- Redes de creencias profundas (DBN)

Para el reconocimiento de imágenes el algoritmo que se ha considerado más apropiado es el de red neuronal convolucional (CNN). En dicha red las capas se organizan en tres dimensiones: ancho, alto y profundo. Su funcionamiento imita al córtex visual del ojo humano identificando características en las “entradas” lo que permite identificar objetos y por tanto “ver”. Las capas de la CNN presentan una jerarquía lo que implica que se van “especializando”, es decir, según llegamos a capas más profundas se reconocen formas más complejas [15].

En cuanto al desarrollo del algoritmo, hemos decidido la utilización del lenguaje de programación Python debido a la mayor rapidez en el manejo de grandes conjuntos de datos en comparación con R y por las sofisticadas herramientas que presenta para la manipulación de imágenes.

A continuación se detalla la metodología utilizada:

- **Exploración los datos.**
  Se ha realizado un análisis exploratorio de los datos con el fin de comprender mejor los mismos evaluando el número de imágenes de cada clase, el tamaño, el modo, la forma…
- **Pre-procesamiento de los datos.**
  La red toma como entrada los píxeles de cada imagen. Antes de alimentar la red es necesario realizar unas transformaciones a las imágenes:

    - Transformar la imagen en una matriz de píxeles.
    - Normalizar los valores. Los colores de los píxeles tienen valores en un rango de 0 a 255, por lo que se realizará una transformación de cada pixel de manera que quede un valor entre 0 y 1 (valor / 255).
    - Unificar el tamaño de las imágenes. Se redimensionan las fotografías para que todas tengan el mismo tamaño de píxeles.
- **Construir el modelo.**
  Creación de la red neuronal convolucional que está formada principalmente por tres tipos de capas: capas de convolución, capas de reducción y capas completamente conectadas [16].

  La arquitectura de la red alterna capas convolucionales y de reducción hasta el final donde se presentan las capas de conexión total conocidas como una red perceptrón multicapa [17]. La última capa de esta red es una capa clasificadora qué posee tantas neuronas como el número de clases a predecir.
- **Entrenar el modelo.**
  Se utiliza el 80% de las imágenes para el entrenamiento.
- **Validar el modelo.**
  Una vez hemos desarrollado el modelo, utilizamos el conjunto de datos de prueba para evaluar la precisión de la clasificación.
- **Mejorar el modelo.**
  Además de probar con distintos parámetros de entrenamiento, se ha intentado mejorar el modelo utilizando aumento de datos, transferencia de aprendizaje y ajuste fino.
- **Validar el modelo.**
  Finalmente se comparan todos los modelos obtenidos.

Para el desarrollo de la aplicación web existen frameworks escritos en Python, siendo los más populares, Flask y Django. Se ha utilizado Flask ya que no se tenía experiencia en la creación de aplicaciones web en lenguaje Python y Flask destaca por su facilidad de uso, además de ser muy utilizado en problemas de aprendizaje profundo.

Con el objetivo de que la aplicación web pueda ser utilizada desde cualquier dispositivo se ha publicado la aplicación en la plataforma PythonAnywhere.

2.4. Planificación del trabajo

2.4.1. Tareas

A continuación se enumeran las tareas a gran escala que surgen de cada uno de los objetivos específicos.

“Crear una red neuronal convolucional (CNN) que identifique patrones característicos de Covid-19 en radiografías de tórax diferenciándola de la neumonía tipo viral y la opacidad torácica”

- Obtener el conjunto de datos del trabajo y realizar un análisis exploratorio.
- Realizar un pre-procesamiento adecuado a los datos.
- Diseño de la arquitectura de la CNN.

“Obtener un nivel de precisión superior al 70% en el diagnóstico de neumonía por SARS-CoV-2.”

- Entrenamiento del modelo.
- Analizar el rendimiento del modelo evaluando su precisión.
- Intentar mejorar el rendimiento del modelo.

“Desarrollar una aplicación web interactiva que permita importar los datos, realizar la clasificación y mostrar los resultados.”

- Comprender el desarrollo de aplicaciones web utilizando Flask.
- Definir la estructura de la aplicación web.
- Desarrollo de la aplicación web, comprobando su buen funcionamiento.

2.4.2. Calendario

En el siguiente apartado se muestra la planificación temporal de las tareas y subtareas. Para ello, se ha utilizado el programa GanttProject.

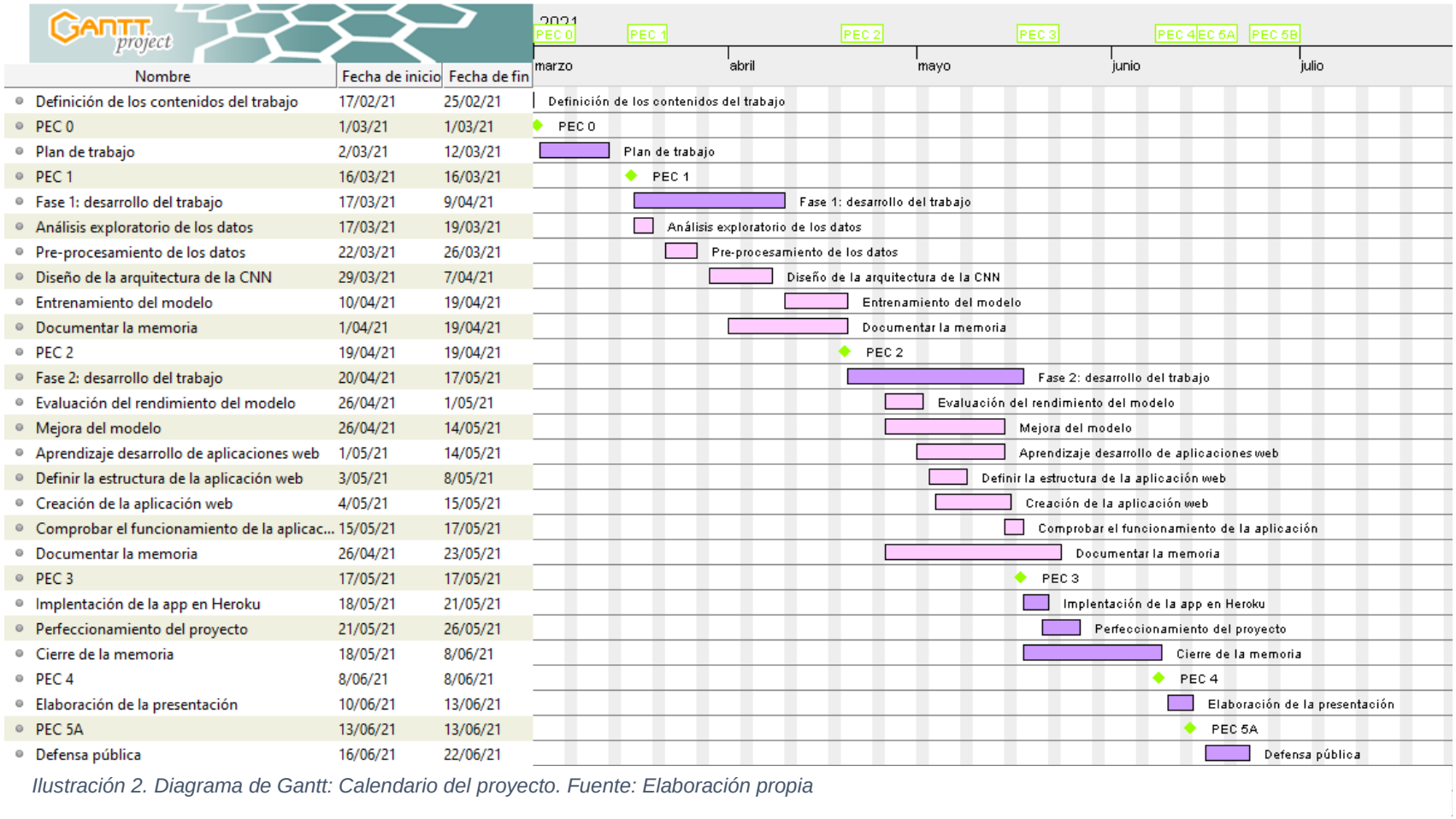

Ilustración 2. Diagrama de Gantt: Calendario del proyecto. Fuente: Elaboración propia

2.4.3. Hitos

A continuación se muestran las fechas claves para la consecución de los objetivos del proyecto.

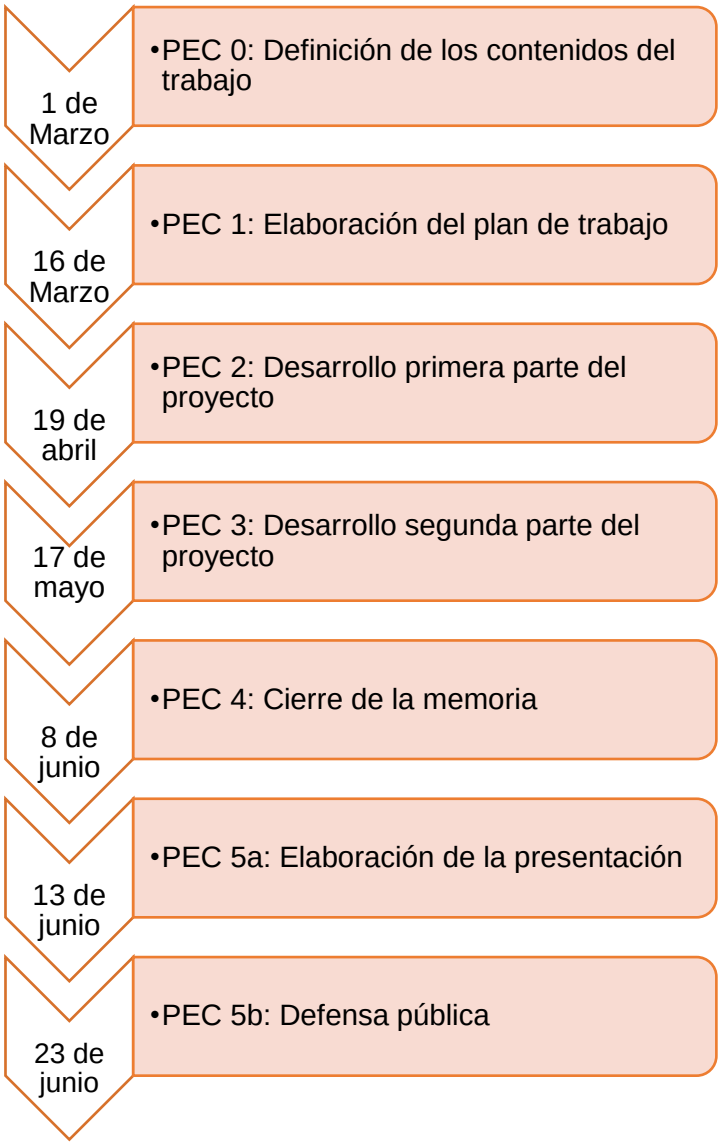

Ilustración 3. Hitos del proyecto. Fuente: elaboración propia

2.4.4. Análisis de riesgos

Descripción del riesgo	Severidad	Probabilidad	Soluciones
Tiempos de ejecución del código extremadamente altos	Alta	Alta	Realizar métodos de optimización
Incapacidad de hacer funcionar las librerías en Python	Media	Media	Buscar librerías alternativas. Explorar ejemplos de código con dichas librerías
Hay imágenes con erratas, no poseen los mismos parámetros o no se pueden procesar	Baja	Alta	Eliminar dichas imágenes del conjunto de datos
El conjunto de datos está desequilibrado o el volumen de datos resulta escaso	Moderada	Moderada	Utilizar técnicas de Data augmentation o Transfer Learning
No se alcanzan los objetivos en cuanto a rendimiento del modelo	Moderada	Moderada	Utilizar modelos pre-entrenados, ajuste fino…
Imposibilidad en la ejecución de la aplicación web con Flask	Moderada	Moderada	Buscar un método alternativo para mostrar los resultados de forma interactiva

Tabla 2. Análisis de riesgos del proyecto. Fuente: elaboración propia.

2.5. Breve sumario de contribuciones y productos obtenidos

A continuación se enumeran los productos obtenidos tras finalizar el proyecto:

- **Memoria.** Reúne toda la información relativa al desarrollo del proyecto, desde la planificación con los objetivos, el contexto y justificación pasando por la metodología utilizada, exponiendo los resultados, mostrando las conclusiones obtenidas, así como la bibliografía utilizada.
- **Repositorio de GitHub** donde se encuentra todo el código que se ha utilizado para el desarrollo del proyecto. Se puede acceder a él a través de este enlace.
- **Aplicación web** capaz de realizar predicciones en imagen subidas por el usuario. Para acceder a la aplicación utilizaremos este enlace.

2.6. Breve descripción de los otros capítulos de la memoria

Los próximos capítulos de la memoria tratarán de:

- *Estado del arte*. Se realiza una comparación de los artículos encontrados en la bibliografía que tienen objetivos similares a los de este proyecto. Además se realizará una exposición de conceptos claves para el desarrollo del trabajo como la inteligencia artificial, el aprendizaje profundo y las redes neuronales convolucionales.
- *Metodología.* En este apartado se explica de forma detallada cómo se ha desarrollado el proyecto, indicando tanto los métodos como los materiales utilizados.
- *Resultados.* Mediante visualización gráfica se mostrarán las métricas de los distintos modelos creados para a continuación realizar una comparación de los mismos. También se mostrará de forma visual el funcionamiento de la aplicación web.
- *Discusión*. Se comentan los resultados obtenidos en nuestro proyecto teniendo en cuenta el contexto del proyecto y la relevancia del mismo.
- *Conclusión.* Exposición de las conclusiones tras finalizar el proyecto. Además se indican las posibles líneas futuras de trabajo.

3 Estado del arte

3.1. Artículos relacionados

Se ha realizado una recopilación de artículos que tienen como objetivo la utilización de técnicas de aprendizaje profundo para la detección de Covid-19. Las tablas que se muestra a continuación expone algunas características importantes de dichos artículos.

Trabajo	Imágenes	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
1 [18]	Radiografia de Torax y TAC. Total de 7390 imágenes.	Crean su propia red desde cero a la que llaman CoroDet	CNN de 22 capas. Activación sigmoidea en capas de convolución y ReLu en capa de agrupación.	Utilizan diferente número de clases: - 2 : Covid o Normal - 3: Covid, Normal o Neumonia bacteriana - 4: Covid, Normal, Neuomía bacteriana o vírica	El mejor modelo se obtiene con la clasificación de dos clases. Obtiene una precisión del 99.1% para dos clases, 94.2 para tres clases y 91.2% para cuatro clases.	Servidor Ubuntu a través de Google Colaboratory. Justificación empírica de modelo con 22 capas.
2 [19]	Radiografía de tórax Total de 13800 imágenes (13617 normales y 183 covid)	Utilización de modelos R-CNN con técnica de validación cruzada de 10 pliegues.	Utilización de la capa conv5_3 de VGG-16 como mapa de características Tamaño de lote = 8 Épocas = 100	Clasificación binaria: - Covid. - Normal	Precisión media obtenida del 97.36%	Trabajo ejecutado en la GPU de Google Colaboratory Utilización de Keras y Tensrflow
3 [20]	Radiografía de tórax 100 imágenes (50 Covid y 50 normal)	Comparación redes prentrenadas: AlexNet, VGG16, VGG19, Google Net, ResNet18, ResNet50, ResNet101, InceptionV3, InceptionResNetV2, DenseNet201 y XceptionNet	Todas las imágenes se rescalan a 280x280 píxeles	Clasificación binaria: - Covid. - Normal.	La mejor precisión se obtiene con ResNet101 con un 98.5%, mientras que la precisión más baja se obtiene con ResNet18 con un 88.64%.	

Tabla 3. Características artículos relacionados. Fuente: Elaboración propia

Trabajo	Imágenes	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
4 [21]	Radiografías de tórax 305 imágenes de cada clase	Creación de una CNN llamada CovXnet	Red compuesta por dos parte: parte no entrenable (transfer learning) y parte entrenable (fine tuning)	Compara diferentes clases: - Normal/ Covid-19 - Neumonía viral/ Covid-19 - Neumonía bacteriana/ Covid-19 - Covid-19/ viral/bacteriana - Covid-19/ Viral/ Bacteriana/ Normal	La mayor precisión se obtiene utilizando la clasificación binaria Covid/ Normal con un valor de 97.4%. Mientras que la precisión más baja se obtiene para la pareja Covid-19/ Neumonía viral con un valor del 87.3%	Proyecto desarrollado en CPU de 48GB de RAM. Las capas convolucionales preentrenadas están entrenadas en imágenes no Covid.
5 [22]	Radiografías de tórax Total de 13.975 imágenes	Red neuronal desde cero llamada COVID-Net	Optimizador Adam Épocas = 22 Tamaño de lote = 64	Tres clases: - Normal. - Neumonía. - Covid-19.	Logra una precisión del 93.3%.	Utilización de keras. Complejidad computacional
6 [23]	Radiografías de tórax. 224 imágenes Covid-19, 700 de neumonía bacteriana y 504 normal	Comparación de redes preentrenadas	Utilización de VGG19, MobileNet v2, Inception, Xception, Inception ResNet v2	Tres clases: - Covid-19. - Neumonía no Covid. - Normal.	La mayor precisión se obtiene utilizando VGG19 con un 98.75% y la menor con Inception ResNet v2 con un 84.38%	
7 [24]	Radiografías de tórax. 341 imágenes Covid, 2800 normales, 2772 neumonía viral y 1493 bacteriana	Comparación de redes preentrenadas	Utilizacion de ResNet50, ResNet101, ResNet152, Inception V3, Inception ResNet V2	Tres clases binarias: - Normal/ Covid-19. - Covid 19/ Neumonía viral - Covid19 / Neumonía bacteriana	Utilizando ResNet 50 se obtiene la mayor precisión (96.1%) y con ResNet153 la menor (93.9%)	Servidor Linux de Google Colaboratory con Ubuntu 16.04. División de los conjuntos en 80% para prueba y 20% para test Validación cruzada con k-fold

Tabla 4. Continuación: Características artículos relacionados. Fuente: Elaboración propia

Trabajo	Imágenes	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
8 [25]	Sonidos respiratorios e imágenes de radiografía de tórax	Creación de CovScanNet con redes preentrenadas	Utilización de Función de activación ReLu para capas convolucionales Combinación de un perceptrón multicapa (MLP) e Inception-v3	Clasificación binaria: - Normal. - Covid-19.	Obtiene una precisión del 80%	Desarrolla una aplicación móvil llamada Ai-CovScan para evaluar de forma individual la presencia de enfermedad.
9 [26]	Radiografías de torax 219389 imágenes de Covid y 24078 normal	Utilización de la red prentrenada AlexNet con modificaciones	Modificación de las dos primeras capas de AlexNet Función activación ReLu para las capas de convolución Tamaño de lote = 100	Clasificación binaria: - Normal. - Covid-19.	Precisión obtenida del 96.5%.	Utilización de GPU Intel i7 NVIDIA 16GB.
10 [27]	Imágenes de TAC de tórax 600 imágenes Covid, 600 de neumonía y 600 normales	Red neuronal desde cero formada por 14 capas	1 capa de entrada, 12 capas ocultas y 1 capa de salida.	Tres clases: - Covid-19 - Normal - Neumonía	Presenta una precisión del 97.802%.	Utilización de Google Colaboratory Entorno de trabajo Python GPU

Tabla 5. Continuación: Características artículos relacionados. Fuente: Elaboración propia.

Observamos como la gran mayoría de los artículos prefieren la utilización de imágenes de radiografías de tórax en lugar de imágenes de TAC. Esto se debe a que, como comentan algunos autores, las imágenes de radiografías de tórax son más accesibles, no necesitan preparación, y el manejo es más fácil.

Otros modelos, como el de Kabid Hassan, S. et al.[18], utilizan ambas, tanto imágenes de radiografías como imágenes de TAC de tórax.

Por otra parte podríamos decir que tenemos dos grupos de artículos, los que realizan únicamente una clasificación binaria Covid-19/Normal, y los que incluyen otras opciones. Estos últimos juegan con las opciones de clasificación, aumentando o disminuyendo el número de clases y comparando los resultados obtenidos.

En nuestro proyecto se ha realizado una clasificación con cuatro clases. Dicha clasificación, a diferencia de algunos artículos de la bibliografía como en el artículo de Wang, L., et al.[21] se ha mantenido constante para todas las técnicas utilizadas. La clase “opacidad torácica” incluida en nuestra clasificación no se ha utilizado en ningún artículo de la bibliografía encontrada.

Siguiendo con la clasificación, entre los que utilizan técnicas multiclase, podemos destacar que hay autores como Kabid Hassan, S. et al.[18] y Wang, L., et al.[21] que hacen una diferenciación entre neumonía viral y neumonía bacteriana, mientras que otros [22] [23] [27], no especifican qué tipo de neumonía no Covid-19 se está estudiando.

En cuanto a la metodología utilizada destacan artículos como el de Apostolopoulos, ID., et al.[22] que crea una red neuronal convolucional desde cero, a la que llama Covid-Net. Al igual que en nuestro trabajo, utiliza la función de optimización Adam y el tamaño de lote 64, sin embargo el número de capas utilizadas en su red es mucho superior, formada por 22.

Otros trabajos [21] [25] crean una red neuronal convolucional con redes pre-entrenadas. Al utilizar una red pre-entrenada, sus redes están formadas por una parte entrenable y otra no entrenable o congelada. En dichos trabajos se modifican algunas de las capas superiores y/o inferiores del modelo pre-entrenado.

Por otro lado, artículos como el de Mahmud, T., et al.[20] y Sait, U., et al.[24] utilizan diferentes modelos de redes pre-entrenadas y comparan los valores de precisión, sensibilidad o especificidad obtenidos con sus datos. Los resultados

de dichos trabajos son diversos, ya que al utilizar diferentes modelos de redes pre-entrenadas no se pueden comparar. Por ejemplo, en el artículo de Sait, U., et al.[24] , la mayor precisión se obtiene utilizando la red ResNet50 mientras que para Narin, A.[23] la mayor precisión se obtiene con la utilización de VGG19, sin embargo Narin, A.[23] no incluía en su modelo la red ResNet y su vez Sait, U., et al.[24] no incluía en el análisis VGG19.

Otro aspecto a destacar es que muchos artículos [19] [24] coinciden en la utilización de Google Colaboratory para el desarrollo de su modelo. Solamente se ha encontrado un artículo [21] que utiliza un procesador CPU utilizando en consideración un número total de imágenes muy inferior a los que usan procesadores GPU. Por ejemplo, podríamos comparar el número de imágenes de Wang, L., et al.[21] que no llega a 1300 imágenes con las utilizadas por Dhiman, G., et al[19] que asciende casi a las 14000 imágenes o el de Perilliat García, I.,et al[26] que supera las 45000 imágenes.

El artículo de Cortés Pérez, E.[25] destaca por la utilización de, además de imágenes de radiografías de tórax, sonidos respiratorios de pacientes covid-19 positivos. Concretamente utiliza sonidos de la tos de los pacientes con diagnóstico de Covid-19. Para analizar estos sonidos se utilizan anomalías respiratorias como ruidos crepitantes o silbilancias ya que la enfermedad por Covid-19 se presenta mayoritariamente con neumonía y asma, ambas muy relacionadas con dichos ruidos pulmonares. El artículo pretende dar una respuesta a las personas que tengan dudas acerca de si padecen o no covid-19. Para ello, Cortés Pérez, E.[25] ha diseñado una aplicación en la que mediante técnicas de aprendizaje profundo que utilizan tres tipos de datos, imágenes de radiografías de tórax, sonidos de tos y prueba rápida de antígeno, el individuo obtenga un autodiagnóstico de que tan probable es que esté padeciendo la enfermedad. Dicho algoritmo se ha implementado en forma de aplicación para teléfonos inteligentes dotada con el nombre de Ai-Cov-Scan.

En la misma línea que el artículo de Cortés Pérez, E.[25] es importante destacar también la labor de Cecilia Mascolo, profesora de la universidad de Cambridge, que junto con un grupo de científicos ha presentado una aplicación para teléfonos móviles que determina si la persona padece la Covid-19 en función de su forma de hablar, respirar y toser. Mascolo hace hincapié en que tras consultar con médicos uno de los rasgos más comunes de los pacientes que presentan el virus es *"la forma de recuperar el aliento al hablar, así como la tos seca y los intervalos de sus patrones respiratorios"*[28]. Sin embargo, uno de los impedimentos a los que se tuvo que enfrentar Mascolo fue la escasa existencia de bases de datos de sonidos respiratorios. Es por ello que también, a través de la aplicación, insiste en que las personas graben muestras de voz, tos y

respiración informando de sus síntomas para aumentar así la base de datos y mejorar las predicciones [29]

En el artículo de Márquez Díaz, J.[29] aunque no se realizan técnicas de aprendizaje profundo de forma directa, se expone la importancia que tiene dicho aprendizaje en el manejo de la pandemia. En dicho artículo también se mencionan otras formas de utilizar las redes neuronales convolucionales para combatir el covid-19. Por ejemplo, una de ellas utiliza aprendizaje profundo para el reconocimiento de rostros con el objetivo de detectar si las personas están cumpliendo la normativa de llevar puesta la mascarilla.

En dicho artículo [29] aparte de considerar la amplia gama de ventajas que supone la implementación de técnicas de inteligencia artificial, aprendizaje profundo o big data, también propone una reflexión acerca de los aspectos éticos ya que en la mayoría de los casos se está tratando con datos, ya sean imágenes u otra medición, de víctimas del Covid-19.

En cuanto a las aplicaciones web para la detección de Covid-19 encontramos en la bibliografía la herramienta creada por Grillo y Balbas [30] a la cual se puede acceder a través del enlace coronavirusxray.com. Para realizar la aplicación, Gillo y Balbas han entrenado un modelo de redes neuronales convolucionales con cuatro tipos de radiografías: neumonía Covid-19, neumonía vírica, neumonía bacteriana e imágenes fallidas. Con su modelo obtienen una precisión de hasta el 97.9%

3.2. INTELIGENCIA ARTIFICIAL

La primera vez que se utilizó el término de inteligencia artificial fue en el verano de 1956, durante la Conferencia Dartmouth. Dicho término fue adoptado por unos investigadores interesados en temas que posteriormente serían la base de la inteligencia artificial. Así surgieron los primeros trabajos de este campo. [31]

Sin embargo, a pesar de los progresos conseguidos después de la reunión de Dartmouth, el éxito de la inteligencia artificial se paralizó a mitad de los sesenta debido a la limitación del funcionamiento de los sistemas implementados a dominios limitados (conocidos como micro-mundos). La adaptación a entornos reales no fue fácil [32].

En los años ochenta se construyeron los primeros sistemas comerciales. A partir de la década de los noventa, considerada la edad de oro de la inteligencia artificial, numerosas empresas deciden apostar por la tecnología de la

inteligencia artificial impulsados por la necesidad de manejar gran cantidad de datos y mejorar la capacidad de procesamiento [33]

El punto álgido de la inteligencia artificial se sitúa en 1997 cuando IBM lanzó un ordenador, llamado Deep Blue, que logró ganar a Gari Kaspárov, quien fue campeón del mundo de ajedrez. Este hecho dio pie a que la inteligencia artificial fuese la protagonista de películas acerca del futuro tecnológico [33].

Posteriormente ocurre otro suceso parecido, IBM desarrolla otro ordenador, conocido como Watson que se proclama como vencedor del gran conocido concurso de preguntas y respuestas 'Jeopardy' [33].

Desde entonces, IMB y Microsoft no dejaron de invertir enormes cantidades de dinero en este ámbito a fin de líder la inteligencia artificial. Actualmente la inteligencia artificial se encuentra asentada en nuestras vidas [33].

Aunque no existe una única definición de inteligencia artificial podríamos decir que es "*la capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano*"[34]. Es decir, la inteligencia artificial otorga a los ordenadores dos importantes capacidades: el aprendizaje y la toma de decisiones.

Aunque la inteligencia artificial se puede implementar prácticamente en la mayoría de las situaciones podemos destacar los siguientes ámbitos en los que está teniendo un crecimiento en los últimos años:

- Reconocimiento, clasificación y etiquetado de imágenes.
- Implementación de estrategias comerciales.
- Mejora de la atención médica mediante el procesamiento de datos de pacientes.
- Mantenimiento predictivo en el sector industrial.
- Asignación de contenido en redes sociales.
- Herramienta de seguridad cibernética.

En definitiva, la IA tiene un gran impacto en la vida de las personas especialmente en las áreas de la salud, educación, trabajo, relaciones sociales y bienestar.

3.3. APRENDIZAJE AUTOMÁTICO

Uno de los principales enfoques de la inteligencia artificial es el aprendizaje automático. El término aprendizaje automático lo acuñó Arthur Samuel en 1959 [35]

En muchas ocasiones se usan los términos de inteligencia artificial y aprendizaje automático indistintamente, sin embargo, se trata de dos términos diferentes. Como se ha explicado anteriormente, la IA según el Instituto de Ingeniería del Conocimiento[35] se refiere al *"concepto general de máquina pensante o la toma de decisiones automatizada"*[35], mientras que Arthur Samuel describió el aprendizaje automático como la capacidad de las computadoras de aprender sin estar programadas explícitamente para ello.

La idea principal del aprendizaje automático es crear algoritmos que a partir de unos datos de entrada sean capaces, mediante análisis informáticos, de predecir valores dentro de un rango aceptable de precisión, identificar en los datos patrones y tendencias y finalmente aprender de la experiencia anterior [35].

Podemos hablar de tres tipos de aprendizaje automático: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje reforzado. A continuación se detalla una breve descripción de cada uno.

Ilustración 4. Tipos de aprendizaje automático. Fuente: [34]

Aprendizaje supervisado

El aprendizaje supervisado es aquel que utiliza datos previamente etiquetados u organizados para establecer cómo debería clasificarse la nueva información. Es por ello, que el aprendizaje supervisado requiere la intervención humana [34].

Aprendizaje no supervisado

El aprendizaje no supervisado no utiliza datos previamente etiquetados sino que encuentra por sí solo la forma de clasificar los datos. Por tanto, los algoritmos no supervisados no precisan la intervención humana [34].

El propio sistema se encarga de observar las características y/o comportamientos de los datos para después encontrar similitudes y patrones que le permitan agrupar los mismos o detectar patrones de interés [36].

Aprendizaje por refuerzo

El aprendizaje por refuerzo consiste en el aprendizaje por "experiencia", es decir, cada vez que el algoritmo acierta se le da un refuerzo positivo [34].

3.4. APRENDIZAJE PROFUNDO

El aprendizaje profundo o Deep Learning (DL) es un subconjunto del aprendizaje automático que se caracteriza por la utilización de redes neuronales profundas, es decir, con muchas capas. El aprendizaje profundo está inspirado en el funcionamiento de un cerebro humano, así el diseño de la red neuronal está basado también en las redes neuronales humanas.

Mediante la tarea de aprendizaje, las redes neuronales son capaces de identificar patrones y clasificar datos de manera similar a como lo haría el cerebro humano.

De manera muy simplificada las redes neuronales artificiales imitan funciones básicas de las redes neuronales biológicas.

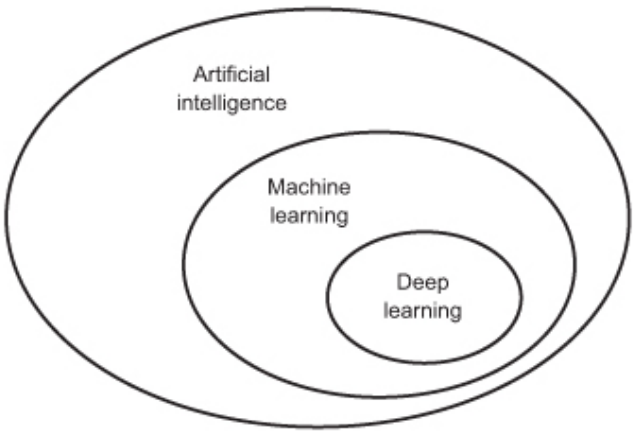

Ilustración 5. Inteligencia artificial, subcampos. Fuente: [41]

Red neuronal biológica

La estructura típica de una neurona es un cuerpo celular o soma, las dendritas y el axón.

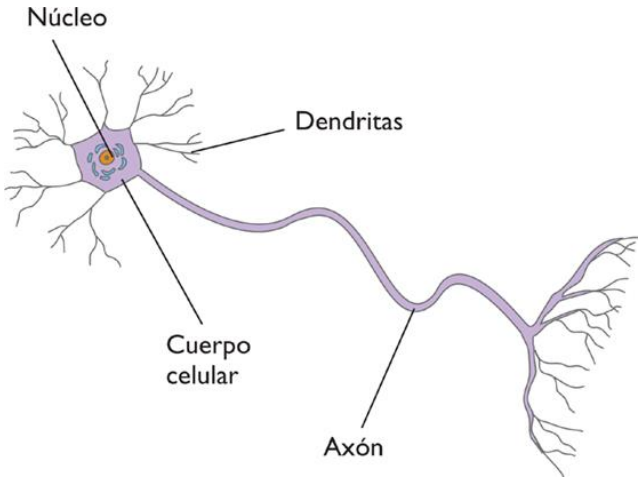

Ilustración 6. Estructura de una neurona. Fuente: 38

De forma general, las neuronas reciben las señales a través de las dendritas y emiten señales a través del axón. Así la sinapsis es el proceso por el cual la señal es transferida desde el axón de una neurona a las dendritas de otra [37]

La señal se transmite en forma de impulso nervioso que puede ser eléctrico o químico. Al llegar el impulso nervioso al axón se liberan unas sustancias denominadas neurotransmisores al espacio entre las neuronas, llamado espacio sináptico. Esta señal química es recepcionada por la dendrita de la neurona contigua, desencadenándose un impulso eléctrico [38].

Las neuronas presentan una determinada carga eléctrica en reposo que difiere de la carga que hay en el exterior de la neurona. Normalmente se encuentra alrededor de los -70mV. A esta carga se la denomina potencial de reposo.

Cuando una neurona recibe una estimulación superior a un umbral se genera el potencial de acción de manera que la neurona se despolarizará generando un impulso nervioso que viajará hasta el axón permitiendo la liberación de neurotransmisores. Dichos neurotransmisores estimulan los receptores de la neurona contigua o postsináptica y producen cambios en el potencial de la neurona postsináptica. Los cambios pueden ser excitatorios o inhibitorios. Los excitatorios facilitan la despolarización necesaria para llegar al umbral y que se genere el potencial de acción, es decir, hacen que la membrana se vuelva más positiva, mientras que los inhibitorios hacen que la distancia hasta llegar al potencial de acción sea mayor, es decir, la hacen más negativa [39].

Red neuronal artificial

La red neuronal artificial está formada por un conjunto de nodos conectados a los que llamaremos neuronas. Las conexiones entre neuronas se establecen a través de pesos que son valores numéricos.

El modelo de una única neurona lo podemos observar en la siguiente imagen:

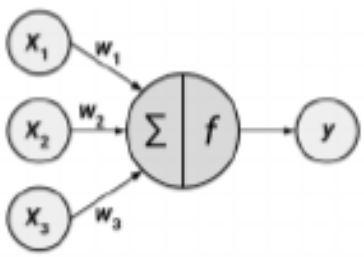

Ilustración 7. Red neuronal artificial sencilla. Fuente: [41]

En la imagen se observa una red sencilla formada por una sola neurona en la que a través de una serie de señales recepcionadas por las dendritas (variables x) se produce la señal de salida (variable y). Cada una de las señales es ponderada (valores de w) en función de su significancia. La neurona suma dichas ponderaciones y finalmente emite la señal a través de una función de activación f.

La siguiente función representa el funcionamiento de una neurona artificial con n dendritas:

= =

$$y(x) = f\left(\sum_{i=1}^{n} w_i x_i\right)$$

Ilustración 8. Función de una neurona. Fuente: [42]

Las redes neuronales tienen las siguientes características:

- Función de activación. Función que transforma las señales de entrada en una sola señal de salida.
- Arquitectura o topología de red. Descripción del número de neuronas, número de capas y la forma en la que se conectan.
- Algoritmo de entrenamiento. Establecimiento de pesos para inhibir o excitar neuronas.

Función de activación

La función de activación es aquella que a través del procesamiento de unos datos de entrada devuelve una salida o resultado.

En la Tabla 6 se pueden observar las características de las distintos tipos de función de activación.

Sigmoide	Tangente hiperbólica	ReLU- Rectified Lineal Unit	Leaky ReLU- Rectified Lineal Unit	Softmax
Resultado en escala (0,1)	Resultado en escala (-1 ,1)			Salida en forma de probabilidades
Valores altos tienden a 1 y los bajos a 0	Valores altos tienden a 1 y bajos a -1	Anula los valores negativos y deja los positivos tal cual	Transforma los valores negativos multiplicándolos por un cociente rectificado y los valores positivos se dejan tal cual	El sumatorio de todas las salidas de probabilidades suma 1
Satura y mata el gradiente	Satura y mata el gradiente	Solo se activa si son positivos	Penaliza los negativos con un coeficiente rectificador	Muy diferenciable
Lenta convergencia	Lenta convergencia			
No centrada en 0	Centrada en 0			
Acotada entre 0 y 1	Acotada entre -1 y 1	No está acotada	No está acotada	Acotada entre 0 y 1
Buen rendimiento en la última capa	Buen desempeño en redes recurrentes.	Buen desempeño en redes convolucionales	Buen desempeño en redes convolucionales	Buen rendimiento en las últimas capas
	Se utiliza para predecir entre una opción o la contraria	Se comporta bien con imágenes. Puede morir demasiadas neuronas	Se comporta bien con imágenes. Similar a la función ReLU	Se utiliza para normalizar tipo multiclase

Tabla 6. Tipos de función de activación. Fuente: [43]

Topología de red

La topología de red es la base para el aprendizaje de una red neuronal. La topología de red se refiere a la estructura o arquitectura de las neuronas interconectadas.

Al conjunto de neuronas lo denominamos nodo. En una ANN los nodos se conectan a través de la sinapsis. A su vez los nodos se agrupan formando capas y el conjunto de una o más capas se denomina red neuronal.

Aunque existen múltiples elementos en la arquitectura de una red neuronal, destacamos los siguientes:

- El número de capas. Cuantas más capas tenga la red más compleja se vuelve.
- El número de nodos dentro de cada capa.
- Capacidad de la información para viajar hacia atrás.

Por norma general, cuanto más compleja es la red neuronal mayor capacidad de aprendizaje y mayor complejidad en los límites de decisión. Sin embargo, no solo el tamaño influye en la potencia de la red, muy importante también la forma en la que se organizan las unidades.

Tipos de capas

Podemos establecer tres tipos de capas en las redes neuronales artificiales:

- Capa de entrada. Podríamos decir que es la capa “sensorial” ya que las neuronas que la forman se encargan de recibir los datos.
- Capa de salida. Es la capa de respuesta ya que proporciona la salida final de la red neuronal.
- Capa oculta. Son las capas encargadas de procesar la información y transmitirla a otras capas.

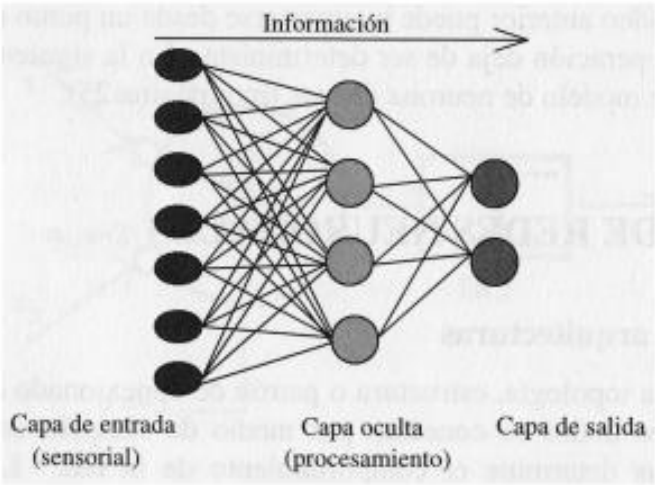

Ilustración 9. Tipos de capas de una red neuronal - Fuente: [44]

Las neuronas se pueden unir de diferentes maneras [44]:

- **Unión todos con todos**: todas las neuronas de una capa se unen con todas las neuronas de la otra capa. Es uno de los tipos de uniones más utilizado.
- **Unión lineal:** cada neurona de una capa se une únicamente con una neurona de la otra capa. Este método es menos utilizado y se suele usar para unir la capa de entrada con la capa de procesamiento.
- **Predeterminado.** En este tipo las redes tienen la capacidad de añadir/eliminar neuronas y/o conexiones en sus capas. Estableciendo un orden en las capas se pueden establecer conexiones hacia delante o feedforward, hacia atrás o feedback y conexiones laterales.

CLASIFICACION DE LAS REDES NEURONALES SEGÚN SU TOPOLOGÍA

Redes monocapa:

Como su propio nombre indica están formadas por una única capa. Las neuronas se unen mediante conexiones laterales (conexiones dentro de la misma capa)

Dentro de las redes monocapa destacan *"la red de Hopfield, la red BRAIN-STATE-IN-A-BOX o memoria asociativa y las maquinas estocásticas de Botzmann y Cauchy"* [44].

Dentro de las redes monocapa, si permiten que las neuronas se conecten con ellas mismas se denominan redes autorecurrentes.

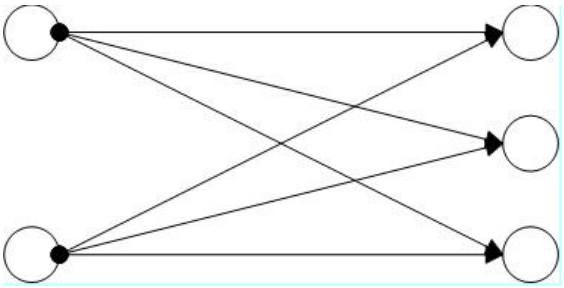

Ilustración 10. Algoritmo entrenamiento redes monocapa. Fuente: [44]

Redes multicapa

Como su propio nombre indica están formadas por carias capas de neuronas. Las podemos clasificar según la dirección en la que se conexionan las neuronas. Cuando reciben la información desde la entrada hasta la salida se denominan conexiones hacia delante o feedforward. Mientras que, cuando aparte de del orden anterior son capaces de recibir información desde la salida hasta la

entrada se llaman conexiones hacia atrás o feedback o retroalimentadas. Las redes con conexiones feedback suelen tener una morfología bicapa [44].

Ejemplos de redes según la dirección de sus conexiones:

- Redes con conexiones feedforward: Perceptrón, Adaline, Madaline, Backpropagation y modelos LQV y TMP de Kohonen.
- Redes con conexiones hacia atrás: ART, Bidirectional Associative Memory (BAM) y Cognitrón.

A continuación se exponen algunas de las redes neuronales más representativas:

PERCEPTRÓN SIMPLE

Fue elaborada por F. Rosenlatt a finales de la década de los cincuenta quien se basó en la regla de aprendizaje de Hebb y en las teorías biológicas de McCulloch y Pitts. El funcionamiento de Perceptrón consiste en asignar a un vector de longitud n un valor binario utilizando una transformación no líneal. Representa la lógica binaria. Su arquitectura está formada por dos capas de neuronas. Las capas de entrada solo admiten valores binarios y las capas de salida también emiten un resultado binario. La forma en la que se conexionan las neuronas es todas con todas.

LA RED DE HOPFIELD

Esta red fue desarrollada por Hopfield en 1982 *"basándose en los modelos de redes de McCulloch y Pitts y los símiles de los campos magnéticos con spin de Amit, Gutfreund, & Sompolinsky"* [44]. Existen dos versiones de la red de Hopfield, la monocapa o la red bicapa de dos capas. En la red bicapa, una capa se encarga de recoger la información y la otra del procesamiento de dicha información. La forma en que se conectan las neuronas en la red bicapa consiste en primero una conexión de neuronas de la primera capa con la neurona correspondiente en la segunda capa y a continuación, en la segunda capa se produce la conexión de todas con todas las neuronas.

Los valores que toma la red pueden ser bipolares (-1,1) o binarios (0,1).

PERCEPTRÓN MULTICAPA

El perceptrón multicapa surge debido a la limitación del perceptrón simple de clasificar datos que no son linealmente independientes. Para ello, el perceptrón multicapa añade capas que transforman la información y la convierten en linealmente independientes. La arquitectura de la red está formada por una capa

de entrada que recibe la información, varias capas ocultas que realizan el procesamiento y una capa de salida. Los valores que toma la red son números reales.

MAPAS DE KOHONEN. REDES NEURONALES AUTORGANIZATIVAS

Se trata de un tipo de redes no supervisadas. Se caracteriza porque en el espacio de salida las neuronas que representan patrones parecidos se organizan juntas. La dimensionalidad del espacio de salida lo establece el propio diseñador del modelo.

La arquitectura de las redes de Kohonen es bicapa y la conexión entre las neuronas es todas con todas. Las neuronas de la capa de salida deben mostrar las coordenadas que posee en el espacio y la proximidad entre neuronas se realiza mediante una regla de proximidad.

3.5. REDES NEURONALES CONVOLUCIONALES

Podemos definir las redes neuronales convolucionales o en inglés, Convolutional Neuronal Network (CNN), como un tipo de red neuronal artificial con aprendizaje supervisado en el que las neuronas actúan como campos receptivos de forma similar a como lo hacen las neuronas de la corteza visual.

Las CNN surgen como una variación del perceptrón multicapa, sin embargo al utilizar matrices bidimensionales las CNN son muy efectivas en tareas de visión artificial [45].

Las CNN están formadas por *"múltiples capas de filtros convolucionales de una o más dimensiones"* [45].

La arquitectura básica de una CNN consta de capas convolucionales donde se realiza la extracción de características, que alternan capas de reducción y finalmente capas de conexión total como las del perceptrón multicapa.

La extracción de características es una fase que se asemeja a la forma de comportamiento de la corteza visual. En dicha fase, a medida que progresan los datos por las capas de convolución se va disminuyendo la dimensionalidad y las capas se van especializando hasta llegar a capas profundas que son capaces de reconocer formas complejas como un rostro.

ESTRUCTURA DE UNA CNN

Están formadas por tres tipos de capas:

- Capas convolucionales.
- Capas de reducción o pooling: reduce el número de parámetros quedándose con los más comunes.
- Capa totalmente conectada clasificadora: proporciona el resultado final.

FUNCIONAMIENTO DE UNA CNN

Las CNN aprenden a reconocer patrones en las imágenes a partir de un entrenamiento con una gran cantidad de muestras. De esta manera son capaces de reconocer *"características únicas de cada objeto y a su vez, poder generalizarlo"*[46].

Píxeles de una imagen

Las imágenes son matrices de píxeles de tamaño variable. Los valores de los píxeles se encuentran en un rango de 0 a 255. La información de entrada de la red neuronal no es una imagen en sí mismo sino sus píxeles. Las dimensiones de la matriz de píxeles de una imagen vienen determinada por el alto, el ancho y el número de canales, siendo por ejemplo 1 si la imagen se encuentra en escala de grises o 3 si la imagen se encuentra a color.

El número de neuronas de la capa de entrada está estrechamente relacionada con el número de píxeles de la imagen, de manera que una imagen con dimensiones 21x21x3 = 1323 tendrá 1323 neuronas en la capa de entrada.

Es importante también normalizar los píxeles antes de que se introduzcan a la red.

Convolución

La convolución distingue a las CNN de cualquier otro tipo de ANN. La convolución es una operación que consiste en recibir unos datos de entrada (matriz de píxeles de una imagen) y aplicarles un filtro o kernel (de menor tamaño que los datos de entrada) que nos proporciona un mapa de características. De esta manera se logra reducir la dimensionalidad [46]. El kernel extrae las características o patrones más importantes de una imagen.

El funcionamiento de los kernels consiste en seleccionar *"grupos de píxeles cercanos"*[45] e ir realizando el producto escalar contra la matriz kernel. El kernel recorre todas las neuronas de entrada. De esta manera se obtiene una nueva

matriz de salida denominada capa de neuronas ocultas. En el caso de las imágenes a color, con tres canales, se tendrán tres kernels que se suman para obtener la matriz de salida.

En una CNN se aplican muchos kernels [47].

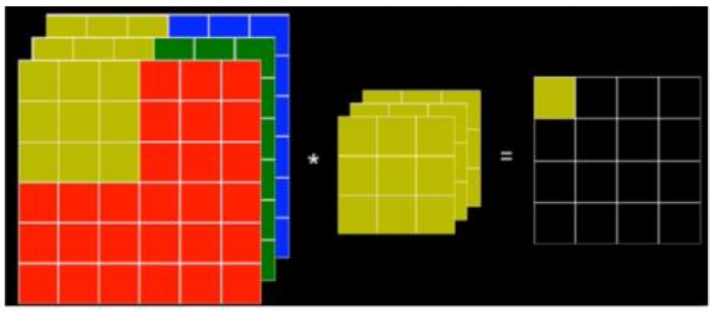

Ilustración 11. Convolución imagen a color. Fuente: [47]

La dimensión de la matriz de salida de una capa de convolución depende de:16

- El número de kernels o filtros.
- Stride: número de píxeles que se desplaza el kernel horizontal y verticalmente.
- Padding: forma de aplicar los filtros en los bordes. Cuando la región no es lo suficientemente grande para realizar convolución hay dos opciones:
    - Ignorar los bordes. Nuestra red no sabrá interpretar la información de los bordes. Recibe el nombre de “Valid”.
    - Agregar ceros en los extremos para conseguir la dimensión necesaria para aplicar la convolución. Se conoce como “Same”.

Tras realizar la convolución con un kernel se aplica una función de activación al mapa de características.

Reducción o pooling o subsampling

Esta capa se suele colocar después de la capa de convolución. El objetivo de la capa de pooling consiste en reducir las dimensiones del volumen de entrada para la siguiente capa convolucional.

Consiste en hacer *“un muestreo preservando las características más importantes que detectó cada filtro”* [45].

Aunque la reducción de tamaño ocasiona pérdida de información esta capa es necesaria ya que disminuye la sobrecarga de la red y reduce el sobreajuste [46].

Hay diversos tipos de subsampling. Los dos principales son [48]:

- Max Pooling: toma el máximo elemento.
- Average Pooling: toma el promedio de los elementos.

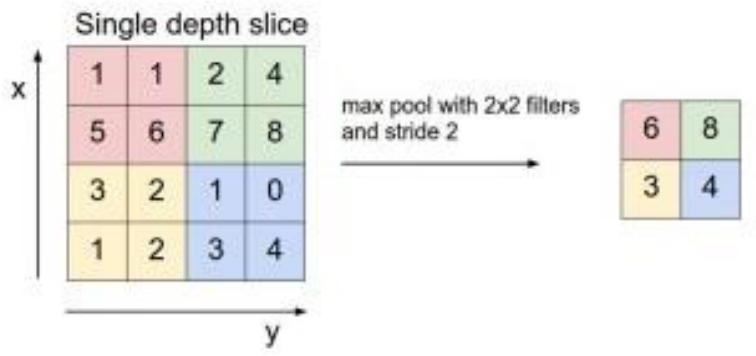

Ilustración 12. Max Pooling. Fuente: 48

Clasificación

Una vez se han aplicado las capas de convolución y pooling necesarias, se utiliza una última capa completamente conectada que proporciona el resultado de la clasificación.

La última capa oculta que ha sido convolucionada y posteriormente reducida tiene una dimensión tridimensional. Para poder entrar en la capa densamente conectada necesitamos “aplanar” esa capa oculta y convertirla en una capa de neuronas “tradicionales”. Tras la aplicación de una función, por ejemplo la llamada función Softmax, obtenemos la capa final de salida.

El número de neuronas de esta capa corresponde con el número de clases que se pretende predecir.

4 Metodología

4.1. Conjunto de datos

El conjunto de datos se ha obtenido a través de Kaggle, una plataforma gratuita que pone a disposición de los usuarios una amplia cantidad de recursos para la ciencia de datos.

La base de datos utilizada recibe el nombre de COVID-19 Radiography Database [49] y ha sido creada por la Universidad de Qatar y la Universidad de Dhaka junto con colaboradores de Pakistán y Malasia y la ayuda de médicos. El propietario de dicho conjunto de datos es Twsifur Rahman y su creación se remonta al 29 de marzo de 2020 recibiendo actualizaciones periódicas, siendo la última el 6 de marzo de 2021.

Las imágenes han sido recopiladas de diferentes fuentes como la Sociedad Italiana de Radiología Médica (SIRM) y han ido incorporándose a este repositorio en diferentes etapas.

El tamaño del conjunto de datos se sitúa en 739.97MB formado por más de 20.000 imágenes pertenecientes a cuatro categorías:

- Imágenes de rayos X de tórax de pacientes positivos en Covid-19
- Imágenes de rayos X de tórax de neumonía viral.
- Imágenes de rayos X de tórax opacidad torácica (infección no Covid-19)
- Imágenes de rayos X de tórax normal.

4.2. Entorno de trabajo

El trabajo se ha realizado en un ordenador personal de marca Lenovo. El procesador del ordenador es un Intel Core i3-4005U CPU con 8 GB de RAM. Cuenta con un sistema operativo de 64bits con procesador basado en x64. El sistema operativo utilizado es Windows 10.

4.3. Lenguaje de programación

El lenguaje de programación elegido para el desarrollo del proyecto ha sido Python. Dicho lenguaje fue presentado por primera en 1991 por Guido Van Rossum convirtiéndose desde entonces en un lenguaje ampliamente utilizado

por la comunidad científica. Entre sus ventajas podemos destacar su *"simplicidad, legibilidad, y sintaxis simple y directa"* [50].

La elección de Python en lugar de otras opciones de programación se debe a que es considerado uno de los lenguajes más apropiados para el procesamiento de imágenes y la implementación de técnicas de aprendizaje automático. Esto se debe a que cuenta con iteraciones rápidas permitiendo concentrar los datos y desarrollar algoritmos. Permite la creación de códigos entendibles con una tasa de aprendizaje rápido lo que resulta muy beneficioso para las técnicas de aprendizaje automático.

Otra ventaja que ofrece la utilización de Python para la IA es que permite combinar librerías, como NumPy y SciPi, aumentando así el rendimiento de los proyectos [51].

La versión de Python utilizada para el desarrollo del proyecto es 3.8.

4.4. Entorno de programación

Para la utilización de Python en un sistema operativo Windows se ha utilizado Anaconda.

Anaconda es una distribución abierta que contiene un conjunto de *"aplicaciones, librerías y conceptos diseñados para el desarrollo de la Ciencia de datos con Python"* [52].

La distribución Anaconda funciona como un gestor de entorno y paquetes y comprende cuatro grupos de soluciones tecnológicas:

- Anaconda Navigator: *"interfaz gráfica de usuario de escritorio que le permite iniciar aplicaciones y administrar fácilmente paquetes, entornos y canales de conda sin usar comandos de línea de comandos"* [53].
- Anaconda Project
- Librerías de Ciencia de Datos
- Conda: interfaz de comandao, en Windows se conoce como Anaconda Prompt.

Se descargó Anaconda Distribution a través de la página oficial de Anaconda [53]. Una vez descargado se utilizó la interfaz gráfica Anaconda Navigator para el desarrollo del proyecto.

Primero se creó un entorno virtual llamado "Tensorflow" con el objetivo de garantizar la estabilidad de nuestro sistema y evitar posibles incompatibilidades entre paquetes.

Una vez creado y activado el entorno virtual Tensorflow se creó un nuevo cuaderno de trabajo a través de la aplicación Jupyter Notebook que permite ejecutar código Python de forma dinámica pudiendo incluir texto, imágenes o gráficos aparte del código. En dicho cuaderno se ha ejecutado todo el código para el desarrollo de los algoritmos de redes neuronales convoluciones.

Para el desarrollo de la aplicación web se utilizó Flask, un marco web de Python que permite crear aplicaciones web de forma sencilla.

Primero instalamos Flask en nuestro entorno de trabajo Tensorflow. Después se creó un archivo llamado app.py donde se ejecutó el código de la aplicación. En dicho archivo se importó Flask además del resto de librerías necesarias.

Para crear el archivo tipo .py se utilizó la aplicación Spyder, también disponible en Anaconda Navigator.

4.5. Librerías utilizadas

- Tensorflow
- Keras
- Os
- Re
- Matplotlib
- Sys
- PIL
- Sklearn
- Pandas

4.6. Importación de datos

Una vez descargado el conjunto de datos de la página web de Kaggle, los datos fueron almacenados en una carpeta de nuestro ordenador llamada “COVID-19_Radiography_Dataset”. Dentro de dicha carpeta se encuentran las imágenes almacenadas en subcarpetas según su clase. El nombre de las subcarpetas, por tanto, es: COVID, Lung_Opacity, Normal y Viral Pneumonia.

Se estableció como directorio de trabajo la ruta de la carpeta principal “COVID-19_Radiography_Dataset”.

Para cargar las imágenes y poder trabajar con ellas se utilizó la librería os con la función os.walk(). A través de dicha función se leyeron todas las imágenes de todas las subcarpetas que forman la carpeta principal.

Para leer las imágenes se utilizó la función load_img() de la librería keras que almacena las imágenes en forma de una instancia PIL. Todas las imágenes fueron guardadas en una lista llamada “imágenes”.

Para obtener la clase de cada imagen se dividió el nombre de cada imagen, por ejemplo “Normal-1604.png” según el símbolo “-“, quedándonos solo con la primera parte y almacenándola también en una lista llamada “clase”.

De esta manera, hemos creado dos listas, una que contiene las imágenes en formato PIL y otra que contiene la clase de cada imagen en forma de string.

4.7. Análisis exploratorio de los datos

El análisis exploratorio de los datos o EDA es uno de los primeros pasos que hay que realizar previo a la utilización de técnicas de aprendizaje automático. El objetivo del EDA es saber más acerca de nuestros datos.

Primero se halló el número exacto de imágenes que forma nuestro conjunto de datos y a modo de ejemplo se visualizó una imagen de forma aleatoria indicando su etiqueta.

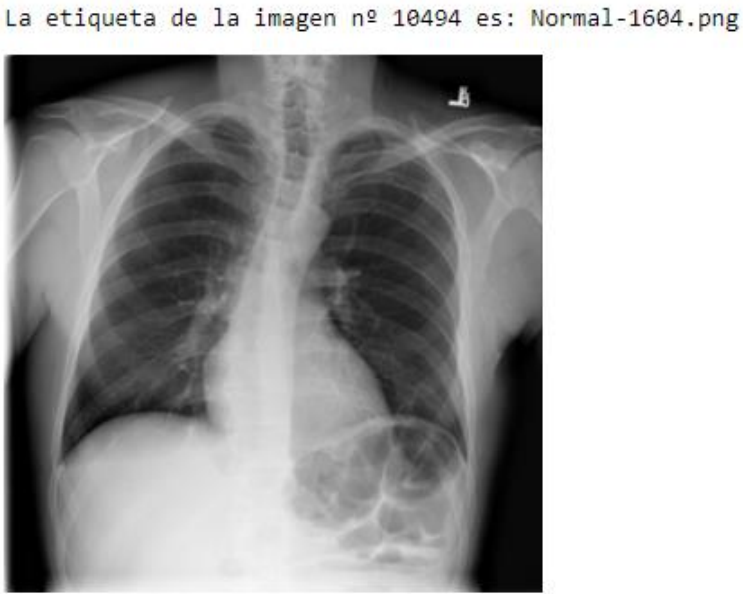

Ilustración 13. Ejemplo imagen Radiografía de tórax. Fuente: elaboración propia.

Para analizar los datos se creó un data frame formado por las variables Tipo (clase de la imagen), Size (tamaño de imagen) y Mode (modo de imagen). El data frame se ha creado con la función pd.DataFrame() de la librería pandas.

Una vez creado el data frame, con el método info() obtenemos un resumen del data frame y las variables que lo forman y con la función print() podemos visualizar las cinco primeras y últimas filas.

Con la función unique() del módulo Numpy corroboramos que solo haya cuatro tipo de clases. También con la función unique() comprobamos que todas las imágenes tengas el mismo tamaño y el mismo modo.

Para saber el número de imágenes de cada clase agrupamos el data frame según la clase con la función groupby.size() de la librería Pandas indicando como argumento la variable Tipo. Dicha función proporciona los recuentos para cada clase. De esta manera se explora cuáles son las clases con mayor y menor tamaño.

Finalmente, continuando con la agrupación por clase, se realiza un diagrama circular con la función plt.pie() de la librería matplotlib. Se obtiene un diagrama circular dividido en secciones según el tamaño de cada clase.

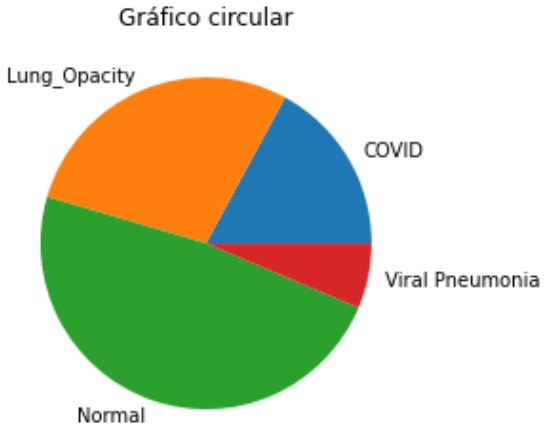

Ilustración 14. Gráfico circular: imágenes agrupadas por clase. Fuente: elaboración propia.

4.8. Reducción del conjunto de datos

El conjunto de datos está formado por un total de 21165 imágenes. Entrenar un modelo de aprendizaje automático con un número tan elevado de imágenes en un ordenador personal de baja potencia es inviable en términos de capacidad computacional. Por ello, se ha realizado una reducción del conjunto datos.

Se ha creado un nuevo directorio llamado “Dataset_reducido” y dentro de él se ha creado una carpeta para cada clase. A continuación, se han seleccionado de forma aleatoria 1300 imágenes de cada clase del directorio original y han sido copiadas en la carpeta de clase correspondiente del nuevo directorio.

4.9. Preprocesamiento de los datos

Antes de que nuestros datos sean introducidos en un modelo es necesario realizar una serie de transformaciones. Para ello, se ha creado una función llamada “preprocesamiento()” que admite imágenes en formato PIL y se encarga de:

- Transformar el modo de imagen a RGB. Si el modo de la imagen es diferente de “RGB” este es transformado para que tenga 3 canales. Dicha transformación es importante para que todas las imágenes tengan el mismo número de canales.
- Transformar las imágenes en un array. Necesitamos que los datos sean válidos para ser utilizados por la red neuronal, la cual no puede leer imágenes directamente, sino que necesita que las imágenes sean transformadas en un vector formado por los píxeles que forman cada imagen. Para ello, utilizamos la función img_to_array() de la libería Keras. Dicha función convierte una instancia PIL, formato en que se encontraban nuestras imágenes, en un vector Numpy.
- Redimensionar las imágenes. El tamaño de las imágenes de nuestro conjunto de datos es de 299 x 299 píxeles por 3 canales (RGB). Dicho tamaño es muy elevado por lo que se realiza una reducción de las dimensiones de las imágenes a 150 x150 con el objetivo de reducir el coste computacional. Para cambiar el tamaño de las imágenes se ha utilizado la función tf.image.resize() de la librería Tensorflow.
- Normalización de los píxeles. Los valores de los píxeles se encuentran entre 0 y 255. Para normalizar los valores dividimos cada píxel entre 255. De esta manera convertimos los datos en una distribución dentro del intervalo de 0 a 1 que nos va a ayudar a que nuestro algoritmo funcione más rápido. Además, la normalización es importante para poner todas las características (píxeles) en la misma escala y que ninguna esté dominada por otra.

Esta función es aplicada a todas las imágenes que usaremos para la creación de los modelos.

Una vez que hemos aplicado la función preprocesamiento() vamos a realizar los siguientes pasos:

Codificación de la variable clase.

La clase de las imágenes la tenemos guardada en forma de lista. Dicha lista contiene la clase en forma de string (“COVID”, “NORMAL”…). Para que el algoritmo supervisado funcione es necesario codificar las clases asignándoles un valor numérico.

Se ha creado un diccionario que asigna a cada clase un número, de manera que la clase Covid se corresponde con el 0, Lung_Opacity con el 1, Normal con el 2 y Viral Pneumonía con el 3. Una vez creado el diccionario utilizamos el método get() para asignar a cada clase su valor numérico iterando la lista que guarda todas nuestras clases. Finalmente, obtenemos una nueva lista con las clases de las imágenes formadas, en lugar de string, por números (0, 1, 2 o 3).

Convertir las listas de imágenes e etiquetas en un vector

Tanto las imágenes (píxeles en formato array) como las etiquetas de cada imagen se encuentran almacenadas en una lista donde cada elemento de la lista se corresponde con cada imagen. El formato de lista no es aceptado para los modelos de CNN por lo que debemos transformar las listas en vectores Numpy. Lo hacemos con la función np.array().

Creación de los conjuntos de entrenamiento, prueba y validación

Dividiremos nuestro conjunto de datos en tres subconjuntos:

- Conjunto de entrenamiento: constituye las imágenes que se usarán para ajustar o entrenar el modelo.
- Conjunto de validación: conjunto de imágenes que se utilizan para *"evaluar el modelo sin sesgo durante el ajuste de hiperparámetros"*[54].
- Conjunto de prueba: imágenes utilizadas para evaluar el modelo final. Dichas imágenes no forman parte del entrenamiento y por tanto nos proporcionan una evaluación imparcial ofreciendo el error real que ha cometido el modelo.

Destinamos el 80% de las imágenes para el entrenamiento o training y el 20% para la prueba o test. De las imágenes destinadas al entrenamiento, el 20% se utilizará para la validación.

Para crear los conjuntos de datos de entrenamiento, prueba y validación se ha utilizado la función train_test_split() de la librería Sklearn.

Finalmente se han creado los tres subconjuntos, de entrenamiento, prueba y validación, formados por 3328, 1040 y 832 imágenes respectivamente.

One-Hot Encoding

Para facilitar el rendimiento y el funcionamiento de nuestra red vamos a realizar One-Hot Encoding de las etiquetas o clases de las imágenes. Como estamos trabajando con categorías o clases que no tienen una relación ordinal, la codificación con números enteros puede provocar un rendimiento o resultado erróneo. Esto se debe a que si dejamos las clases codificadas con números

enteros, estos tienen una relación ordinal que podría ser aprendida por el algoritmo.

La codificación One-Hot consiste en convertir las etiquetas en una clasificación binaria con 0 y 1. Para ello se crea una columna para cada clase y se coloca un uno en la columna que corresponda con la clase y un 0 en las demás.

Para realizar One-Hot Encoding de las etiquetas tanto de las imágenes de entrenamiento, prueba y validación se ha utilizado la función to_categorical() de la librería keras.

4.10. Construcción del modelo de CNN

A continuación se van a ir detallando las características del modelo de red neuronal convolucional creado:

- Para crear un modelo utilizamos la función Sequential() de la librería Keras que agrupa un conjunto lineal de capas. A este modelo secuencial le vamos a ir añadiendo capas. Las capas se añaden utilizando el método add().
- Nuestro modelo consta de cuatro capas de convolución cada una de ellas seguida por una capa de reducción. Las capas convolucionales se añaden utilizando la clase Conv2D() donde especificamos el número de filtros y el tamaño de cada filtro. También especificaremos la función de activación utilizada en las capas convolucionales. Además en la primera capa de convolución indicaremos la forma o shape de nuestros datos, que en nuestro caso particular es 150 x 150 x 3 (hemos reducido el tamaño previamente de las imágenes de 299 x 299 a 250 x 150 píxeles).
- La función de activación elegida para las capas convolucionales es la función ReLu. Se ha elegido esta función ya que es una de las funciones más adecuadas para el procesamiento de imágenes. Además es una buena opción cuando tenemos los datos normalizados y es la función más utilizada en modelos de redes neuronales convolucionales.
- El tamaño del filtro o kernel en todas las capas convolucionales se ha establecido en 3 x 3.
- El número de filtros va aumentando desde la primera hasta la cuarta capa convolucional, siendo los valores de 32, 64, 128 y 512 respectivamente.
- Después de cada capa de convolución añadimos una capa de reducción. El tipo de reducción utilizado es el Max Pooling y la capa se añade mediante la clase MaxPooling2D(). Esta capa "*Reduce la muestra de la entrada a lo largo de sus dimensiones espaciales (alto y ancho) tomando el valor máximo sobre una ventana de entrada (de tamaño definido por pool_size) para cada canal de la entrada.*"[55]. El tamaño establecido para pool_size es 2 x 2.

- Después de las capas de convolución añadimos la capa de aplanamiento. Dicha capa se añade con la clase Flatten() y su función es aplanar las dimensiones de la información de entrada.
- A continuación creamos una capa completamente conectada (capa de neuronas tradicionales) con 512 filtros y función de activación ReLu.
- Finalmente añadimos una capa de salida formada por 4 neuronas, ya que tenemos cuatro clases. La función de activación para esta última capa es la función Softmax. Esta función devuelve un resultado cuyos valores suman 1, es decir, delvuelve la probabilidad de que la entrada pertenezca a cada clase. Es la función indicada cuando realizamos una clasificación multiclase.

Una vez que hemos definido el modelo debemos compilarlo. Esto consiste en asignarle una función de coste, u optimizador y las métricas de rendimiento.

Para compilar el modelo se utiliza el método compile ().

La función de coste determina *"el error entre el valor estimado y el valor real, con el fin de optimizar los parámetros de la red neuronal."*[56]. La función de coste elegida para nuestro modelo es la Entropía cruzada categórica.

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 15, 15, 512)       590336
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 7, 7, 512)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_6 (Dense)              (None, 512)               12845568
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_7 (Dense)              (None, 4)                 2052
=================================================================
Total params: 13,531,204
Trainable params: 13,531,204
Non-trainable params: 0
```

Ilustración 15. Modelo secuencial red neuronal convolucional Fuente: elaboración propia.

4.11. Entrenamiento y validación del modelo

Finalmente el modelo ha sido entrenado con la función fit() con las imágenes y sus respectivas etiquetas del conjunto de datos de entrenamiento.

Se ha establecido batch_size o tamaño de lote de 64. Este valor nos indica el número de imágenes que serán utilizadas para cada actualización de gradiente.

Para entrenar el modelo se han fijado 20 épocas.

Una vez que el modelo se ha entrenado, evaluaremos su precisión mediante la función evaluate() y el conjunto de imágenes y etiquetas de prueba.

4.12. Mejora del modelo

Con el objetivo de intentar aumentar la precisión del modelo se han realizado las siguientes técnicas:

Aumento de datos

Consiste en aumentar las imágenes de entrenamiento mediante la realización de ligeras transformaciones en las imágenes de entrenamiento existentes. Dichas transformaciones son aleatorias y crean imágenes creíbles. De esta manera nuestro modelo tiene más imágenes para el entrenamiento lo que ayuda a generalizar el modelo y reducir el sobreajuste [57].

Para el aumento de datos se han guardado las imágenes en dos carpetas, 1000 imágenes de cada clase se han guardado en la carpeta Train y las otras 300 se han guardado en la carpeta Valid. Dentro de dichas carpetas las imágenes se encuentran guardadas en carpetas según su clase. Se ha realizado dicha metodología para poder utilizar el método flow_from_directory() que se explicará a continuación.

Después se han establecido los generadores de imágenes a partir de la función ImageDataGenerator(). Se ha diseñado un generador para cada conjunto de datos (entrenamiento, prueba y validación). En el único conjunto en el que se va a realizar realmente un aumento de datos es en el de entrenamiento, en el resto se utiliza dicha función únicamente para normalizar los píxeles.

Las transformaciones a las imágenes de entrenamiento (además de la normalización de los píxeles) son las siguientes:

- shear_range = 0.2. Realiza transformaciones de corte de manera aleatoria.
- zoom_range = 0.2. se amplía o aleja la imagen en un range de -20% y +20%.
- horizontal_flip = True. Voltea aleatoriamente la mitad de las imágenes de forma horizontal.

Las transformaciones se aplican a cada conjunto de datos gracias al método flow_from_directory() que accede a las carpetas Train y Valid creadas en nuestro ordenador. La carpeta Valid se ha utilizado tanto para el generador del conjunto de entrenamiento como para el de validación.

Transferencia de aprendizaje

Se va a utilizar un modelo previamente entrenado sobre nuestros datos.

Un modelo previamente entrenado es un modelo que se entrenó con gran cantidad de imágenes de manera que las características aprendidas por dicho modelo pueden ayudarnos en nuestro problema de clasificación aunque las clases sean totalmente diferentes.

El modelo previamente entrenado elegido es la arquitectura VGG16 que fue creada en 2014 por Karen Simonyan y Andrew Zisserman [57]. Este modelo ha sido entrenado con 1.4 millones de imágenes y 10.000 clases distintas.

Se va a utilizar la red VGG16 con aumento y sin aumento de datos. Para ello, se ha realizado una extracción de características que consiste en *"utilizar las representaciones aprendidas por un modelo previamente entrenado para extraer características interesantes de nuevas muestras"* [57]. Para la extracción de características tomamos la parte convolucional de la red VGG16, ejecutamos nuestros datos a través de la red y finalmente entrenamos un clasificador densamente conectado en la parte superior.

Comenzamos importanto el modelo VGG16 mediante la función VGG16() de la librería Keras. Especificaremos el argumente "include_top=False" para que no se incluya el clasificador de la red preentrenada ya que debemos crear nuestro propio clasificador que se ajuste a nuestros datos.

Para extraer las características sin aumento de datos realizamos un preprocesamiento de las imágenes con la función vgg16.preprocess_input() que realiza un escalado de los píxeles y posteriormente utilizamos el método predict() del modelo VGG16 para extraer características de las imágenes de entrenamiento, prueba y validación. Dichas características son almacenadas en matrices Numpy para cada conjunto y tienen una forma de (número de muestras, 9, 9, 512).

Finalmente establecemos nuestro clasificador densamente conectado y entrenamos el modelo utilizando las características extraídas.

Para la extracción de características junto con aumento de datos comenzamos congelando la base convolucional del modelo. Después creamos el modelo secuencial con la base convolucional congelada y estableciendo nuestro propio clasificador densamente conectado. Finalmente entrenamos el modelo aumentando los datos.

Ajuste fino

El ajuste fino de un modelo previamente entrenado consiste en *"descongelar algunas de las capas superiores de una base de modelo congelada utilizada para la extracción de características y entrenar conjuntamente tanto la parte recién agregada del modelo (en este caso, el clasificador completamente conectado) como estas capas superiores"* [40].

Para ello, descongelamos la parte convolucional del modelo VGG16 (previamente la habíamos congelado) y a continuación congelamos todas las capas convolucionales desde la última hasta la cuarta. Finalmente compilamos el modelo y lo volvemos a entrenar utilizando también aumento de datos.

4.13. Diseño de la aplicación web

La aplicación web ha sido creada con Flask, un framework de Python que proporciona herramientas para crear aplicaciones de forma sencilla.

En el entorno virtual de nuestro proyecto, llamado "Tensorflow", instalamos Flask. Una vez instalado Flask, utilizamos Spyder, programa también disponible en Anaconda Navigator, para crear un archivo .py.

Para la visualización del contenido de la aplicación web, Flask permite la utilización de plantillas Jinja con la fución render_template(). Las plantillas Jinja son archivos con código HTML.

Para poder emplear un modelo de red neuronal convolucional y realizar predicciones, debemos cargar el modelo. En nuestro caso, el modelo que hemos utilizado para la aplicación se encontraba guardado en forma de archivo .json. Cargamos el modelo y sus pesos.

Para la creación de la aplicación web primero diseñamos una pantalla principal con la plantilla 'index.html'. En dicha pantalla aparece el título de la aplicación. Además indica al usuario que debe seleccionar una imagen desde su ordenador para poder realizar la predicción.

Una vez que la imagen es seleccionada se realiza un procesamiento de la misma, de forma similar al realizado cuando se llevaron a cabo los modelos:

- Convertir la imagen en una matriz de píxeles.
- Convertir la imagen a modo RG, en caso necesario.
- Redimensionar la imagen, en caso necesario.

- Expandir dimensiones.

Cuando el usuario selecciona una imagen, esta es cargada en la pantalla principal y es entonces cuando se puede utilizar el botón “predict” para realizar la predicción.

Para realizar la predicción únicamente se utiliza el modelo previamente cargado, utilizando el método predict(). Como la predicción de nuestro modelo se muestra en forma de probabilidades, ya que utilizamos la función Softmax, seleccionamos la probabilidad más alta y emitimos un resultado indicando la clase a la que se refiere dicha probabilidad.

4.14. Publicación de la aplicación web

La aplicación web ha sido publicada en la plataforma PythonAnywhere para que se pueda acceder a ella desde cualquier dispositivo.

Primero se creó una cuenta en dicha plataforma y a continuación se creó una aplicación vacía a la que se fueron añadiendo todas las configuraciones y elementos necesarios.

- Se creó un entorno virtual en la consola de la plataforma. En dicho entorno se instalaron todas las librerías necesarias para la aplicación.
- En el entorno virtual creado se clonó nuestro repositorio Github, en el que se encontraban todos los archivos necesarios para el funcionamiento de la aplicación (archivo de código .py y plantillas Templates).
- Una vez que hemos creado el entorno y tenemos descargado los datos en dicho entorno virtual, configuramos la aplicación web para que funcione con dicho entorno y con dichos datos.
- A continuación configuramos el WSGI de nuestra configuración, indicando la ruta en la que se encuentran los archivos en la plataforma.
- Finalmente, ya tenemos la aplicación web publicada.

5 Resultados

5.1. Exposición de los resultados de cada modelo

A continuación se detallan las métricas de precisión y pérdida de cada modelo creado junto con la visualización de dichas métricas durante el entrenamiento.

5.1.1. Modelo general

Modelo formado por cuatro capas de convolución. Ha sido entrenado con 1300 imágenes de cada clase.

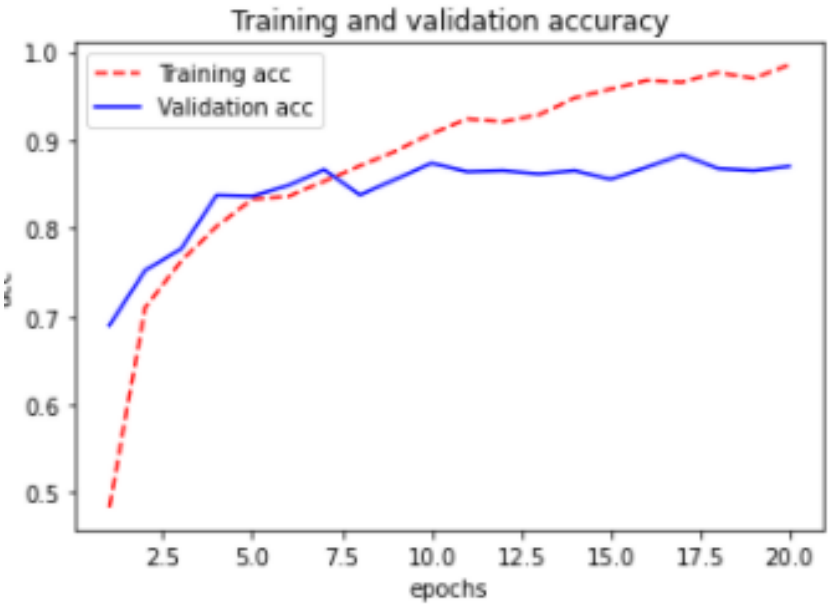

Ilustración 16. Visualización de la precisión durante el entrenamiento y validación del modelo general. Fuente: elaboración propia.

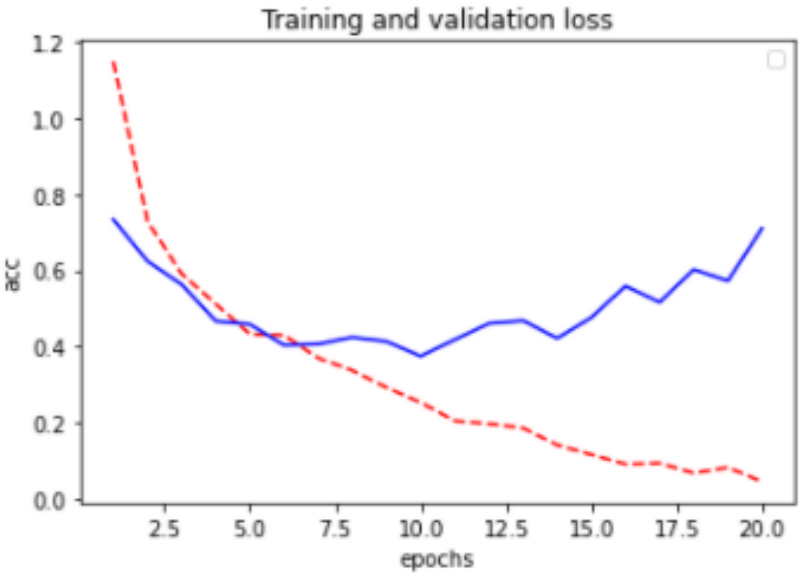

Ilustración 17. Visualización de la función de pérdida durante el entrenamiento y validación del modelo general. Fuente: elaboración propia

Precisión	0.8337
Pérdida	0.7734

Tabla 7. Métricas modelo general. Fuente: elaboración propia.

En los gráficos podemos observar el patrón típico del sobreajuste. Observamos como la precisión del entrenamiento avanza rápidamente llegando casi al 100%, sin embargo, la precisión de la validación alcanza su máximo en las primeras épocas, quedándose “estancada”. Lo mismo ocurre con la pérdida, que durante el entrenamiento disminuye casi a cero mientras que durante la validación alcanza su mínimo en las primeras épocas, a partir de las cuales aumenta la pérdida.

El sobreajuste o también conocido como Overfitting es habitual cuando tenemos pocos datos y nuestro modelo aprende características demasiado específicas de los datos sin tener capacidad de generalizar los resultados ya que solo se consideran válidos los datos que sean prácticamente idénticos a los del entrenamiento.

5.1.2. Modelo con aumento de datos

Este modelo se ha entrenado mediante un generador de imágenes que ha producido un aumento de las mismas únicamente en las imágenes de entrenamiento.

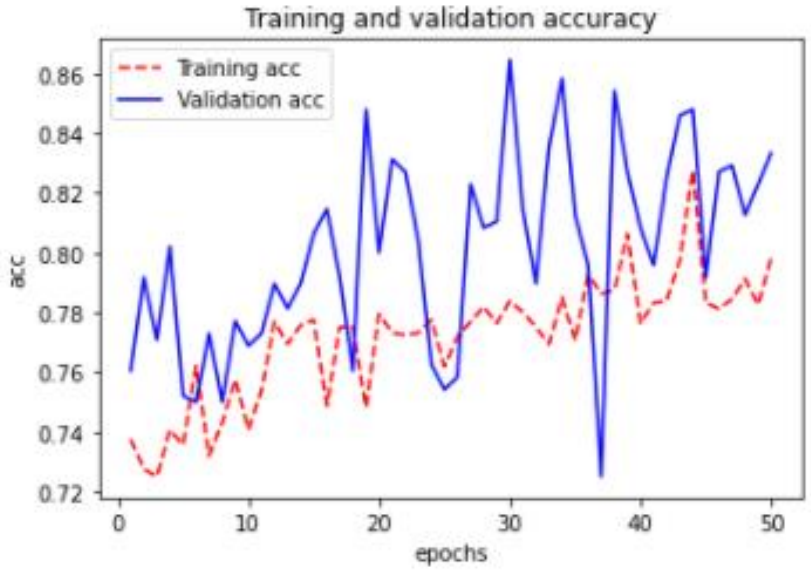

Ilustración 18. Visualización de la precisión durante el entrenamiento y validación del modelo con aumento de datos. Fuente: elaboración propia.

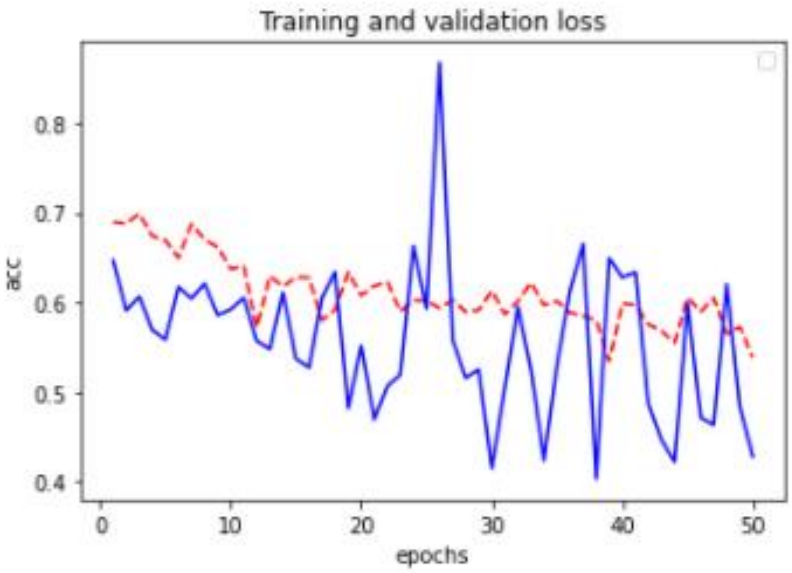

Ilustración 19. Visualización de la función de pérdida durante el entrenamiento y validación del modelo con aumento de datos. Fuente: elaboración propia

En las gráficas podemos observar, tanto en la precisión como en la pérdida, que el entrenamiento y la validación siguen la misma dirección. Cabe destacar que en las gráficas no se observan todavía signos de sobreajuste. Además cabría esperar una mejora de las métricas si se aumentara el número de épocas.
Observamos una mejoría con respecto al modelo anterior.

Precisión	0.8592
Pérdida	0.4289

Tabla 8. Métrica modelo con aumento de datos. Fuente: elaboración propia.

5.1.3. Transferencia de aprendizaje sin aumento de datos

Para la construcción de este modelo se ha realizado una extracción de características de los conjuntos de datos de entrenamiento, prueba y validación. A continuación se ha utilizado la red pre-entrenada VGG16 pero con nuestro clasificador densamente conectado. El modelo ha sido entrenado y evaluado con las características extraídas.

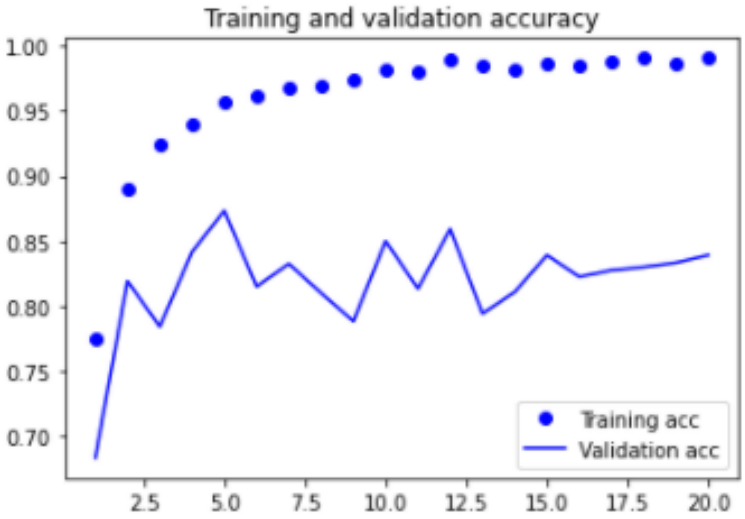

Ilustración 20. Visualización de la precisión durante el entrenamiento y validación del modelo VGG16 sin AD. Fuente: elaboración propia.

Ilustración 21. Visualización de la función de pérdida durante el entrenamiento y validación del modelo VGG16 sin AD Fuente: elaboración propia

En estos gráficos observamos bastante diferencia entre las métricas de entrenamiento y validación. Además podemos comprobar que se produce sobreajuste prácticamente desde el principio.

Precisión	0.8258
Pérdida	99.2840

Tabla 9. Métricas modelo VGG16 sin DA. Fuente: elaboración propia.

La evaluación del modelo muestra una pérdida de 99.28, lo que nos indica que en este modelo tenemos un serio problema de sobreajuste.

5.1.4. Transferencia de aprendizaje y aumento de datos

En este modelo además de utilizar una red pre-entrenada (VGG16) se ha utilizado aumento de datos.

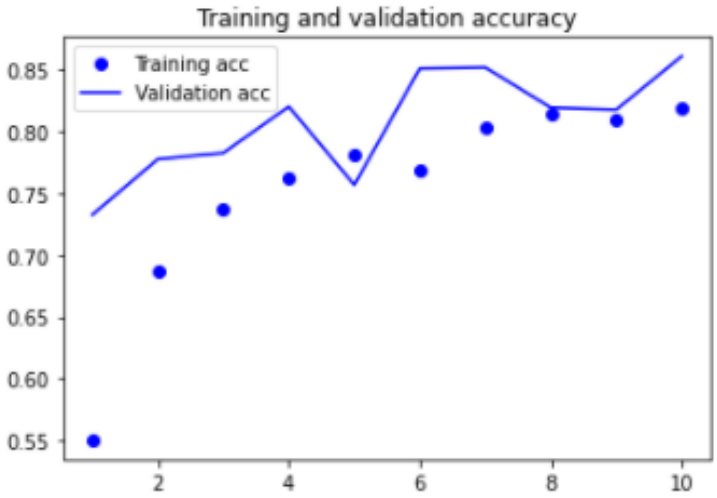

Ilustración 22. Visualización de la precisión durante el entrenamiento y validación del modelo VGG16 con AD. Fuente: elaboración propia.

Ilustración 23. Visualización de la función de pérdida durante el entrenamiento y validación del modelo VGG16 con AD Fuente: elaboración propia.

En los gráficos podemos observar una gran mejoría con respecto al modelo anterior. Observamos como las métricas de entrenamiento y validación van a la

par, siendo casi paralelas. No se observan indicios de sobreajuste y además se obtienen valores de precisión y pérdida bastante razonables. También se intuye de las gráficas que un aumento en las épocas podría suponer una mejora del rendimiento del modelo.

Precisión	0.875
Pérdida	03459

Tabla 10. Métricas modelo VGG16 con AD. Fuente: elaboración propia.

5.1.5. Ajuste fino con aumento de datos

Por último se ha desarrollado un modelo realizando un ajuste fino del modelo pre-entrenado VGG16 junto con aumento de datos.

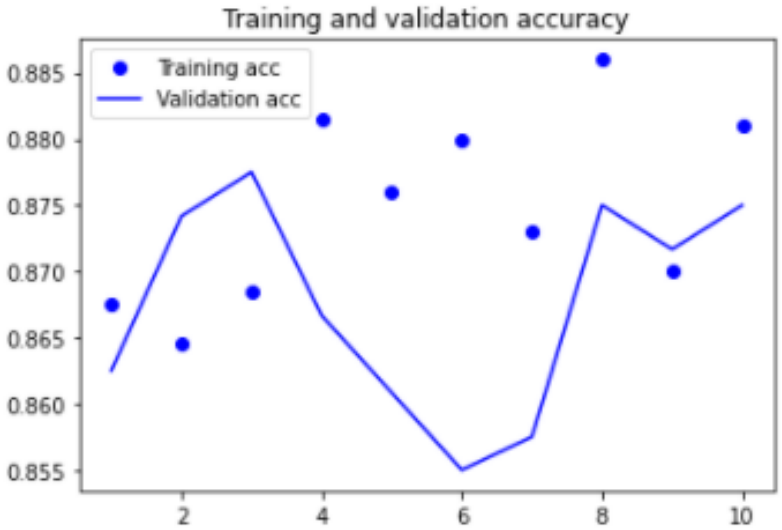

Ilustración 24. Visualización de la precisión durante el entrenamiento y validación del modelo VGG16 con ajuste fino y AD. Fuente: elaboración propia.

Ilustración 25. Visualización de la función de pérdida durante el entrenamiento y validación del modelo VGG16 con ajuste fino y AD Fuente: elaboración propia.

En estas gráficas observamos que sería necesario aplicar más épocas para poder evaluar el modelo y mejorar el rendimiento. No obstante, los resultados obtenidos utilizando un número pequeño de épocas son prometedores.

Precisión	0.8642
Pérdida	0.3664

Tabla 11. Métricas modelo VGG16 con ajuste fino y AD. Fuente: elaboración propia.

5.2. Comparación de modelos

Modelo	Precisión	Pérdida	Épocas	Seg/ step	Seg/ época	Tiempo total
General	0.8337	0.7734	20	4 seg	218 seg	1 hora 15 min
Aumento de datos	0.8591	0.4289	50	3 seg	243 seg	Más de 3 horas
VGG16 sin aumento de datos	0.8258	99.284	50	58 mseg	29 seg	25 min
VGG16 con aumento de datos	0.875	0.346	10	28 seg	2805 seg	Casi 8 horas
VGG16 con ajuste fino y aumento de datos	0.8642	0.3664	10	28 seg	2880 seg	8 horas Aprox.

Tabla 12. Comparación de modelos. Fuente: elaboración propia.

Hay que destacar el elevado coste computacional de la mayoría de los modelos al utilizar un entorno CPU.

También observamos como los mejores modelos se obtienen cuando se realizan técnicas de aumento de datos. Esto se puede apreciar claramente cuando utilizamos la red pre-entrenado VGG16 con y sin aumente de datos.

5.3. Elección final de un modelo

El modelo que se ha considerado que se ajusta mejor a los datos es el modelo VGG16 con aumento de datos. Aunque el modelo VGG16 con ajuste fino también tiene un buen rendimiento, se prefiere el modelo VGG16 sin ajuste fino, ya que con el mismo número de épocas consigue mejores ajustes.

En los modelos que no se realiza un aumento de datos también se obtienen buenos valores de precisión, sin embargo la pérdida si es bastante más elevada en comparación con el resto, especialmente la pérdida del modelo VGG16 sin AD.

5.4. Uso de la aplicación web

Se puede acceder a la aplicación web a partir de este enlace. A continuación se realiza un breve tutorial de la aplicación web.

La pantalla principal de la aplicación se muestra en la Ilustración 26

Ilustración 26. Pantalla principal aplicación web. Fuente: elaboración propia.

En la pantalla principal aparece un mensaje que indica al usuario que debe seleccionar una imagen desde su ordenador, lo que significa que la imagen debe estar descargada.

Al pulsar el botón de “seleccionar archivo” la aplicación nos permite seleccionar una imagen de nuestro ordenador como se puede comprobar en la Ilustración 27.

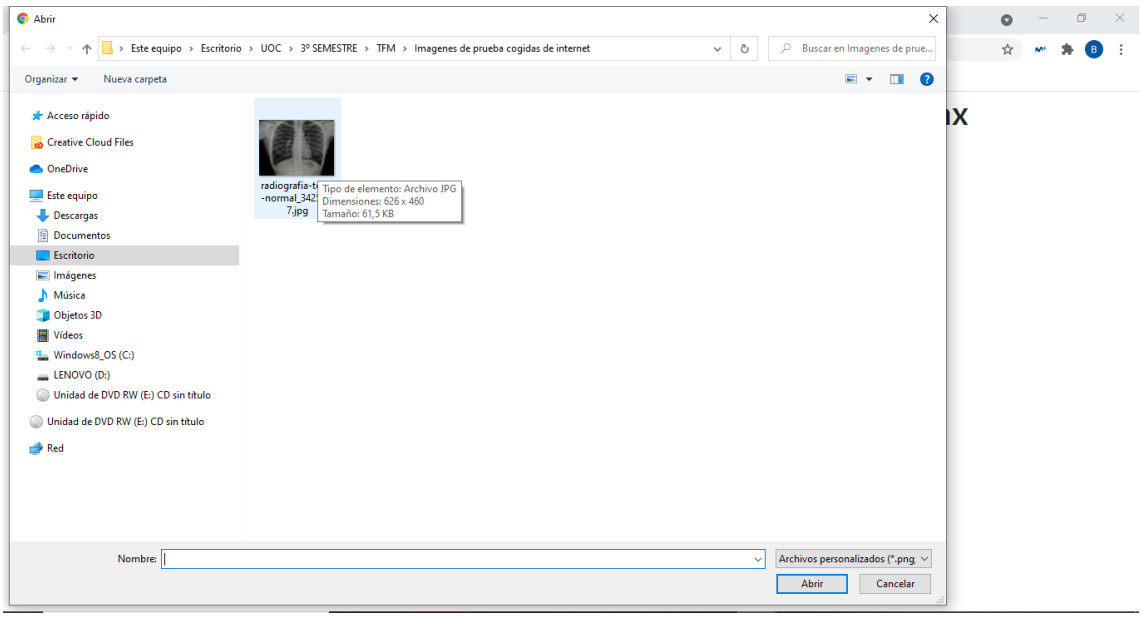

Ilustración 27. Seleccionar una imagen en la aplicación web. Fuente: elaboración propia.

Una vez que hemos seleccionado la imagen, esta aparece en la pantalla principal siendo el momento de realizar la predicción.

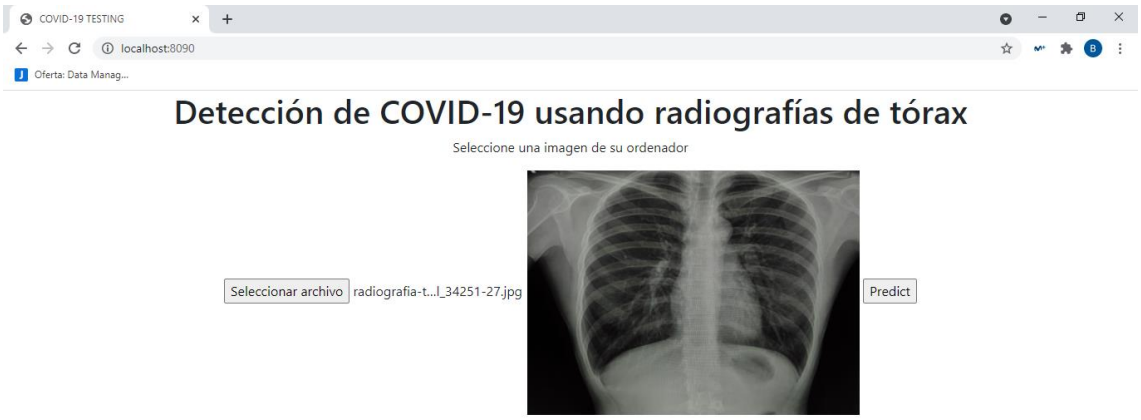

Ilustración 28. Imagen seleccionada en la aplicación web. Fuente: elaboración propia.

Finalmente al pulsar sobre el botón “predict” la aplicación nos devuelve el resultado de la predicción indicando si se trata de una imagen propia de covid-19 o no. Dicho resultado viene acompañado de la probabilidad de que se haya realizado correctamente la clasificación.

Ilustración 29. Resultado predicción aplicación web. Fuente: elaboración propia.

6 Discusión

El Covid-19 es una emergencia sanitaria global lo que hace que haya en la bibliografía gran cantidad de proyectos que pretenden servir como apoyo a los profesionales a la hora de tomar decisiones.

No se pretende diagnosticar a una persona de Covid-19 si no que se pretende crear una herramienta que sirva de ayuda a los profesionales aumentando su capacidad de diagnóstico y además permitiendo comparar sus resultados con miles de imágenes etiquetadas.

Se ha demostrado que con pocas líneas de código se pueden alcanzar altos niveles de precisión en la detección del Covid-19, sin embargo, a excepción de estudios como el de Wang, L., et al.[21], la mayoría utiliza servidores mucho más potentes que los utilizados en este trabajo, ahorrando de esta manera miles de minutos de tiempo computacional.

Este proyecto está enfocado de forma similar a los existentes en la bibliografía como el de Wang, L., et al.[21] o Narin, A.[23], utilizando métodos de aumento de datos y de transferencia de aprendizaje.

Sin embargo, en cuanto a la clasificación de las imágenes, la mayoría de los trabajos como el de Cortés Pérez, E.[25], opta por realizar una clasificación binaria, estableciendo las clases normal y Covid-19 y obteniendo así los niveles de precisión más elevados encontrados en la bibliografía.

La utilización de la red pre-entrenada VGG-16 la podemos observar en los trabajos de 2 y 6. En el de 2 se obtiene una precisión del 97.36%. En nuestro caso, aunque también se ha utilizado la red VGG-16 no se han podido realizar el mismo número de épocas debido al alto coste computacional obteniendo resultados por debajo del 90%.

7 Conclusiones

7.1. Conclusiones

Se han conseguido desarrollar diferentes modelos de redes neuronales convolucionales con una precisión mayor del 70% en la detección de Covid-19. El modelo que ofrece un mejor rendimiento es el que utiliza la red pre-entrenada VGG-16 junto con aumento de datos, llegando a una precisión del 87%.

Dicho modelo se ha utilizado para la creación de una aplicación web capaz de clasificar las imágenes en cuatro categorías: normal, opacidad torácica, neumonía viral y neumonía Covid-19.

Al realizar el proyecto ha quedado constancia de la importancia de realizar modelos de aprendizaje automático en entornos con mayor capacidad computacional como mediante la utilización de sistema operativo GNU/Linux o la utilización de plataformas como Google Colab que permite acceso gratuito a servidores GPUs.

7.2. Líneas de futuro

- En lo que refiere a la salud, no nos podemos conformar con los resultados obtenidos, sino que debemos intentar mejorar nuestro modelo para que sea lo más preciso posible. Por ello, una línea futura podría ser implementar nuestros modelos en un entorno GPU modificando los parámetros de entrenamiento o aumentando el número de imágenes.

- Mejora de la aplicación web. La aplicación además de emitir un resultado, sería muy útil que mostrara las características de la imagen que han provocado que la red tome esa decisión, por ejemplo señalando las zonas más opacas (con menores niveles aéreos).

- Aumento de la base de datos para el entrenamiento. Una idea prometedora al realizar una aplicación web es utilizar las imágenes subidas por los usuarios para aumentar la base de datos de nuestro entrenamiento pidiendo al mismo usuario que proporcione una especie de verificación de la clase predicha. Por ejemplo, si se realiza una aplicación para uso médico, se pediría al profesional que indicara su grado de acuerdo con la clase predicha por el modelo. De esta manera, la imagen se añadiría a la base de entrenamiento etiquetada.

- Se ha demostrado que el aprendizaje profundo es muy útil en el apoyo para el diagnóstico de imágenes médicas por lo que en un futuro estas herramientas podrían estar implementadas en los programas que utilizan los profesionales para visualizar las radiografías.

7.3. Seguimiento de la planificación

En cuanto a la planificación, aunque se intentó llevar la temporalización de las actividades según el calendario inicial, el transcurso del proyecto ha obligado a retrasar algunas actividades alterando así la planificación.

Algunos de los aspectos que han hecho retrasar la consecución de algunos objetivos ha sido encontrarnos con actividades no previstas que han consumido

demasiado tiempo. Entre ellas destaca la creación/preparación de un entorno de trabajo, ya que inicialmente se iba a trabajar con Python desde una máquina virtual utilizando Ubuntu, sin embargo, las librerías de Keras y Tensorflow no eran compatibles con dicha máquina y se decidió utilizar Python en Windows a través de Anaconda.

Otro de los problemas más importantes surgidos durante el desarrollo del proyecto fue la falta de espacio que se consiguió solventar aumentando la memoria virtual o archivo de paginación.

Además, algunas actividades no fueron planificadas con la duración adecuada, especialmente la actividad de “mejora del rendimiento del modelo” que al requerir la creación y entrenamiento de diferentes modelos ha consumido tiempo a otras actividades.

8 Glosario

- **AD**: aumento de datos.
- **ANN**: del inglés, Artificial Neural Network.
- **CNN**: del inglés, Convolutional Neuronal Network.
- **Covid-19**: del inglés, Coronavirus disease 2019.
- **DL:** del inglés, Deep Learning.
- **IA**: del inglés, Inteligencia Artificial.
- **OMS**: Organización Mundial de la Salud.
- **WSGI:** del inglés, Web Server Gateway Interface.

9 Bibliografía

A continuación se detallan las referencias bibliográficas utilizadas en el proyecto.

1. **Abelllan Madrid, M., Guijarro Marín, A., Verdú Naranjo, J**. (2020). “Detección del COVID-19 con técnicas de Deep Learning”. Puerta de la investigación [artículo en línea]. [Fecha de consulta: 16 de marzo de 2021]. <https://www.researchgate.net/publication/341542106_Deteccion_del_COVID-19_con_tecnicas_de_Deep_Learning >

2. **Singhal, T** (2020). “Una revisión de la enfermedad por coronavirus-2019 (COVID-19)”. Indian J Pediatr [artículo en línea]. 281–28 (vol. 87). [Fecha de consulta: 25 de abril del 2021]. <https://doi.org/10.1007/s12098-020-03263-6 >

3. **Calleja García, J.** “Epidemiología – COVID Reference”. [Internet]. Covidreference.com. 2021 [citado el 26 de Febrero 2021]. Disponible en: https://covidreference.com/epidemiology_es

4. Ministerio de Sanidad, Gobierno de España. “Enfermedad por nuevo coronavirus, COVID-19”. Mscbs.gob.es. [información sanitaria en línea]. [Fecha de consulta: 28 de Febrero de 2021]. <http://www.mscbs.gob.es/profesionales/saludPublica/ccayes/alertasActual/nCov-China/home.htm >

5. **Orus, Abigail** (2021, 10 de junio). “COVID-19: número de muertes a nivel mundial por continente 2021”. [artículo en línea]. [fecha de consulta: 15 de abril de 2021]. <https://es.statista.com/estadisticas/1107719/covid19-numero-de-muertes-a-nivel-mundial-por-region/>

6. **Rtve. “**Coronavirus”. [artículo en línea]. [Fecha de consulta: 20 de abril del 2021]. <https://www.rtve.es/noticias/coronavirus-graficos-mapas-datos-covid-19-espana-mundo/>

7. **Orús, Abigail** (2021, 14 de junio). “COVID-19: fallecidos por día en España 2020-2021”. [artículo en línea]. [Fecha de consulta: 21 de abril del 2021]. <https://es.statista.com/estadisticas/1104277/fallecidos-a-causa-de-covid-19-por-dia-espana/>

8. **Ministerio de Sanidad, Gobierno de España** (2021, 11 de junio). “Gestión integral de la vacunación COVID-19”. [Informe en línea]. [Fecha de consulta: 20 de abril del 2021]. https://www.mscbs.gob.es/profesionales/saludPublica/ccayes/alertasActual/nCov/documentos/Informe_GIV_comunicacion_20210611.pdf

9. **Chen, N., Zhou, M., Dong, X. et al.** (2020, 30 de junio). “Características epidemiológicas y clínicas de 99 casos de neumonía por el nuevo coronavirus de 2019 en Wuhan, China: un estudio descriptivo”. Lanceta [artículo en línea]. 507-513 (vol. 395). [Fecha de consulta: 12 de abril de 2021]. <https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(20)30211-7/fulltext >

10. **Centro de Coordinación de Alertas y Emergencias Sanitarias, Ministerio de Sanidad, Gobierno de España** (2021, 29 de abril). “COVID-19 en distintos entornos y grupos de personas”. [PDF en línea]. [Fecha de consulta: 24 de abril del 2021]. <https://www.mscbs.gob.es/profesionales/saludPublica/ccayes/alertasActual/nCov/documentos/20210429_GRUPOSPERSONAS.pdf>

11. **Martínez Chamorro, E., Díez Tascón, A., Ibáñez Sanz, L., Ossaba Vélez, S., Borruel Nacenta, S**. (2021). “Radiologic diagnosis of patients with COVID-19. Diagnóstico radiológico del paciente con COVID-19”. Radiologia [artículo en línea].56–73 (vol. 63, núm. 1). [Fecha de consulta: 1 de mayo de 2021]. <https://doi.org/10.1016/j.rx.2020.11.001 >

12. **Orozco, C., Xamena, E., Martínez, C., Rodríguez, D**. (2021). “COVID–XR: A Web Management Platform for Coronavirus Detection on X-Ray Chest Images”. IEEE LATIN AMERICA TRANSACTIONS [artículo en línea]. (vol.19, núm. 6). [Fecha de consulta: 12 de mayo de 2021]. <https://latamt.ieeer9.org/index.php/transactions/article/view/4402 >

13. **Wang, S., Zha, Y., Li, W., Wu, Q., Li, X., Niu, M., Wang, M., Qiu, X., Li, H., Yu, H., Gong, W., Bai, Y., Li, L., Zhu, Y., Wang, L., Tian, J**. (2020). “Un sistema de aprendizaje profundo completamente automático para el análisis de diagnóstico y pronóstico de COVID-19”. Revista respiratoria europea [artículo en línea]. (vol. 56, núm. 2). [Fecha de consulta: 10 de mayo de 2021]. < https://doi.org/10.1183/13993003.00775-2020 >

14. **Brownlee J.** Un recorrido por los algoritmos de aprendizaje automático [Internet]. Dominio del aprendizaje automático. 2019 [citado el 16 de marzo de 2021]. Disponible en: https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

15. Aprende Machine Learning (2018, 29 de noviembre). “Redes neuronales convolucionales: La Teoría explicada en Español”. [artículo en línea]. [Fecha de consulta: 16 de marzo de 2021]. <https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/>

16. **Martínez Llamas J.** (2021). “Reconocimiento de imágenes mediante redes neuronales convolucionales”. [artículo en línea]. [Fecha de consulta: 2 de mayo de 2021]. <http://oa.upm.es/53050/1/TFG_JAVIER_MARTINEZ_LLAMAS.pdf>
17. **Calvo D**. Red Neuronal Convolucional CNN [Internet]. Diegocalvo.es. 2017 [citado el 16 de Marzo de 2021]. Disponible en: https://www.diegocalvo.es/red-neuronal-convolucional/

18. **Emtiaz, H., Mahmudul, H., Md Anisur, Rahman., Ickjai, Lee., Tasmi Tamanna, Mohammad Zavid Parvez** (junio de 2021). “CoroDet: A deep learning based classification for COVID-19 detection using chest X-ray images”. Chaos, Solitons & Fractals [artículo en línea].(Vol.142, num, 110495). Elsevier. [Fecha de consulta: 10 de mayo de 2021].< https://www.sciencedirect.com/science/article/pii/S0960077920308870> ISSN 0960-0779

19. **Kabid Hassan Shibly, Samrat Kumar Dey, Md Tahzib-Ul Islam, Md Mahbubur Rahman** (2020). “COVID faster R–CNN: A novel framework to Diagnose Novel Coronavirus Disease (COVID-19) in X-Ray images”. Informatics in Medicine Unlocked, [artículo en línea]. (Vol. 20, núm. 100405). Elsevier. [Fecha de consulta: 10 de mayor de 2021].< https://www.sciencedirect.com/science/article/pii/S2352914820305554 > ISSN 2352-9148

20. **Dhiman, G., Vinoth Kumar, V., Kaur, A. et al** (2021). “DON: Deep Learning and Optimization-Based Framework for Detection of Novel Coronavirus Disease Using X-ray Images”. Interdiscip Sci Comput Life Sci [artículo en línea]. (Vol.13, págs..260–272).[Fecha de consulta: 10 de mayo de 2021]. <https://www.sciencedirect.com/science/article/pii/S2352914820305554?via%3Dihub >

21. **Mahmud, T., Rahman, MA., Fattah, SA.** (2020, junio).”CovXNet: A multi-dilation convolutional neural network for automatic COVID-19 and other pneumonia detection from chest X-ray images with transferable multi-receptive feature optimization”. Comput Biol Med. [artículo en línea].[Fecha de consulta: 10 de mayo de 2021]. < https://pubmed.ncbi.nlm.nih.gov/32658740/ >

22. **Wang, L., Lin, Z.Q. & Wong, A**. (2020, 11 de noviembre). “COVID-Net: a tailored deep convolutional neural network design for detection of COVID-19 cases from chest X-ray images”. Sci Rep [artículo en línea]. (Vol.10, núm. 19549). [Fecha de consulta: 21 de mayo de 2021]. < https://www.nature.com/articles/s41598-020-76550-z >

23. **Apostolopoulos ID, Mpesiana TA** (2020, 3 de abril). “Covid-19: automatic detection from X-ray images utilizing transfer learning with convolutional neural networks”. Phys Eng Sci Med [artículo en línea]. 635-

640 (vol.2, núm. 43). [Fecha de consulta: 20 de mayo de 2021]. < https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7118364/ >

24. **Narin, A., Kaya, C. & Pamuk, Z** (2021, 9 de mayo). "Automatic detection of coronavirus disease (COVID-19) using X-ray images and deep convolutional neural networks". *Pattern Anal Applic*. [artículo en línea]. [Fecha de consulta: 20 de mayo de 2021. < https://link.springer.com/article/10.1007/s10044-021-00984-y >

25. **Sait U, K V GL, Shivakumar S, et al** (2021, 26 de mayo). "A deep-learning based multimodal system for Covid-19 diagnosis using breathing sounds and chest X-ray images". Appl Soft Comput. [artículo en línea]. [Fecha de consulta: 28 de mayo de 2021]. < https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8149173/ >

26. **Cortes Pérez, E**. (2020, diciembre). "Deep Learning Transfer with AlexNet for chest X-ray COVID-19 recognition".[artículo en línea]. [Fecha de consulta: 10 de mayo de 2021]. < https://www.researchgate.net/publication/348481655_Deep_Learning_Transfer_with_AlexNet_for_chest_X-ray_COVID-19_recognition >

27. **Perrilliat García, I., Gámez Guerrero, M. A., Rocha Nava, S. L., Hernández Oropeza, J. I**. (2020). "Sistema Auxiliar para el Diagnóstico de COVID-19 Mediante Análisis de Imágenes de CR Torácica Basado en Deep Learning". Memorias Del Congreso Nacional De Ingeniería Biomédica. [artículo en línea]. 556 – 563 (Vol. 7, núm. 1). [Fecha de consulta: 10 de mayo de 2021]. < http://memorias.somib.org.mx/index.php/memorias/article/view/810 >
28. Cordis (2021, 24 de abril). "El sonido de la COVID-19: una nueva aplicación identifica los síntomas a través de la voz y la tos". [noticia en línea]. [Fecha de consulta: 15 de mayo de 2021]. < https://cordis.europa.eu/article/id/417988-the-sound-of-covid-19-new-app-to-identify-symptoms-through-voices-and-coughs/es >

29. University of Cambridge (2020) "COVID-19 Sounds App". [Página web]. [Fecha de consulta: 10 de mayo de 2021].< https://www.covid-19-sounds.org/es/ >

30. **Márquez Díaz, J** (2020, 23 de noviembre). "Inteligencia artificial y Big Data como soluciones frente a la COVID-19". Bioética y Derecho [artículo en línea]. (núm. 50). [Fecha de consulta: 10 de mayo de 2021]. < https://scielo.isciii.es/scielo.php?pid=S1886588720200003000 19&script=sci_arttext&tlng=pt >

31. Agencia SINC (2020, 5 de agosto). "Radiografías de tórax para detectar la COVID-19". [artículo en línea]. [Fecha de consulta: 10 de mayo de 2021]. < https://www.agenciasinc.es/Noticias/Radiografias-de-torax-para-detectar-la-COVID-19 >

32. **Torra i Reventós, V.** “Qué es la inteligencia artificial”. UOC [libro electrónico]. [Fecha de consulta: 10 de mayo de 2021]. < http://cvapp.uoc.edu.eu1.proxy.openathens.net/autors/MostraPDFMaterialAction.do?id=163094 >

33. **De la Torre, D** (2019, 8 de enero). “¿Cómo nació la Inteligencia Artificial?”. [artículo en línea]. [Fecha de consulta: 10 de mayo de 2021]. < https://blogthinkbig.com/historia-como-nacio-inteligencia-artificial >

34. **Rouhiainen, L** (2018). “Inteligencia artificial 101 cosas que debes saber hoy sobre nuestro futuro”. [libro electrónico]. Alienta Editorial. [Fecha de consulta: 10 de mayo de 2021]. < https://static0planetadelibroscom.cdnstatics.com/libros_contenido_extra/40/39308_Inteligencia_artificial.pdf >

35. **Handelman, GS., Kok, HK., Chandra, RV., Razavi, AH., Lee, MJ., Asadi, H.** (2018, 13 de Agosto). “eDoctor: machine learning and the future of medicine. (Review)” J Intern Med [artículo en línea]. 603– 619 (vol. 284). [Fecha de consulta: 12 de mayo de 2021]. < https://onlinelibrary.wiley.com/doi/full/10.1111/joim.12822 >

36. Instituto de Ingeniería del Conocimiento. “Machine Learning & Deep Learning”. [artículo en línea]. [Fecha de consulta: 13 de mayo de 2021]. < https://www.iic.uam.es/inteligencia-artificial/machine-learning-deep-learning/ >

37. **Oppermann, Artem** (2019, 12 de noviembre). “What is Deep Learning and How does it work?”. [artículo en línea]. [Fecha de consulta: 21 de mayo de 2021]. < https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work-2ce44bb692ac >

38. **Peris Ripollés, G.** (2015, 9 de diciembre). “Acertando quinielas con redes neuronales”. Naukas [artículo en línea]. [Fecha de consulta: 21 de mayo de 2021]. <https://naukas.com/2015/12/09/acertando-quinielas-redes-neuronales/>

39. Ambientech. “La neuorna”. [artículo en línea]. [Fecha de consulta: 21 de mayo de 2021]. < https://ambientech.org/la-neurona >

40. **Vega Pedrero, MJ**. (2016, 7 de noviembre). “Conexión neuronal: el potencial de acción”. Neuropsicología [artículo en línea]. [Fecha de consulta: 10 de mayo de 2021]. <https://hablemosdeneurociencia.com/conexion-neuronal-potencial-accion/ >

41. **Chollet, F** (2017, diciembre). “Deep Learning with Python”. Manning Publications [libro electrónico]. [Fecha de consulta: 14 de abril de 2021]. <https://learning.oreilly.com/library/view/deep-learning-with/9781617294433/ >

42. **Lantz, B**. (2015, 31 de julio). “Machine Learning with R – Second Edition”. [libro electrónico]. [Fecha de consulta: 1 de junio de 2021]. <https://ebookcentral.proquest.com/lib/bibliouocsp-ebooks/detail.action?docID=2122139 >

43. **Calvo, D.** (2018, 7 de diciembre). “Función de activación – Redes neuronales”. [artículo en línea]. [Fecha de consulta: 1 de mayo de 2021]. < https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/ >

44. **Ballesteros, A**. “Neural Network Framework”. [tutorial en línea]. [Fecha de consulta: 10 de mayo de 2021]. < http://www.redes-neuronales.com.es/tutorial-redes-neuronales/tutorial-redes.htm >

45. **Barrios, J.** (2021). “Redes Neuronales Convolucionales”. Health Big Data [artículo en línea]. [Fecha de consulta: 15 de mayo del 2021]. < https://www.juanbarrios.com/redes-neurales-convolucionales/ >

46. **López Briega, R**. (2016, 2 de agosto). “Redes neuronales convolucionales con TensorFlow”. [artículo en línea]. [Fecha de consulta: 21 de mayo de 2021]. <https://relopezbriega.github.io/blog/2016/08/02/redes-neuronales-convolucionales-con-tensorflow/ >

47. Bootcamp AI. (2019, 23 de noviembre), “Introducción a las redes neuronales convolucionales”. [artículo en línea]. [Fecha de consulta: 20 de mayo de 2021]. < https://bootcampai.medium.com/redes-neuronales-convolucionales-5e0ce960caf8 >

48. **Martínez, J.** (2019). “Redes Neuronales Convolucionales en Profundidad”. [artículo en línea]. [Fecha de consulta: 23 de mayo de 2021]. <https://datasmarts.net/es/redes-neuronales-convolucionales-en-profundidad/ >

49. **Rahman, T.** (2020). “COVID-19 Radiography Database”. [conjunto de datos en línea]. [Fecha de consulta: 15 de marzo de 2021]. <https://www.kaggle.com/tawsifurrahman/covid19-radiography-database>

50. **Bernardes, F.** (2019, 27 de agosto). “Top 15: los lenguajes de programación más usados en 2021”. [artículo en línea]. [Fecha de consulta: 25 de mayo de 2021]. <https://es.classpert.com/blog/lenguajes-de-programacion-mas-usados#python>

51. Graph Every Where. “Machine Learning en Python”. [artículo en línea]. [Fecha de consulta: 12 de mayo de 2021]. <https://www.grapheverywhere.com/machine-learning-en-python/>

52. **Toro, L**. “Anaconda Distribution: La Suite más completa para la Ciencia de datos con Python”. [artículo en línea]. [Fecha de consulta: 11 de abril de 2021]. <https://blog.desdelinux.net/ciencia-de-datos-con-python/>

53. Anaconda Documentation. “Getting started with Anaconda”. [guía de usuario en línea]. [Fecha de consulta: 13 de mayo de 2021]. <https://docs.anaconda.com/anaconda/user-guide/getting-started/>

54. **Stojiljkovic, M**. (2020, 23 de noviembre). “Split your dataset with scikit-learn’s train_test_split()”. [artículo en línea]. [Fecha de consulta: 12 de mayo de 2021]. <https://realpython.com/train-test-split-python-data/>

55. TensorFlow. ”tf.keras.layers.MaxPool2D”. [documentación en línea]. [Fecha de consulta: 10 de mayo de 2021]. <https://www.tensorflow.org/api_docs/python/tf/keras/layers/MaxPool2D>

56. **Calvo, D.** (2018, 10 de diciembre). “Función de coste - redes neuronales”. Aprendizaje automático. [artículo en línea]. [Fecha de consulta: 10 de mayo de 2021]. <https://www.diegocalvo.es/funcion-de-coste-redes-neuronales/>