|  |  |
| --- | --- |
| Logotipo  Descripción generada automáticamente con confianza baja | Logotipo  Descripción generada automáticamente |

Trabajo Fin de Máster

Máster en Ciencia de Datos e Ingeniería de Datos en la Nube

CIDANet, red neuronal convolucional para el diagnóstico de COVID-19 mediante radiografías CXR

|  |  |
| --- | --- |
| **Autor:** | Daniel Fernández Gómez |
| **Tutor:** | Tutor nombre |
| **Co-Tutor:** | Co-tutor nombre |

07, 2021

Dedicado a mi madre, por su apoyo incondicional…

Declaración de Autoría

Yo, DANIEL FERNÁNDEZ GÓMEZ con DNI 48154227R, declaro que soy el único autor del trabajo fin de grado titulado “......” y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Albacete, a.....

Fdo: ......

Resumen

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

Agradecimientos

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

Índice general

[1.1 Introducción 1](#_Toc77619889)

[1.2 Objetivos 2](#_Toc77619890)

[1.3 Estructura del proyecto 3](#_Toc77619891)

[2.1 Cloud Computing 5](#_Toc77619892)

[2.2 Deep Learning 6](#_Toc77619893)

[2.2.1 Red neuronal convolucional (CNN) 6](#_Toc77619894)

[2.2.2 Transfer learning 7](#_Toc77619895)

[2.3 Trabajos relacionados 9](#_Toc77619896)

[3.1 Introducción 11](#_Toc77619897)

[3.2 Dataset 11](#_Toc77619898)

[3.3 Pre procesamiento 12](#_Toc77619899)

[3.4 Preparación de los experimentos 13](#_Toc77619900)

[3.5 Experimento 1: MobileNet 14](#_Toc77619901)

[3.6 Experimento 2: Red InceptionV3 16](#_Toc77619902)

[3.7 Experimento 3: Red VGG16 18](#_Toc77619903)

[3.8 Experimento 4: MobileNet vs InceptionV3 20](#_Toc77619904)

[3.9 Evaluación de los resultados 24](#_Toc77619905)

[4.1 Introducción. 25](#_Toc77619906)

[4.2 Arquitectura Cloud. 26](#_Toc77619907)

[4.2.1 S3 26](#_Toc77619908)

[4.2.2 DynamoDB 27](#_Toc77619909)

[4.2.3 ECR 27](#_Toc77619910)

[4.2.4 APIGateway 27](#_Toc77619911)

[4.2.5 Lambda 27](#_Toc77619912)

[4.3 Página web 28](#_Toc77619913)

[5.1 Conclusiones 30](#_Toc77619914)

[5.2 Trabajo futuro 30](#_Toc77619915)

Índice de figuras

[Figura 1 Comparación técnica Fuzzy 13](#_Toc77619916)

[Figura 2 Capas añadidas a la red CIDANet 13](#_Toc77619917)

[Figura 3 Arquitectura MobileNet 14](#_Toc77619918)

[Figura 4 Estadísticas MobileNet 15](#_Toc77619919)

[Figura 5 Epochs MobileNet 15](#_Toc77619920)

[Figura 6 Epoch-accuracy MobileNet 15](#_Toc77619921)

[Figura 7 Arquitectura InceptionV3 16](#_Toc77619922)

[Figura 8 Estadísticas InceptionV3 16](#_Toc77619923)

[Figura 9 Tiempos InceptionV3 17](#_Toc77619924)

[Figura 10 Epoch-accuracy InceptionV3 17](#_Toc77619925)

[Figura 11 Arquitectura VGG16 18](#_Toc77619926)

[Figura 12 Estadísticas VGG16 19](#_Toc77619927)

[Figura 13 Tiempos VGG16 19](#_Toc77619928)

[Figura 14 Evolución epoch-accuracy VGG16 19](#_Toc77619929)

[Figura 15 Epochs-accuracy InceptionV3 FullDataset 21](#_Toc77619930)

[Figura 16 Estadísticas InceptionV3 FullDataset 21](#_Toc77619931)

[Figura 17 Velocidad InceptionV3 FullDataset 22](#_Toc77619932)

[Figura 18 Estadísticas MobileNet FullDataset 23](#_Toc77619933)

[Figura 19 Matriz de confusión MobileNet FullDataset 23](#_Toc77619934)

[Figura 20 Red CIDANet 24](#_Toc77619935)

[Figura 21 Arquitectura AWS 26](#_Toc77619936)

[Figura 22 Página Web 29](#_Toc77619937)

[Figura 23 LIME y GRADCAM 31](#_Toc77619938)

Índice de tablas

[Tabla 1 Comparación modelos dataset reducido 20](#_Toc77619939)

[Tabla 2 Comparativas experimento final 24](#_Toc77619940)

# Introducción

## Introducción

## Introducción

La población global está sufriendo el avance de un recientemente descubierto coronavirus (COVID-19). El SARS-CoV-2, o comúnmente conocido como COVID-19, ha demostrado ser un virus muy contagioso y además peligroso para la salud de las personas, especialmente en aquellas que ya padecen alguna otra enfermedad. La similitud de síntomas con otras enfermedades respiratorias que también provocan neumonía, el desconocimiento del virus y su evolución, junto a los pocos medios para combatirla han hecho de esta una enfermedad mortal, cobrándose la vida de millones de personas alrededor del mundo [1].

La rapidez con la que se expande el virus lleva a los países, incluso a los más aventajados, al colapso hospitalario. Actualmente se utilizan pruebas de tipo *Reverse Transcription Polymerase Chain Reaction* (RT-PCR) para la detección del virus [2].

Otro método utilizado para la detección de la COVID-19 es el análisis de radiografías de tórax (CXR) y las tomografías computacionales (CT), ya que la enfermedad causa una serie de patrones radiológicos llamados “*ground-glass opacities (CGO)* [3]” en la caja torácica de los pacientes.

Este método ofrece grandes ventajas sobre todo en términos de rapidez con respecto a las RT-PCR. Sin embargo, la similitud entre las radiografías de pacientes con COVID-19 y neumonías causadas por otras infecciones complicaban la labor de los médicos a la hora de interpretarlas, y tenían que recurrir a expertos radiólogos [4].

Motivados por la urgente situación que atravesamos, fueron publicadas una gran cantidad de bases de datos con imágenes de radiografías con el fin de que la comunidad científica tratase de aunar esfuerzos en busca de un clasificador preciso que aliviase la situación sanitaria.

Este proyecto incorpora parte de esas bases de datos para elaborar un modelo capaz de distinguir entre COVID-19 y otro tipo de neumonías con el fin de servir de ayuda para el análisis prematuro de los síntomas ocasionados por la enfermedad.

## Objetivos

Se propone la elaboración de un modelo con la tecnología de *Deep Learning*, capaz de detectar esas pequeñas diferencias que separan una neumonía causada por el SARS-CoV-2 y otras neumonías procedentes de otras infecciones de una manera eficiente, económica y con una tasa baja de error a la hora de la clasificación.

Para ello se creará una red neuronal capaz de clasificar radiografías de tórax en tres categorías distintas en función del tipo de neumonía que presente el paciente.

Una vez elaborado el modelo, se subirá a una página web desplegada en *cloud* (Amazon Web Services) para poder ser utilizada de manera pública por todo el mundo.

Puesto que por suerte no nos encontramos en una situación de colapso sanitario, el objetivo de este modelo será ayudar en la detección de las complicaciones generadas por el virus de la COVID-19, apoyando de esta forma no tanto a los radiólogos, sino a personal médico no especializado en radiografía digital, quienes en primera instancia realizan dicho análisis.

## Estructura del proyecto

Son dos grandes bloques los que conforman este proyecto. Por un lado, tenemos el del diseño de la red neuronal, junto con su experimentación y pre procesado de imágenes, además de la búsqueda de un *dataset* que contenga una buena variedad de imágenes.

Por otro lado, la creación de una arquitectura *cloud* en AWS para el despliegue *serverless* de una aplicación web que realice las predicciones de las imágenes que subamos a la plataforma.

# Estado del Arte

## *Cloud computing*

De una manera simple, podríamos definir la computación en la nube (*Cloud computing*) como una tecnología que permite acceso remoto a herramientas, almacenamiento y procesamiento de datos, todo a través de internet sin necesidad de ejecutar en nuestra maquina local.

En nuestro proyecto utilizaremos AWS, un proveedor de servicios en la nube.

Algunos de los servicios que usaremos son:

* **S3**: Amazon S3 es un servicio de almacenamiento de objetos creado para almacenar y recuperar cualquier volumen de datos desde cualquier ubicación. Es un servicio de almacenamiento sencillo que ofrece excelente durabilidad, disponibilidad, rendimiento, seguridad y escalabilidad prácticamente ilimitada a costos muy reducidos.
* **Lambda**: Lambda es un servicio informático que le permite ejecutar código sin aprovisionar ni administrar servidores. Lambda ejecuta el código en una infraestructura informática de alta disponibilidad y realiza toda la administración de los recursos informáticos, incluido el mantenimiento del servidor y del sistema operativo, el aprovisionamiento de la capacidad y el *auto scaling*, la supervisión y el registro de código. Con Lambda, puede ejecutar código para prácticamente cualquier tipo de aplicación o servicio de *backend* [5].
* **ECR**: *Amazon Elastic Container Registry* (ECR) es un registro de contenedores de *Docker* completamente administrado que facilita a los desarrolladores el almacenamiento, la administración, el uso compartido y la implementación de artefactos e imágenes de contenedores en cualquier parte [5].
* **DynamoDB**: Amazon DynamoDB es un servicio de base de datos NoSQL totalmente administrado que ofrece un rendimiento rápido y predecible, así como una perfecta escalabilidad. DynamoDB le permite transferir las cargas administrativas que supone tener que utilizar y escalar bases de datos distribuidas, para que no tenga que preocuparse del provisionamiento, la instalación ni la configuración del hardware, ni tampoco de las tareas de replicación, aplicación de parches de software o escalado de clústeres [5].

## *Deep Learning*

El nombre viene de la idea de imitar el funcionamiento de las redes neuronales de organismos vivos: un conjunto de neuronas conectadas entre sí y que trabajan en conjunto, sin que haya una tarea concreta para cada una. Con la experiencia, las neuronas van creando y reforzando ciertas conexiones para “aprender” algo que se queda fijo en el tejido nervioso. El objetivo de la red neuronal es resolver los problemas de la misma manera que el cerebro humano, aunque las redes neuronales son más abstractas. Las redes neuronales actuales suelen contener desde unos miles a unos pocos millones de unidades neuronales [6] .

### Red neuronal convolucional (CNN)

Las redes profundas que se utilizan en tareas como la detección de objetos, la segmentación de imágenes, el reconocimiento de imágenes y otras tareas relacionadas con la visión por computadora son todas redes neuronales convolucionales o CNN.

Una red convolucional intenta extraer características esenciales de la imagen de entrada a través de una progresión de capas convolucionales con canales (Kernels), capas agrupadas, capas completamente conectadas (*Full Connected*, FC), y luego, se aplica una función de activación para clasificar la imagen con características probabilísticas. La red neuronal de convolución, como su nombre indica, contiene capas de convolución. Las capas de convolución se componen de varios filtros, y cada filtro extrae diferentes tipos de características. La operación de convolución se introdujo en las redes neuronales para evitar el sobreajuste en las redes neuronales y para hacer que la red neuronal se concentre en características esenciales [7].

### Transfer learning

Consiste en el proceso por el cual se obtiene un modelo ya entrenado y se utiliza para obtener una clasificación en otro problema distinto para el que fue entrenado.

El *transfer learning* se puede realizar de 2 formas:

* Extracción de características: Mediante esta técnica, se toma un modelo que se entrena previamente con algún conjunto de datos como puede ser ImageNet. La red restante se trata como un extractor de características en el que se puede ejecutar cualquier algoritmo clasificador [8].
* *Fine Tuning*: Mediante esta técnica, bloqueamos parte de las capas de la red para que no se entrenen, y agregamos algunas capas a la parte superior con el fin de que se entrene conjuntamente tanto las capas de clasificador recién agregadas como las últimas capas del modelo base. Esto nos permite "ajustar" las representaciones de características de orden superior en el modelo base para hacerlas más relevantes para la tarea específica [9].

## Trabajos relacionados

La detección de COVID-19 mediante técnicas de *Deep Learning* es un tema de tendencia y ha atraído una gran atención en los últimos meses. Se ha publicado una gran cantidad de resultados prometedores utilizando CNN y aún siguen avanzando, obteniendo cada vez mejores resultados. En la detección de COVID-19, las bases de datos con imágenes tipo CXR y CT son las más utilizadas para la clasificación. Además, recientemente se han utilizado señales de sensores portátiles como entradas para la detección de COVID-19 [10].

Durante estos meses, se han diseñado una gran variedad de modelos de *Deep Learning* con una gran tasa de acierto. En el artículo [11], los investigadores propusieron un tipo de método de *Deep Learning* denominado *DarkCovidNet,* que logró una precisión en las pruebas de un 98,08% en la clasificación binaria y una precisión del 87,02% en la clasificación de tres categorías. En [12], se utilizó COVID-Net, un modelo de *Deep Learning* que hacía uso de la denominada *generative synthesis* [13] para la detección de COVID-19 en CXR. Los autores de esta investigación además utilizaron imágenes CXR de fuentes abiertas y las pusieron a disposición del público general. Para las imágenes de CT de tórax se propuso un tipo de modelo COVIDNet-CT en el artículo [14] para identificar COVID-19, neumonía no producida por COVID-19 y casos normales. Sin embargo, debe reconocerse que también existen limitaciones en la detección de COVID-19 mediante estas técnicas. En primer lugar, la mayoría de las bases de datos fueron recogidas durante las épocas de mayor presión sanitaria en la que muchos países impusieron cuarentenas para controlar el desplazamiento de las personas y por ello, los pacientes que acudían al hospital y a las que se les realizaba pruebas CXR Y CT eran pacientes que presentaban síntomas graves de COVID-19 y por tanto es mucho mayor el número de este tipo de imágenes con respecto a los pacientes con síntomas más leves, los cuales eran aislados de manera preventiva en sus viviendas [15]. En segundo lugar, las investigaciones, especialmente las que se llevaron a cabo en el año 2020, adolecían de una falta de imágenes de COVID-19 y un desbalanceo de datos. Por ejemplo, en diversos estudios [16] [17] [15], el número de imágenes de COVID-19 variaba de 50 a 300, mientras el número de datos en otras categorías es significativamente mayor, complicando la labor de la red neuronal. En tercer lugar, no existe un criterio estándar en la evaluación de modelos, lo que conduce a una difícil comparación de rendimiento entre diferentes modelos. Además, con el creciente número de métodos de *Deep Learning* es cada vez más difícil para los investigadores y las organizaciones de salud seleccionar el método de clasificación más apropiado para la detección de COVID-19 [18].

Motivado por todos estos factores, este proyecto propone una CNN basada en MobileNet para la clasificación de CXT. Actualmente disponemos de una mayor cantidad de bases de datos con gran variedad de imágenes de CXT. Se utilizarán bases de datos de distintas fuentes con el fin de crear un *dataset* lo más equilibrado posible. Además, se aplicarán técnicas de *data-augmentation* en la fase de entrenamiento con el fin de proporcionar más variedad a las imágenes y técnicas de modificación de imagen para el preprocesamiento de estas con el fin de obtener mejores resultados con tiempos de ejecución bajos.

# Experimentación con *Deep Learning*

## Introducción

La motivación principal detrás del desarrollo del modelo propuesto es diferenciar automáticamente entre una persona que padece COVID-19, una persona que padece una neumonía de otro tipo y una persona sana, ayudando de esta forma al personal sanitario. En este capítulo se explicará el proceso llevado a cabo para elaborar este modelo. Desde la recolección de los datos en *dataset* públicos, pasando por el preprocesamiento que se realiza a las imágenes, hasta los distintos experimentos en la creación de la red neuronal para finalmente llegar al modelo final utilizado.

La primera aproximación que se hizo fue la de realizar una red neuronal convolucional desde cero. Sin embargo, tras consultar distintos artículos -véase Trabajos relacionados- y tras comprobar que los resultados obtenidos en esta red eran muy negativos, se decidió utilizar otras técnicas, concretamente *transfer learning*.

## *Dataset*

Hasta hace poco tiempo, encontrar un *dataset* con imágenes de calidad era complicado. Por un lado, teníamos muy pocas imágenes de COVID-19, y por otro lado las pocas que son públicas son de casos muy graves, con lo que realmente no aportaban información suficiente dado el objetivo del proyecto.

Por ello, además de aplicar técnicas de *data-augmentation* para conseguir más variedad, se seleccionaron distintos *dataset* y se aunaron para asegurarse de que se disponía de suficientes datos, balanceados y de calidad.

Los *dataset* utilizados fueron los siguientes:

* [ieee8023/covid-chestxray-dataset: We are building an open database of COVID-19 cases with chest X-ray or CT images. (github.com)](https://github.com/ieee8023/covid-chestxray-dataset) : *Dataset* completo en el que podemos encontrar imágenes tanto de COVID-19 como de otro tipo de neumonías.
* [Chest X-Ray Images (Pneumonia) | Kaggle](https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia) : *Dataset* de 3 años de antigüedad, apropiado para la obtención de imágenes de personas sin signos clínicos asociados a neumonía.
* [COVID-19 Radiography Database | Kaggle](https://www.kaggle.com/tawsifurrahman/covid19-radiography-database) : *Dataset* completo y variado con 4 categorías distintas de neumonía.

Para la elección de las imágenes conté con la ayuda de personal sanitario, que me asesoró para escoger la mejor variedad posible. En total se utilizaron 10.245 imágenes.

Una vez obtenidas todas las imágenes, se separaron en *train,* *test* y *val* para su posterior uso.

## Pre procesamiento

Antes de utilizar las imágenes para entrenar, son sometidas a un proceso para destacar más los blancos y oscurecer los negros. Esta técnica recibe el nombre de *Fuzzy Color Image enhancement* [19]*.* Mejora la calidad de la imagen y reduce el ruido de esta. Se realizaron dos experimentos para comprobar la eficacia del método y este aumentaba en un 1.5% de precisión al entrenamiento con las imágenes sin modificar.

Podemos observar una comparación de cómo quedan las imágenes en la Figura 3 , a la izquierda la foto original.

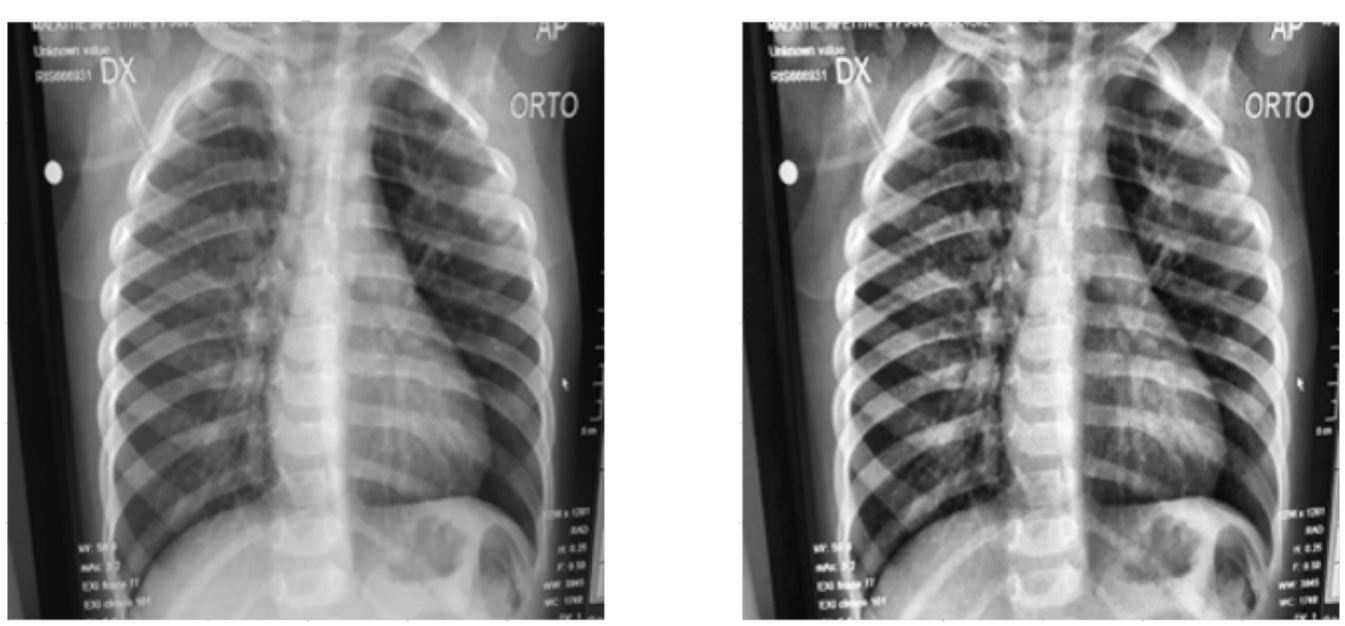


Figura 1 Comparación técnica *Fuzzy*

## Preparación de los experimentos

Basándome en lo leído en otros artículos, véase Trabajos relacionados, seleccionamos algunas redes que ofrecían buen rendimiento. Estas fueron VGG16, ImageNet y InceptionV3.

Todas las pruebas que se realizaron fueron llevadas a cabo con un *dataset* reducido, con el fin de agilizar el proceso. Además, a todas las redes se les aplicó *fine tunning* con el fin de adaptarlas a las necesidades del problema, añadiendo algunas capas a la red.

Podemos ver estas capas en Figura 3.

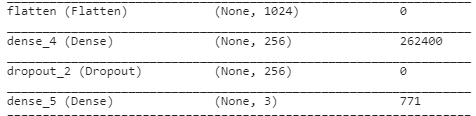


Figura 2 Capas añadidas a la red CIDANet

La capa de *dropout* fue añadida con el fin de evitar sobreajuste en las predicciones.

La última capa es una capa densa de 3 salidas, con una activación *softmax* para obtener las predicciones en forma de probabilidad.

Todos los modelos de las pruebas fueron importados con los pesos extraídos de la base de datos de ImageNet.

Como optimizador para las redes escogimos Adam, por su buen rendimiento con problemas de clasificación de imagen [20]. En cuanto a la función de perdida, utilizamos *categorical\_crossentropy* al tratarse de un problema de clasificación en 3 clases.

Como métrica, nos centramos en *accuracy*, que es la que queremos optimizar y, además, definimos una serie de *callbacks* para controlar el entrenamiento.

Por otro lado, para controlar el balanceo de las clases utilizamos una librería para calcular la ponderación de los pesos y con ello evitar ese desbalanceo.

Todas las pruebas fueran llevadas a cabo con 10 *epochs*.

## Experimento 1: MobileNet

MobileNet [21] es un modelo de red convolucional propuesto por Google, y especialmente diseñado para trabajar en entornos con poca capacidad computacional.

Podemos ver una imagen de la arquitectura en Figura 3 .

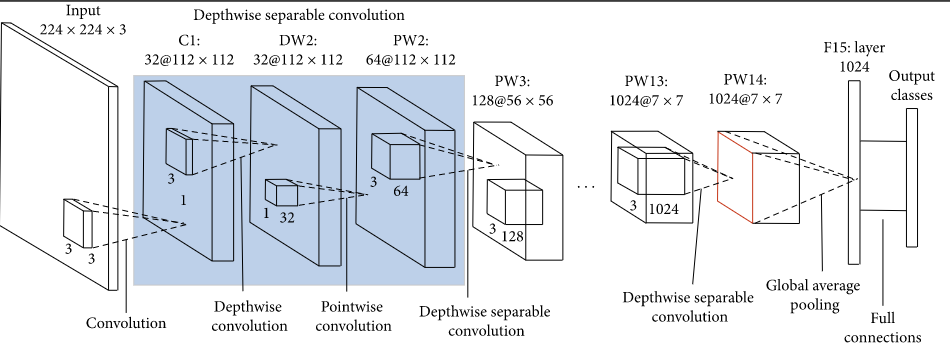


Figura 3 Arquitectura MobileNet

Tras entrenarla con el *dataset* reducido, obtuvimos un 85,29% de *accuracy*.

Lo más destacable del entrenamiento de esta red, fue su reducido tiempo de entrenamiento (Figura 5), y sus buenas estadísticas para ser entrenado con tan pocas iteraciones (Figura 4).

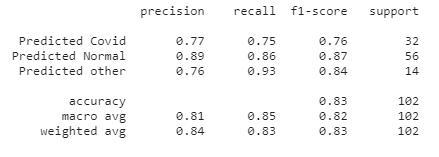


Figura 4 Estadísticas MobileNet

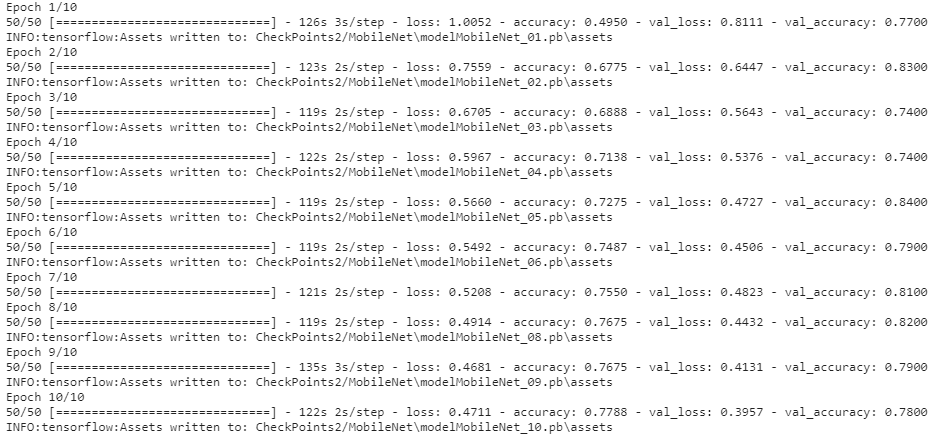


Figura 5 Epochs MobileNet

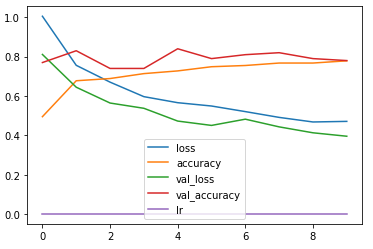


Figura 6 Epoch-accuracy MobileNet

## Experimento 2: Red InceptionV3

InceptionV3 [22] es una red convolucional muy utilizada en análisis de imagen y detección de objetos y que comenzó como un modulo de otra red denominada GoogleNet.

Se trata de una red muy profunda diseñada con tecnología punta de *Deep Learning*. Podemos observar una imagen de la arquitectura en Figura 7.



Figura 7 Arquitectura InceptionV3

Las pruebas con esta red fueron muy prometedoras, obteniendo un 90,19% de *accuracy*.

Tanto los tiempos (Figura 9) como las estadísticas (Figura 8) fueron sorprendentes por lo que este modelo parecía un buen candidato para convertirse en el modelo final.

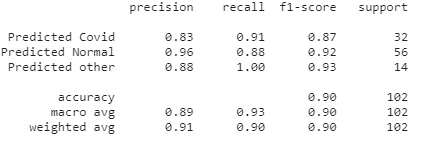


Figura 8 Estadísticas InceptionV3

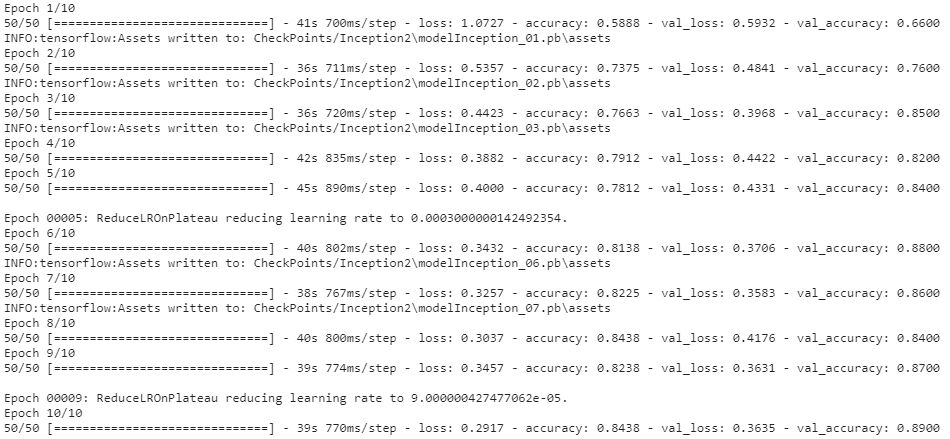


Figura 9 Tiempos InceptionV3

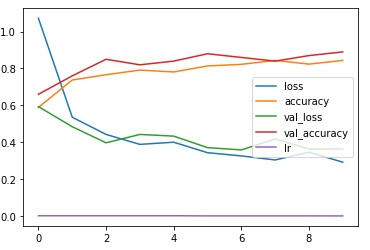


Figura 10 Epoch-accuracy InceptionV3

## Experimento 3: Red VGG16

La red VGG16 [23] es una red convolucional propuesta por K. Simonyan y A. Zisserman de la universidad de Oxford. El modelo obtuvo un 92.7% de precisión en ImageNet, un *dataset* de alrededor de 14 millones de imágenes que pertenecen a 1000 clases.

Podemos ver una imagen de la arquitectura en Figura 11 .



Figura 11 Arquitectura VGG16

Lo primero que se hizo fue realizar una prueba de rendimiento sobre ese *dataset* reducido que habíamos creado.

La evaluación sobre el conjunto de prueba fue de un 83,33% de *accuracy*.

A pesar de los buenos resultados (Figura 3), parecía que el modelo era muy lento de entrenar (Figura 4), por lo que debido a la capacidad computacional de la maquina en la que se estaba entrenando el modelo, se decidió descartar este modelo para la prueba final sobre el *dataset* completo pues encontrábamos otros como ImageNet, con una *accuracy* similar, pero con un tiempo mucho más reducido.

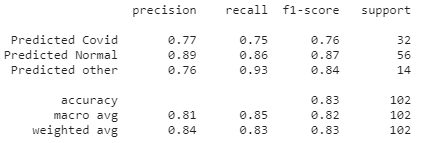


Figura 12 Estadísticas VGG16

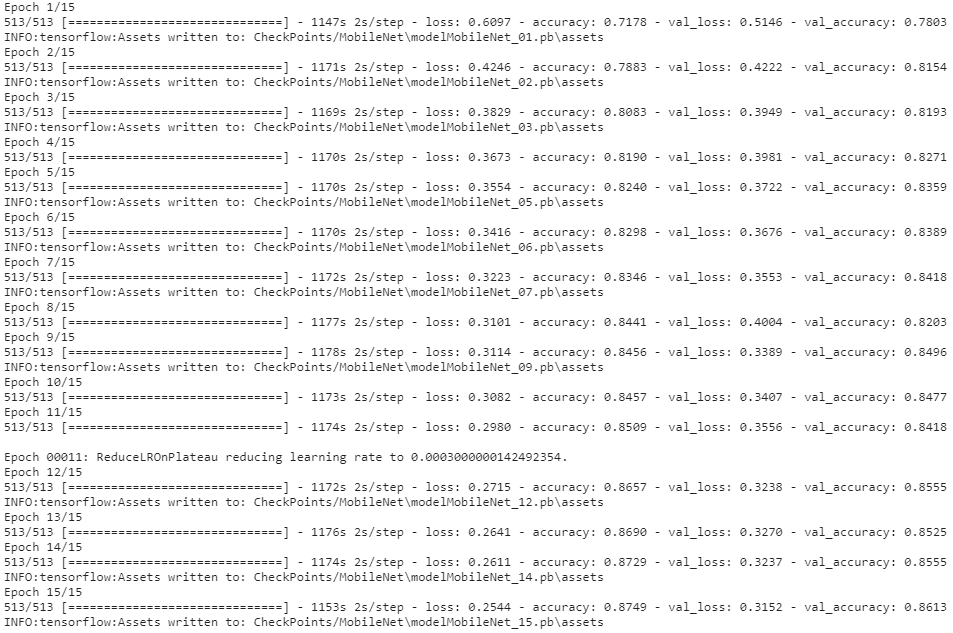


Figura 13 Tiempos VGG16

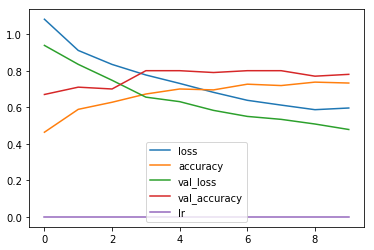


Figura 14 Evolución epoch-accuracy VGG16

## Experimento 4: MobileNet vs InceptionV3

Una vez obtenidos y comparados los datos de los 3 experimentos, visibles en Tabla 1, y apoyándonos en ellos, decidimos descartar el modelo VGG16. Además de tener un *accuracy* similar a la red MobileNet, sus tiempos de entrenamiento eran casi 10 veces superiores a la red MobileNet.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modelo |  | accuracy | recall | F1-Score |
| InceptionV3 |  | 89% | 93% | 90% |
| MobileNet |  | 84% | 85% | 83% |
| VGG16 |  | 81% | 85% | 83% |

Tabla 1 Comparación modelos dataset reducido

Una vez llegado a estos dos modelos, quedaba seleccionar uno, por lo que se repitieron las pruebas de los experimentos anteriores, pero, en este caso, se utilizó el *dataset* completo y se lanzaron 50 *epochs*.

El primero modelo en lanzarse fue el de InceptionV3. A pesar de haber obtenido tan buenos resultados en la fase de experimentación previa, obtuvo un porcentaje de acierto en el test del 88,99%. A pesar de ser buenos resultados, tenía una precisión limitada a la hora de predecir los casos de COVID-19 (Figura 16), y la evolución del *accuracy* con respecto al número de *epochs* parecía descender (Figura 15).

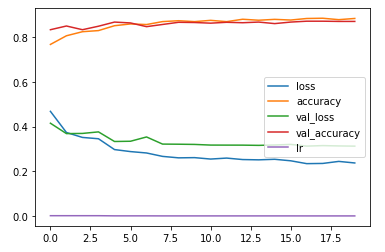


Figura 15 Epochs-accuracy InceptionV3 FullDataset

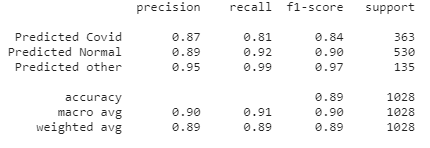


Figura 16 Estadísticas InceptionV3 FullDataset

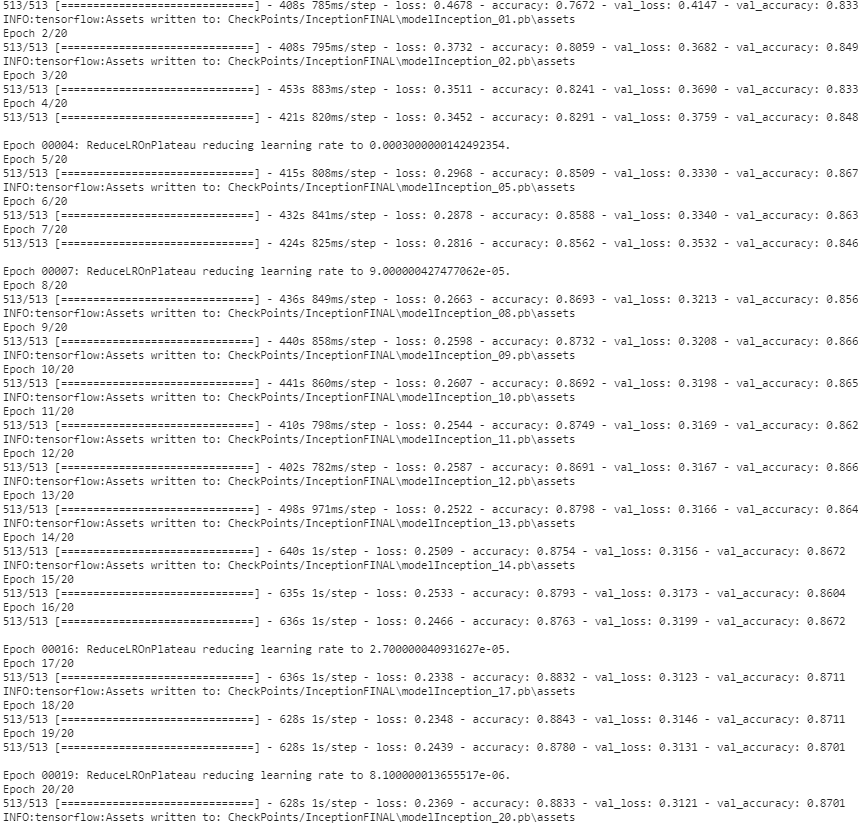


Figura 17 Velocidad InceptionV3 FullDataset

En cuanto a MobileNet, se obtuvieron unos sorprendentes resultados con el *dataset* completo, con una *accuracy* del 94,16%.

Con unas muy buenas estadísticas (Figura 18), fue capaz de predecir la mayoría de los casos de COVID (Figura 19) y de otros tipos.

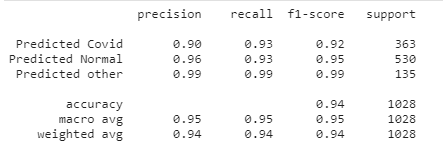


Figura 18 Estadísticas MobileNet FullDataset

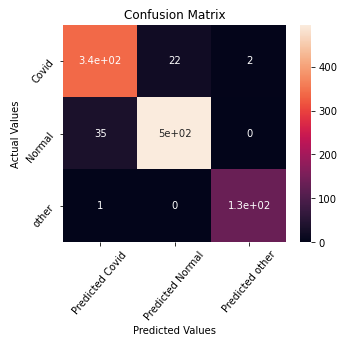


Figura 19 Matriz de confusión MobileNet FullDataset

## Evaluación de los resultados

Tras realizar estos dos últimos experimentos, cuyos resultados podemos observar en Tabla 2, se tomó la decisión de utilizar MobileNet como modelo final.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modelo |  | accuracy | recall | F1-Score |
| InceptionV3 |  | 90% | 91% | 90% |
| MobileNet |  | 95% | 95% | 95% |

Tabla 2 Comparativas experimento final

Se trata de una red liviana, con unos tiempos de entrenamiento bajos, y con un *accuracy* muy elevado.

La Figura 20 nos muestra un resumen del modelo que se propone, CIDANet.

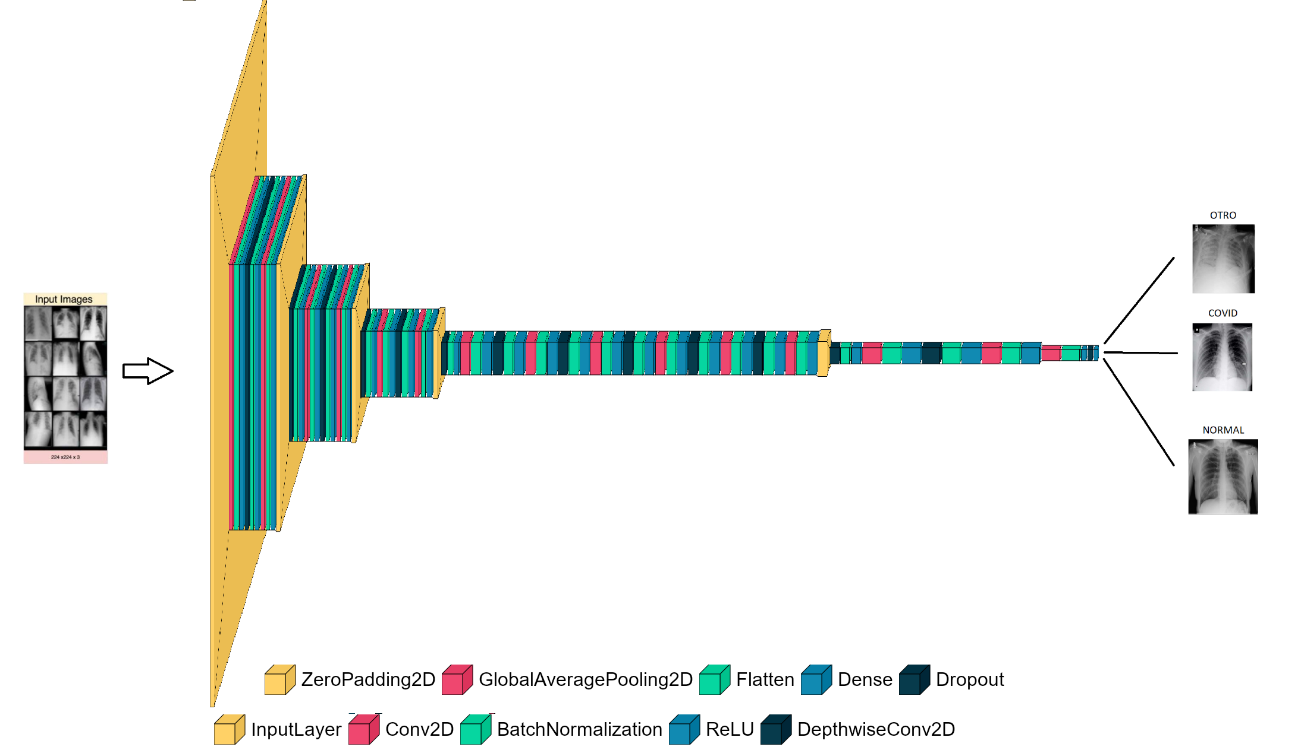


Figura 20 Red CIDANet

La red está diseñada para trabajar con una clasificación en 3 categorías diferentes.

Las imágenes que se introducen tienen un tamaño de 224x224x3

# Despliegue del modelo de *Deep Learning* en servicios en la nube (AWS)

## Introducción.

Uno de los objetivos del proyecto, era poder hacer que fuese público y accesible para todo el mundo. Para ello, se planteó utilizar tecnologías *cloud.* Nuestra opción de distribuidor de servicios en la nube fue Amazon Web Services (AWS).

Para realizar la arquitectura se barajaron diferentes opciones. Una de ellas era la de realizar el despliegue utilizando la herramienta de AWS SageMaker, un servicio de aprendizaje automático que puede utilizar para crear, entrenar e implementar modelos de aprendizaje automático, sin embargo, los altos precios de este servicio para el desarrollo del proyecto nos llevó a desechar esta idea.

Otra de las opciones era la de levantar una máquina de EC2 en la que se realizase el entrenamiento, y acceder directamente a esa máquina para obtener los resultados. Esta opción era interesante pues permitía disponer de la capacidad de cómputo necesaria para hacer los tiempos de entrenamiento más ligeros. Debido a que la elección del modelo fue MobileNet, la capacidad de cómputo no era un problema por lo que también se rechazó esta idea.

Finalmente se planteó un escenario más sencillo, utilizar los servicios de S3 para desplegar una web estática y hacer uso de servicios como Lambda, S3, DynamoDB, APIGateway y ECR. Un entorno *serverless* bajo demanda. Además, puesto que no esperábamos muchas peticiones, el precio no sería elevado.

## Arquitectura Cloud.

La arquitectura *Cloud* planteada, hace uso de distintos servicios de AWS. Podemos ver un resumen de la arquitectura en Figura 21 .

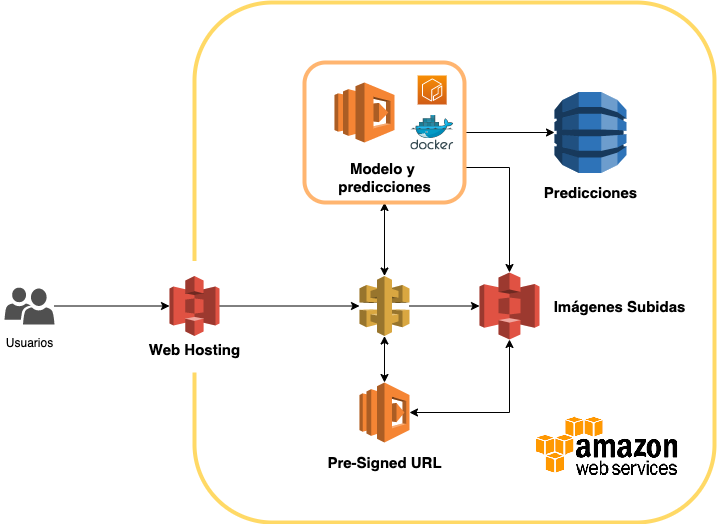


Figura 21 Arquitectura AWS

### S3

Se crearon dos *buckets*, uno para alojar la página web. Fue configurada para alojamiento web estático. Además, se configuró la política del *bucket* para permitir leer objetos y se modificó la configuración CORS, para permitir el uso de métodos PUT y GET.

Por otro lado, se creó un segundo *bucket* en el cual se alojarían las imágenes que subimos para ser analizadas por el algoritmo. De igual forma se configuró la política del *bucket*, el CORS y los permisos.

### DynamoDB

Se creo una tabla denominada ‘’predictions-covid’’. En esta se almacenan los valores de predicción de la imagen, junto con un *photoid*, que corresponde con el nombre del fichero guardado en S3.

### ECR

Utilizamos ECR para subir una imagen de *Docker* con las dependencias necesarias para realizar la predicción. Elegimos cargar el modelo en Docker por las limitaciones de lambda [24].

Creamos la imagen con 4 archivos: código app.py, DockerFile, requirements.txt y finalmente el modelo, en formato .pb, que generamos previamente.

### APIGateway

Se creo una API REST para realizar dos llamadas. Por un lado, tenemos una llamada denominada *upload-photo*, conectada con la lambda *upload-photo*, que se encarga de obtener una url firmada para que la web pueda subir el archivo a S3.

Una vez que la imagen se encuentra subida en S3, se lanza una segunda llamada denominada *get-prediction*, conectada a la lambda con el mismo nombre, para devolver las predicciones, mostrarlas por pantalla y almacenar los resultados en DynamoDB.

### Lambda

Se crearon dos lambdas, una para obtener la url *pre-signed* y otra con la imagen cargada de ECR para realizar la predicción y la subida a DynamoDB.

Usamos la *pre-signed* url en lugar de enviar la imagen por api gateway con el fin de evitar el límite de 15MB de datos de entrada que tiene api Gateway. De esta forma enviamos la foto directamente a S3

## Página web

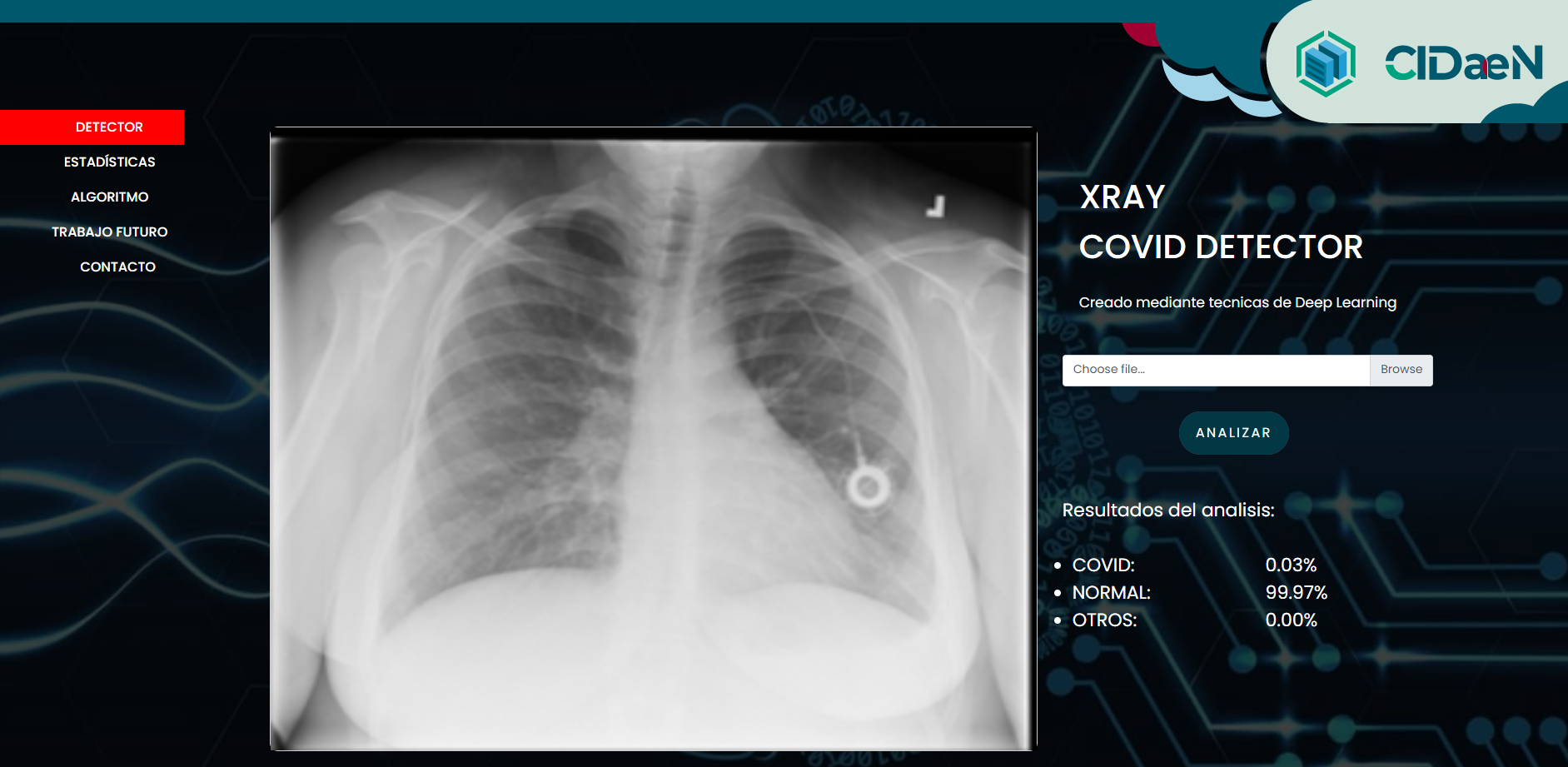
La creación de la web fue concebida tomando como base una plantilla html. A partir de ahí, se fueron modificando el html, el js y el css para adaptarla a las necesidades del proyecto. Los códigos de la página serán adjuntados junto con la memoria del proyecto.

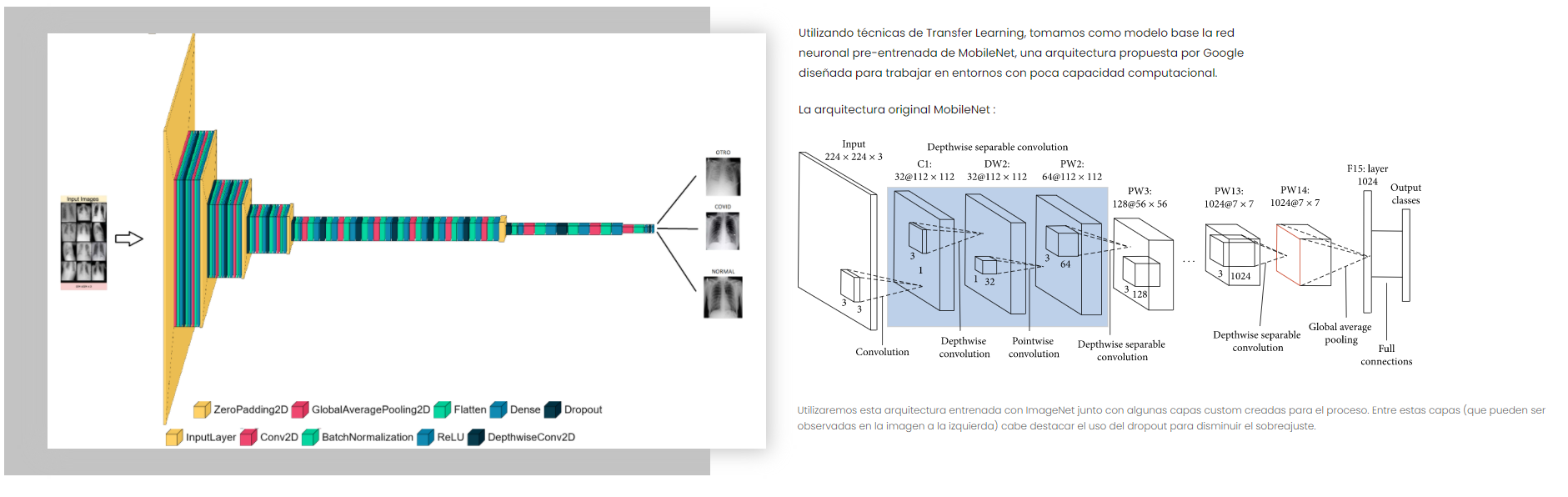
La web es accesible de manera pública a través del siguiente enlace:

<http://web-tfm.s3-website-us-east-1.amazonaws.com/>

El funcionamiento es sencillo, solo debe subirse una imagen de una radiografía y darle al botón analizar. En un tiempo medio de 5 segundos obtendremos la predicción.

Podemos observar algunas imágenes de la página web en Figura 22.





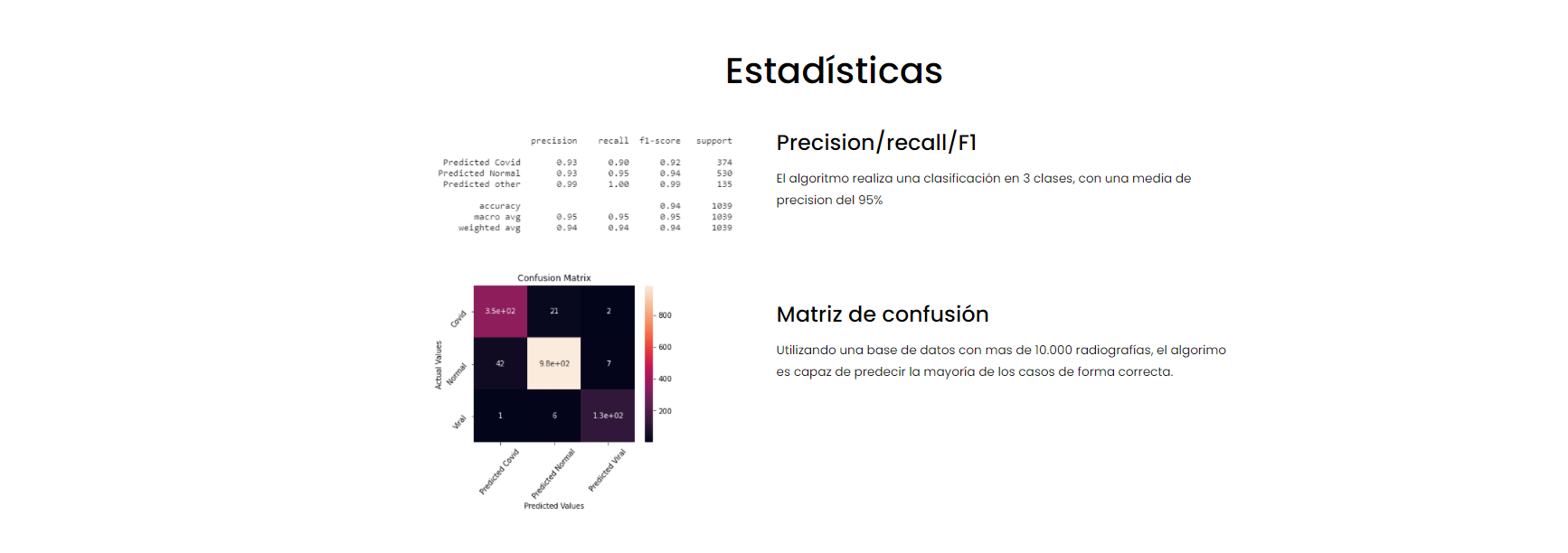


Figura 22 Página Web

# Conclusiones y Trabajo Futuro

## Conclusiones

Mediante este proyecto, se ha tratado de elaborar una herramienta para ayudar al personal sanitario no especializado a identificar una posible neumonía por COVID-19 en radiografías de tórax.

Se ha creado un modelo robusto que funciona con un 95% de precisión en la clasificación en tres categorías. No se trata de una herramienta que sustituya la labor del personal sanitario, sino una herramienta para apoyarles en la toma de decisiones.

## Trabajo futuro

El campo del *Deep Learning* es un campo muy amplio que no para de crecer a cada instante. Las líneas en las que puede seguir el trabajo por tanto son amplias, sin embargo, cabe destacar algunas de ellas:

* Explicabilidad de las imágenes: En el proyecto, se ha intentado aplicar técnicas LIME y GRADCAM para mejorar la explicabilidad de las imágenes. Estas técnicas utilizan el gradiente de la última capa de convolución para elaborar un mapa de calor que explica las áreas en las que el modelo está mirando para hacer la predicción. Sin embargo, puesto que los valores que ofrecían no eran del todo precisos, se ha decidido apartarlo del proyecto en sí y dejarlo como un proceso a mejorar. Se pueden ver algunas imágenes de estos intentos en Figura 23 LIME y GRADCAM .



Figura 23 LIME y GRADCAM

* Ensemble de modelos: Es una técnica en la que utilizamos las redes neuronales como submodelos para realizar una primera clasificación y luego utilizar esas predicciones como entrada para otra red neuronal conocida como *estimate-learner* [25]. El principal inconveniente de esta técnica es la capacidad de cómputo. Por cuestiones logísticas se decidió dejar esta técnica para una línea futura.

Bibliografía

Alexa Internet Inc. (2014). The top 500 sites on the web. Retrieved February 11, 2014, from http://www.alexa.com/topsites

Anda, B., Sjøberg, D., & Jørgensen, M. (2001). Quality and Understandability of Use Case Models. In 15th European Conference on Object-Oriented Programming (ECOOP’01) (pp. 402–428). London, UK: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45337-7\_21

Babar, M. A., Kitchenham, B. A., Zhu, L., Gorton, I., & Jeffery, R. (2006). An empirical study of groupware support for distributed software architecture evaluation process. Journal of Systems and Software, 79(7), 912–925. https://doi.org/10.1016/j.jss.2005.06.043

Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). The Goal Question Metric Approach. In Encyclopedia of Software Engineering (Vol. 2, pp. 528–532). Wiley. Retrieved from http://wwwagse-old.informatik.uni-kl.de/pubs/repository/basili94b/encyclo.gqm.pdf

Basili, V. R., Shull, F., & Lanubile, F. (1999). Building Knowledge through Families of Experiments. IEEE Transactions on Software Engineering, 25(4), 456–473. https://doi.org/10.1023/A:1009742216007

Bereiter, C. (2002). Education and Mind in the Knowledge Age (1st ed.). Routledge. Retrieved from http://www.amazon.com/Education-Mind-Knowledge-Carl-Bereiter/dp/0805839437

Biostat Inc. (2006). Comprehensive Meta-Analysis. Retrieved April 17, 2013, from http://www.meta-analysis.com/

Booch, G., Rumbaugh, J., & Jacobson, I. (2005). The Unified Modeling Language User Guide (2nd ed.). Addison-Wesley Professional.

Canfora, G., Cimitile, A., Garcia, F., Piattini, M., & Visaggio, C. A. (2006). Evaluating advantages of test driven development. In 2006 ACM/IEEE international symposium on International symposium on empirical software engineering (ISESE’06) (p. 364). Rio de Janeiro, Brazil: ACM Press. https://doi.org/10.1145/1159733.1159788

Castro, J., Kolp, M., & Mylopoulos, J. (2001). A requirements-driven development methodology. In 13th Int. Conf. On Advanced Information Systems Engineering (CAiSE’01) (pp. 108–123). London, UK: Springer-Verlag. https://doi.org/10.1007/3-540-45341-5\_8

Celko, J., Davis, J. S., & Mitchell, J. (1983). A demonstration of three requirements language systems. ACM SIGPLAN Notices, 18(1), 9–14. https://doi.org/10.1145/948093.948094

Cockburn, A. (2000). Writing Effective Use Cases (1st ed.). Addison-Wesley Professional.

Cruz-Lemus, J. A., Genero, M., Caivano, D., Abrahão, S., Insfrán, E., & Carsí, J. A. (2011). Assessing the influence of stereotypes on the comprehension of UML sequence diagrams: A family of experiments. Information and Software Technology, 53(12), 1391–1403. https://doi.org/10.1016/j.infsof.2011.07.002

Cruz-Lemus, J. A., Genero, M., Manso, M. E., Morasca, S., & Piattini, M. (2009). Assessing the understandability of UML statechart diagrams with composite states—A family of empirical studies. Empirical Software Engineering, 14(6), 685–719. https://doi.org/10.1007/s10664-009-9106-z

Cruz-Lemus, J. A., Maes, A., Genero, M., Poels, G., & Piattini, M. (2010). The impact of structural complexity on the understandability of UML statechart diagrams. Information Sciences, 180(11), 2209–2220. <https://doi.org/10.1016/j.ins.2010.01.026>.

Cysneiros, L. M., & Yu, E. S.-K. (2004). Non-Functional Requirements Elicitation. In J. C. S. do Prado Leite & J. H. Doorn (Eds.), Perspectives on Software Requirements (pp. 115–138). Springer US. https://doi.org/10.1007/978-1-4615-0465-8\_6

Damian, D. (2001). An empirical study of requirements engineering in distributed software projects: is distance negotiation more effective? In 8th Asia-Pacific Software Engineering Conference (APSEC’01) (pp. 149–152). Macao, China: IEEE Comput. Soc. https://doi.org/10.1109/APSEC.2001.991471

De Lucia, A., Fasano, F., Oliveto, R., & Tortora, G. (2006). Can Information Retrieval Techniques Effectively Support Traceability Link Recovery? In 14th IEEE International Conference on Program Comprehension (ICPC’06) (pp. 307–316). Athens, Greece: IEEE. https://doi.org/10.1109/ICPC.2006.15

Dieste, O., Fernández, E., García Martínez, R., & Juristo, N. (2011). Comparative analysis of meta-analysis methods: when to use which? In 15th International Conference on Evaluation & Assessment in Software Engineering (EASE’11) (pp. 36–45). Durham, UK: IET.

Dourish, P., & Bellotti, V. (1992). Awareness and coordination in shared workspaces. In ACM conference on Computer-supported cooperative work (CSCW’92) (pp. 107–114). Toronto, Canada: ACM Press. <https://doi.org/10.1145/143457.143468>