



Trabajo Fin de Máster:

**SIIM-FISABIO-RSNA COVID-19 Detection
Detección anomalías en radiografías de tórax**

Aránzazu García Boronat

Tutor: Pablo Ramos Criado

Master Big Data & Data Science

Octubre 2021

INDICE

1.	INTRODUCCIÓN	4
2.	ESTADO DE LA CUESTION	7
2.1.	REDES DE NEURONAS ARTIFICIALES.....	7
2.1.1.	Estructura Básica de una Red Neuronal.....	8
2.1.2.	Entrenamiento de Redes de Neuronas Artificiales	12
2.2.	REDES DE NEURONAS CONVOLUCIONALES	21
2.2.1.	Estructura y propiedades de las CNN	21
2.2.2.	La transferencia de aprendizaje.....	23
2.2.3.	Xception.....	23
2.3.1.	Otras Arquitecturas de Redes Neuronales Convolucionales.....	25
2.3.	CLASIFICACIÓN AUTOMATIZADA DE ANOMALÍAS DE LAS RADIOGRAFÍAS DE TÓRAX CON EL USO DE REDES CONVOLUCIONALES PROFUNDAS.	28
2.3.1.	Descripción del estudio realizado.....	30
2.3.2.	Conclusiones	36
3.	PLATEAMIENTO DEL PROBLEMA	38
3.1.	DEFINICIÓN DEL PROBLEMA Y OBJETIVOS	39
3.1.1.	Objetivos	43
3.2.	HERRAMIENTAS	43
3.2.1.	Entorno de trabajo	43
3.2.2.	Lenguaje de programación	44
3.2.3.	Entorno de Programación	44
4.	DESARROLLO.....	45
4.1	Conjunto de datos	46
4.1.1.	SIIM-FISABIO-RSNA COVID-19 Detection.....	46
4.1.2.	SIIM COVID-19: Resized to 512px JPG.....	47
4.1.3.	COVID-19 Radiography Database.....	48
4.1.4.	SIIM COVID-19: Resized to 256px PNG.....	48
4.2	Almacenamiento de los datos.....	49
4.3	Cargar los datos (Ficheros csv, carpetas ficheros de imágenes) Análisis exploratorio y preparación de los datos	51
4.3.1.	Solo con los conjuntos de datos aportados por la Competición	51
4.3.2.	Unión de bases de datos de los 4 conjuntos de datos presentados en el punto 4.1 Conjunto de datos de la Memoria del Trabajo Fin de Máster.	58
4.4	Visualización imágenes train con cajas que identifican las anomalías.....	63
4.5	Preprocesamiento de los datos	65
4.6	Aumento y Partición de datos.....	67
4.6.1.	Aumento de datos:.....	67

4.6.2.	Partición de los datos:	67
4.7	Creación de Modelos	69
4.7.1.	Modelo con red neuronal convolucional de 9 capas con aumento de datos	73
4.7.2.	Modelo con transfer learning – cargamos Xception, con aumento de datos	75
4.7.3.	Modelo con transfer learning Xception con datos de entrada y etiqueta de salida convertidos a un vector, con aumento de datos	77
4.7.4.	Modelo con transfer learning Xception con datos de entrada y etiqueta de salida convertidos a un vector, SIN aumento de datos.....	79
4.7.5.	Modelo con transfer learning Xception (DESDE CERO) con datos de entrada y etiqueta de salida convertidos a un vector, con aumento de datos..	80
4.7.6.	Modelo con transfer learning – cargamos VGG16, con aumento de datos	81
4.7.7.	Modelo con transfer learning – cargamos Inception V3, con aumento de datos	82
4.7.8.	Modelo con transfer learning – cargamos Xception, con aumento de datos – 3 etiquetas de salida	84
4.7.9.	Modelo con transfer learning – cargamos Xception, con aumento de datos – 2 etiquetas de salida: Con anomalía versus sin anomalía	86
4.7.10.	Modelo con transfer learning – cargamos Xception, con aumento de datos – 2 etiquetas de salida: Con neumonía de Covid-19 típica versus sin ella.	88
4.7.11.	Modelo con transfer learning – cargamos InceptionV3, con aumento de datos – 2 etiquetas de salida: Con neumonía de Covid-19 típica versus sin ella.	90
4.7.12.	Conclusión tras la ejecución de los 12 modelos anteriores	92
4.8	Uso Conjunto de datos externo	93
4.9	Modelos con la unión de los conjuntos de datos: Modelo Xception con Data Augmentation – Clasificación binaria (con opacidad/con anomalía y sin opacidad/sin anomalía).	97
4.10	Mejora de Resultados. Modelo con modelo preentrenado aplicado a solo datos de la competición inicial	103
4.11	Visualización de los Mejores Resultados	105
4.11.1.	Gráficas con la evolución de las métricas de “accuracy” y “función de pérdidas” tanto en el entrenamiento como en la validación final del subconjunto de test.	105
4.11.2.	Matriz de confusión	106
4.11.3.	Evolución tasa de acierto en los datos de validación Test Final	107
5.	CONCLUSIONES Y LINEAS FUTURAS	108
6.	INDICE DE FIGURAS.....	111
7.	REFERENCIAS	114

1. INTRODUCCIÓN

Título del trabajo:	SIIM-FISABIO-RSNA COVID-19 Detection. Detección de anomalías en radiografías de tórax
Nombre del autor:	Aranza García Boronat
Nombre del Tutor:	Pablo Ramos Criado
Fecha de entrega:	3/10/2021 23:59
Titulación:	Master Big Data & Data Science
Palabras clave:	Anomalías, Aprendizaje profundo, Rayos-X
Resumen del Trabajo	
Descripción del problema:	
<p>Cinco veces más mortal que la gripe, COVID-19 causa una morbilidad y mortalidad significativas. Al igual que otras neumonías, la infección pulmonar con COVID-19 produce inflamación y líquido en los pulmones. COVID-19 se parece mucho a otras neumonías virales y bacterianas en las radiografías de tórax, lo que dificulta su diagnóstico. El modelo de visión por computadora para detectar y localizar COVID-19 ayudaría a los médicos a proporcionar un diagnóstico rápido y confiable. Como resultado, los pacientes podrían recibir el tratamiento adecuado antes de que se produzcan los efectos más graves del virus.</p> <p>Actualmente, neumonía por COVID-19 se puede diagnosticar mediante la reacción en cadena de la polimerasa para detectar material genético del virus o una radiografía de tórax. Sin embargo, pueden pasar algunas horas y, a veces, días antes de que se obtengan los resultados de las pruebas moleculares. Por el contrario, las radiografías de tórax se pueden obtener en minutos.</p> <p>Si bien existen pautas para ayudar a los radiólogos a diferenciar la neumonía por COVID-19 de otros tipos de infección, sus evaluaciones varían. Además, los no radiólogos podrían recibir apoyo con este modelo de detección.</p> <p>La misión de la Sociedad de Informática por Imágenes en Medicina (SIIM) es promover la informática de imágenes médicas a través de la educación, la investigación y la innovación. SIIM se ha asociado con la Fundación para el Fomento de la Salud y la Investigación Biomédica de la Comunitat Valenciana (FISABIO), el Banco de Datos de Imágenes Médicas de la Comunitat Valenciana (BIMCV) y la Sociedad Radiológica de Norteamérica (RSNA) para este concurso.</p>	



La Sociedad de Informática por Imágenes en Medicina (SIIM) es quien ha promovido una competición en Kaggle: (1) , **de la cual he sido participante.**

Lo que pretende la competición es ayudar a los radiólogos a diagnosticar millones de pacientes con neumonía por COVID-19 con mayor confianza y rapidez. Esto también permitirá a los médicos ver el alcance de la enfermedad y les ayudará a tomar decisiones con respecto al tratamiento. Dependiendo de la gravedad, los pacientes afectados pueden necesitar hospitalización, ingreso en una unidad de cuidados intensivos o terapias de apoyo como ventilación mecánica. Como resultado de un mejor diagnóstico, más pacientes recibirán rápidamente la mejor atención para su afección, lo que podría mitigar los efectos más graves del virus.

Objetivos del Trabajo:

El objetivo en este trabajo es crear un modelo de clasificación binario que detecte una radiografía con patología /opacidad de una radiografía sin patología / sin opacidad por ello se realizará la agrupación (típica, atípica e indeterminada) versus negativa.

La radiografía de tórax es el examen radiológico más solicitado por su eficacia en la caracterización y detección de anomalías cardiorácicas y pulmonares. Se usa tanto en la prevención como en la detección de patologías.

Lo que se necesitaría es que el radiólogo informe a tiempo de cada imagen, pero no siempre es posible, debido a la gran carga de trabajo en muchos grandes centros de salud o también la falta de radiólogos experimentados en áreas menos desarrolladas.

Por lo que disponer de, un sistema automatizado de clasificación de anomalías en la radiografía de tórax sería muy importante ya que permitiría a los radiólogos poder centrarse más en la evaluación de aquellas radiografías con anomalías (patologías de tórax).

En este trabajo se pretende crear el modelo de clasificación y la visualización de los resultados de este.

Restricciones y limitaciones del trabajo:

Restricciones:

Divido este apartado en dos partes:

- Concurso de Kaggel: Las restricciones del concurso eran diversas entre las que destacan:
 - ✓ No se permite el uso de internet para la publicación de resultado lo que implica tener que usar los modelos y recursos cargados en el propio portal de Kaggel.
 - ✓ Uso limitado mensual de GPU y memoria RAM, principalmente afecta el uso de GPU para este tipo de trabajos.
 - ✓ No se permite el uso de bases de datos de otras competiciones.
 - Mi Trabajo publicado en Kaggel: (2)
- Para la presentación en U-TAD de este proyecto no existen estas restricciones.

Limitaciones:

La principal limitación que tiene el trabajo son los datos ofrecidos para la resolución del trabajo. Son pocos y además las clases están muy desbalanceadas.

Otra limitación son las herramientas utilizadas con límites computacionales para el desarrollo de un buen rendimiento en un modelo de una red neuronal convolucional.

Los mejores resultados se podrían obtener con el uso de conjunto de datos externo, pero existe la limitación de que el nuevo conjunto de datos de entrenamiento sea representativo del conjunto de producción.

La limitación de tiempo en la entrega de TFM hace que no se pueda desarrollar la obtención y computación de un nuevo conjunto de datos externos de mayor dimensión que el ya utilizado.

2. ESTADO DE LA CUESTION

En este apartado vamos a realizar una descripción de lo que son las redes de neuronas artificiales, se detallará cuál es su estructura básica, así como la forma en la que se realiza el entrenamiento. También se hablará de redes de neuronas convolucionales, que son claves en la solución de este trabajo. Se concluye este apartado con la síntesis de los aspectos más destacados de un artículo médico que trata del uso de las redes de neuronas convolucionales en la clasificación automatizada de anomalías de las radiografías de tórax.

2.1. REDES DE NEURONAS ARTIFICIALES.

“El cerebro humano” (3) es el sistema de cálculo más complejo conocido por el hombre. Los ordenadores y los humanos pueden realizar bien diferentes tipos de tareas, por lo tanto, la operación de reconocer rostros humanos es una tarea relativamente simple para los humanos, pero es más difícil para las computadoras, mientras que la contabilidad empresarial es para los expertos en contabilidad, una tarea difícil. Las tareas costosas son rutinas simples para computadoras básicas.

La capacidad del cerebro humano para pensar, recordar y resolver problemas ha inspirado a muchos científicos a intentar simular la forma en que funciona el cerebro humano en las computadoras a través de la creación de modelos.

Debido al potencial que ofrece esta tecnología, profesionales de diferentes campos como la ingeniería, la filosofía, la fisiología y la psicología se han unido y encontrado diferentes aplicaciones en sus respectivas profesiones.

Un grupo de investigadores ha estado trabajando en la creación de un modelo en una computadora para igualar o adoptar diferentes funciones básicas del cerebro. El resultado es una nueva tecnología llamada computación neuronal o redes neuronales artificiales.

Este nuevo método de cálculo ha recuperado interés después de varios años de estar olvidado debido al extraordinario progreso y éxito en la teoría y la aplicación en los últimos años.

“La Red Neuronal Artificial”, ANN (Artificial Neural Network) (3) está inspirada en la red neuronal biológica del cerebro humano. Está compuesta por elementos que se comportan de forma similar a las neuronas biológicas en las funciones más habituales.

La ANN además de parecerse también presenta características similares. Por ejemplo, las redes neuronales artificiales aprenden de la experiencia, generalizan a partir de ejemplos anteriores y abstraen las principales características de la serie de datos.

- Aprender: Pueden cambiar su comportamiento según el entorno, se les muestra un conjunto de entradas y ellas mismas se ajustan para producir una salida consistente.
- Generalizar: Pueden ofrecer, dentro de un margen, respuestas correctas a entradas que presentan pequeñas variaciones debido a los efectos de ruido o distorsión.
- Abstraer: Algunas son capaces de abstraer la esencia de un conjunto de entradas que en principio no parecen tener aspectos comunes o relativos.

2.1.1. Estructura Básica de una Red Neuronal

La unidad fundamental del sistema nervioso y del cerebro es la “neurona” (3). Cada neurona es una procesadora que recibe y combina señales desde y hacia otras neuronas. Si las entradas son suficientemente fuertes se activa la salida de la neurona.

La Figura (1.1) muestra las partes que constituyen una neurona.

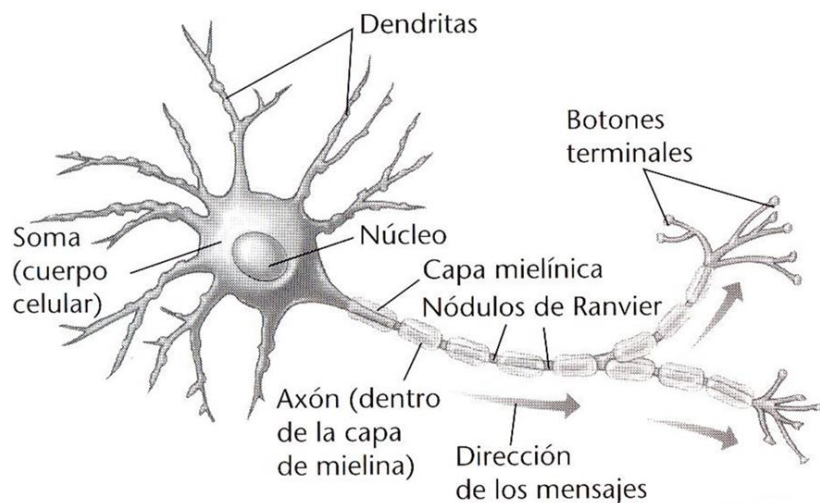


FIGURA 1 - COMPONENTES DE UNA NEURONA (3)

Uno o varios billones de neuronas están densamente conectadas en el cerebro. La salida se conecta a la entrada (axón, dendritas respectivamente) a través de uniones llamadas sinapsis. La eficacia de la sinapsis es modificable durante el proceso de aprendizaje de la red.

El elemento procesador, PE (process element), en las Redes Neuronales Artificiales, ANN, (3) es la unidad análoga a la neurona biológica. Un elemento procesador tiene varias entradas que normalmente combina con

una suma. Esta suma de entradas es modificada por la función de transferencia y la salida resultante es la que pasa al elemento procesador.

La salida del PE se puede conectar a las entradas de otras neuronas artificiales (PE) mediante conexiones ponderadas correspondientes a la eficacia de la sinapsis de las conexiones neuronales.

La Figura (2) representa un elemento procesador de una red neuronal artificial implementada en un ordenador si comparamos con la Neurona biológica.

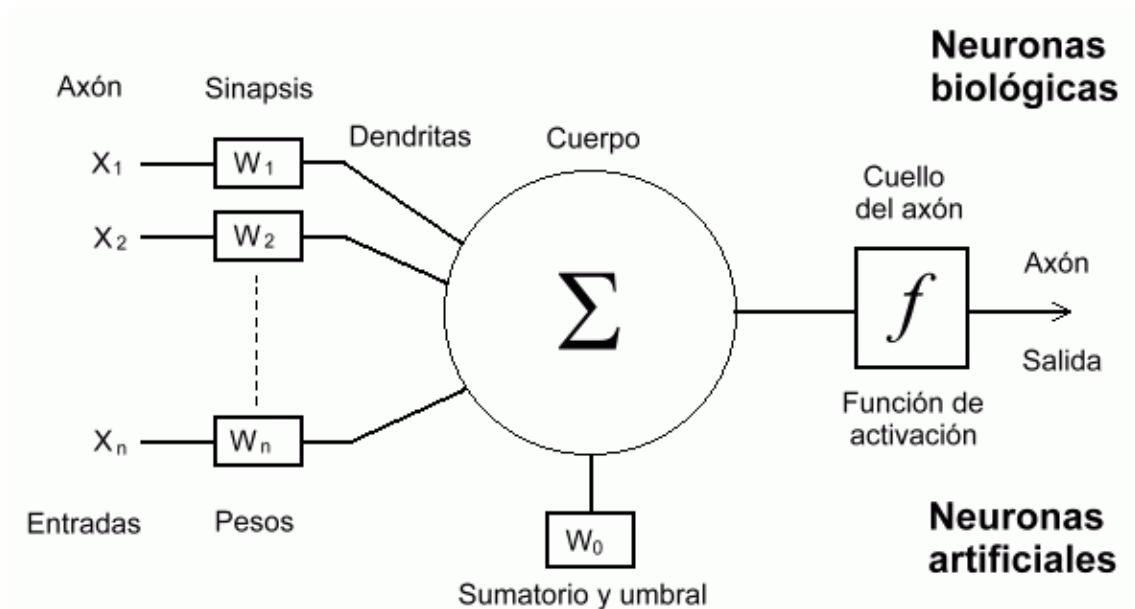


Figura 2. Diagrama de una Neurona Artificial (PE) vs Neurona Biológica (3)

“Los elementos que constituyen una neurona artificial” (4) son:

- **Conjunto de entradas, $x_j(t)$.** Las variables de entrada y salida pueden ser binarias (digitales) o continuas (analógicas) dependiendo del modelo de aplicación.
- **Pesos sinápticos, w_{ij} .** Representan la intensidad de interacción entre cada neurona presináptica j y la neurona postsináptica i . Pueden ser excitadores o inhibidores.
- **Regla de propagación, $o_i(w_{ij}, x_j(t))$.** Integra la información proveniente de las distintas neuronas artificiales y proporciona el valor del potencial postsináptico de la neurona i , en función de sus pesos y entradas. La función más habitual es de tipo lineal, y se basa en una suma ponderada de las entradas con los pesos sinápticos.
- **Función de activación o de transferencia, $f_i(a_i(t-1), h_i(t))$.** Provee el estado de activación actual de la neurona i . En muchos modelos de ANS se considera que el estado actual de la neurona no depende de su estado anterior, sino únicamente del actual.

- **Función de salida, $F_i(a_i(t))$.** Representa la salida actual de la neurona i .

De esta forma, la salida producida por una neurona i , para un determinado instante de tiempo t puede ser escrita en forma general de la siguiente manera

$$y_i(t) = F_i(f_i[a_i(t-1), \sigma_i(w_{ij}, x_j(t))])$$

A continuación, se estudian más en detalle los “dos últimos elementos que constituyen dicha neurona” (5):

Regla de propagación

La regla de propagación establece el potencial resultante de la interacción de la neurona i con las N neuronas vecinas. Este potencial resultante h_i se expresa:

$$h_i(t) = \sigma_i(w_{ij}, x_j(t))$$

La regla de propagación más simple y utilizada consiste en realizar una suma de las entradas ponderadas con sus pesos sinápticos correspondientes:

$$h_i(t) = \sum_j w_{ij} * x_j(t)$$

Función de activación o de transferencia

El estado de activación de la neurona para un determinado instante de tiempo t puede ser expresado de la siguiente manera:

$$a_i(t) = f_i(a_i(t-1), h_i(t))$$

En la mayoría de los modelos se suele ignorar el estado anterior de la neurona, definiéndose de la siguiente manera:

$$a_i(t) = f_i(a_i(h_i(t)))$$

La siguiente imagen muestra las principales funciones de activación o de transferencia utilizadas:

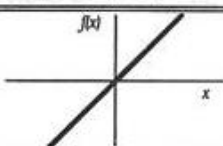
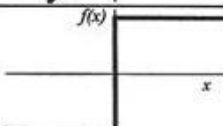
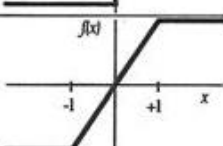
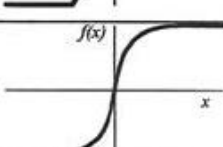
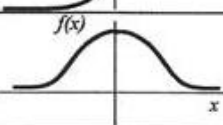
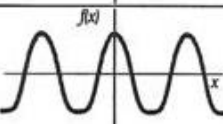
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = \text{sign}(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & \text{si } x < -1 \\ x, & \text{si } -1 \leq x \leq 1 \\ +1, & \text{si } x > 1 \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1 + e^{-x}}$ $y = \text{tgh}(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0, +1]$	
Sinusoidal	$y = A \sin(\omega x + \varphi)$	$[-1, +1]$	

Figura 3. Funciones de activación habituales (4)

Función de salida

La función de salida proporciona el valor de salida de la neurona, en base al estado de activación de la neurona. En general se utiliza la función identidad, es decir:

$$y_i(t) = F_i(a_i(t)) = a_i(t)$$

En la siguiente imagen podemos ver la interconexión entre varias neuronas:

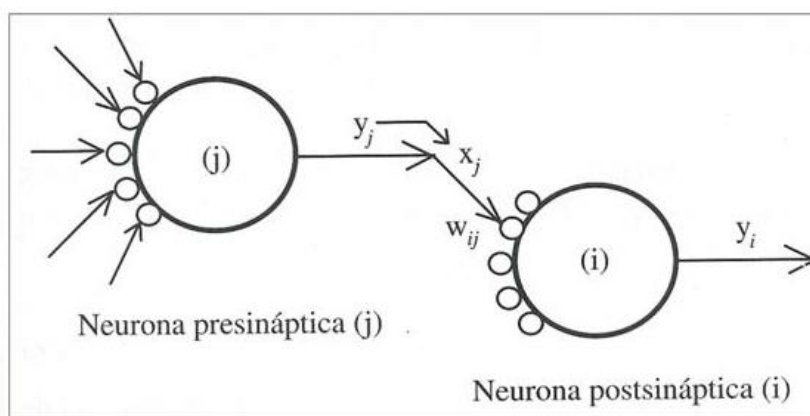


Figura 4. Interconexión entre una neurona presináptica y una neurona postsináptica (4)

2.1.2. Entrenamiento de Redes de Neuronas Artificiales

Existen dos fases claramente diferenciadas en la operatoria de una red neuronal: “la fase de aprendizaje o entrenamiento, y la fase de operación o ejecución” (5).

En la primera fase, la red se entrena para realizar un tipo de procesamiento determinado. Tras alcanzar un entrenamiento adecuado, se pasa a la siguiente fase. En la fase de operación, es donde la red es utilizada para la tarea entrenada.

Fase de entrenamiento.

El principal objetivo que se persigue en “la fase de entrenamiento” (4) es el de establecer unos valores para el vector de los pesos. Con estos pesos se pretende que el error cometido al evaluar el conjunto de ejemplos del entrenamiento sea mínimo.

Se parte de un conjunto de pesos sinápticos aleatorio y el proceso de aprendizaje busca un conjunto de pesos que permitan a la red desarrollar correctamente una determinada tarea. En este proceso se va refinando iterativamente la solución hasta alcanzar un nivel suficientemente bueno.

El proceso de aprendizaje se puede dividir en cuatro grandes grupos de acuerdo con sus características (6), (7):

- **Aprendizaje supervisado.** En este caso hay un conjunto de patrones de entrada y una salida esperada. Los pesos se van modificando de manera proporcional al error que se produce entre la salida real de la red y la salida esperada.
- **Aprendizaje no supervisado.** Sólo hay un conjunto de patrones de entrada. No se dispone de información sobre la salida esperada. El proceso de entrenamiento en este caso deberá ajustar sus pesos en base a la correlación existente entre los datos de entrada.
- **Aprendizaje por refuerzo.** Este se ubica entre medio de los dos anteriores. Se le presenta un conjunto de patrones de entrada y se le indica a la red si la salida obtenida es o no correcta. Aquí, no se le proporciona el valor de la salida esperada. Es muy útil en aquellos casos en que se desconoce cuál es la salida exacta que debe proporcionar la red.
- **Aprendizaje evolutivo:** La evolución biológica nos sirve como un proceso de aprendizaje, ya que organismos biológicos se adaptan para mejorar su supervivencia y así aumentar la posibilidad de tener descendencia. El objetivo es el de modelar esto en un ordenador, usando aptitudes, que corresponden a cuánto de buena es la solución actual.

Fase de operación también llamada Fase de Recuerdo o Ejecución.

Tras finalizar la fase de entrenamiento, el aprendizaje se desconecta. Quedando fijos los pesos y la estructura obtenida en la fase anterior. De esta manera, la red neuronal está preparada para procesar datos. Una de las principales ventajas que posee este modelo es que la red aprende la relación existente entre los datos, adquiriendo la capacidad de generalizar conceptos. De esta manera, una red neuronal puede tratar con información que no le fue presentada durante de la fase de entrenamiento

Algoritmo de retropropagación

“El algoritmo de retropropagación para el error” (8), es uno de los métodos más efectivos para entrenar redes neuronales artificiales. El cual modifica los pesos de conexión entre las neuronas iterativamente, acercándose poco a poco a la respuesta necesaria. El entrenamiento se lleva en dos fases:

- **Propagación hacia delante (forward propagation):** aquí se calculan todas las activaciones de las neuronas de la red. Desde la primera capa hasta la última. En esta los valores de los pesos sinápticos son todos fijos, en la primera iteración se toman los valores por defecto.
- **Propagación hacia atrás (backward propagation):** Se compara la respuesta de la red con la respuesta deseada, obteniendo el error de la red. El error que se ha calculado se propaga en dirección opuesta a la de las conexiones sinápticas. Al final de esta fase, los pesos se modifican según los errores calculados, con el fin de minimizar la diferencia entre la salida actual y la deseada.

Métodos de gradiente descendente

“El método del gradiente descendente” (4) se basa en buscar la dirección en la que una pequeña variación del vector de pesos hace que el error decrezca más rápidamente.

La dirección usamos la que marca el gradiente de la función de error con respecto al vector de pesos.

$$\nabla E(\bar{w}) \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right]$$

Figura 5. Dirección resultante de la función de error con respecto al vector de pesos (4)

Aunque en muchos casos es suficiente encontrar un mínimo local lo suficientemente bueno (9)

Para comprenderlo mejor la siguiente figura representa el hiperplano resultante de la evaluación de una función de error dependiente de un vector de pesos bidimensional:

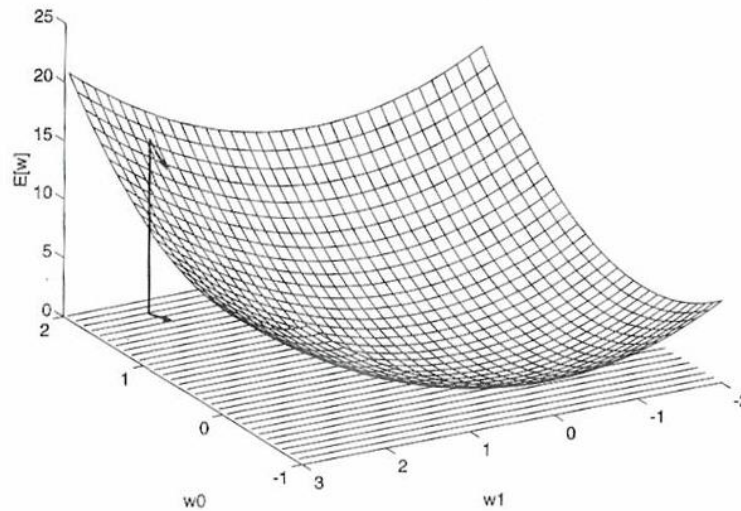


Figura 6. Hiperplano resultante de la evaluación de una función de error. (4)

El método de entrenamiento más utilizado es el método del gradiente descendente. Este método define una función $E(W)$ que proporciona el error que comete la red en función del conjunto de pesos sinápticos W . El objetivo del aprendizaje será encontrar la configuración de pesos que corresponda al mínimo global de la función de error, aunque en muchos casos es suficiente encontrar un mínimo local lo suficientemente bueno (Cauwenberghs, 1993).

Este método tiene el siguiente principio general (8): dado un conjunto de pesos $W(0)$ para el instante de tiempo $t=0$, se calcula la dirección de máxima variación del error. La dirección de máximo crecimiento de la función $E(W)$ en $W(0)$ viene dado por el gradiente $\nabla E(W)$. Posteriormente se actualizan los pesos siguiendo el sentido contrario al indicado por el gradiente $\nabla E(W)$, dirección que indica el sentido de máximo decrecimiento. Así se va produciendo un descenso por la superficie de error hasta alcanzar un mínimo local.

$$W(t+1) = W(t) - \alpha \nabla E(W)$$

donde α indica el tamaño del paso tomado en cada iteración, pudiendo ser diferente para cada peso e idealmente debería ser infinitesimal. "El tamaño del paso" es un factor importante a la hora de diseñar un método de estas características. Un paso muy pequeño, el entrenamiento será muy lento y si el paso es grande corremos el riesgo de quedarnos en un mínimo local y no alcanzar el mínimo global.

El algoritmo Backpropagation

Para representar problemas complejos, no basta un perceptrón simple. Necesitamos una red de perceptrones interconectados entre ellos.

Es un método de aprendizaje supervisado de gradiente descendente, en el que se distinguen claramente dos fases: ya definidas apartado de Algoritmo de Backpropagation (10)

Primero se aplica un patrón de entrada, el cual se propaga por las distintas capas que componen la red hasta producir la salida de esta. Esta salida se compara con la salida deseada y se calcula el error cometido por cada neurona de salida. Estos errores se transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de las capas intermedias. Cada neurona recibe un error que es proporcional a su contribución sobre el error total de la red. Basándose en el error recibido, se ajustan los errores de los pesos sinápticos de cada neurona.

Optimizadores

Tal como se ha expuesto el descenso de gradiente es un algoritmo de aprendizaje automático iterativo para reducir la función de pérdidas.

“Los optimizadores” (11) actualizan los parámetros de peso para minimizar la función de pérdida. La función de pérdida actúa como una guía en el terreno, indicando al optimizador si se está moviendo en la dirección correcta como para llegar al fondo del valle de la figura 6. Como para llegar al mínimo global.

Entre los optimizadores utilizados destacamos los siguientes:

- Adam - Adam puede verse como una combinación de Adagrad, que funciona bien en gradientes dispersos, y RMSprop, que funciona bien en entornos en línea y no estacionarios. Se implementa el promedio móvil exponencial de los gradientes para escalar la tasa de aprendizaje en lugar de un promedio simple como en Adagrad. Mantiene un promedio en decadencia exponencial de gradientes pasados. Es computacionalmente eficiente y requiere muy poca memoria. Es uno de los algoritmos de optimización de descenso de gradientes más populares
- Nadam - Es una combinación de
 - **NAG**: (Gradiente acelerado de Nesterov) que funciona como una bola cuesta abajo, pero que sabe exactamente cuándo reducir la velocidad antes de que la pendiente de la colina vuelva a aumentar. Se calcula el gradiente no con respecto al paso actual sino con respecto al paso futuro. Se evalúa el gradiente de lo anticipado y en función de la importancia, luego se actualizan los pesos
 - y Adam.

Nadam se emplea para gradientes ruidosos o para gradientes con curvaturas altas. El proceso de aprendizaje se acelera sumando la caída exponencial de las medias móviles para el gradiente anterior y actual.

Generalización

Tras finalizar la fase de aprendizaje, la red puede ser utilizada para el propósito para el que fue entrenada. La principal ventaja en el aprendizaje es que el modelo creado aprende la relación entre los datos y con ello obtiene “capacidad de generalización” (8).

Al evaluar la red neuronal es necesario evaluar si la red ha sido capaz de aprender patrones de entrenamiento y lo que es también importante es evaluar su comportamiento ante patrones nunca vistos. Es lo que se conoce como su capacidad de generalización adquirida en el entrenamiento (extrae las características de la muestra claves, para responder luego a nuevos patrones).

En el entrenamiento pueden surgir dos tipos de errores:

- El error de aprendizaje: Es la calidad de la respuesta de la red a los patrones de entrenamiento.
- El error de generalización: La calidad de la respuesta de la red a patrones nunca vistos.

Para poder obtener una medida de ambos errores es necesario dividir el set de datos disponibles en dos, el set de datos de entrenamiento, y el set de datos de evaluación.

El set de datos de entrenamiento se utiliza durante la fase de entrenamiento para que la red pueda extraer las características de estos y, mediante el ajuste de sus pesos sinápticos, la red logre una representación interna de la función.

El set de evaluación se utiliza para evaluar la capacidad de generalización de la red.

Sobreaprendizaje

Es la causa más común de la pérdida de generalización. Sucede cuando la cantidad de ciclos de entrenamiento es elevada. Podemos observar que la respuesta de la red a los patrones de entrenamiento es elevada mientras que su respuesta a los nuevos patrones es muy baja o sufren altibajos muy bruscos. Con mayor número de epochs la red tiende a sobreajustar la respuesta a los patrones de entrenamiento, a expensas de una menor capacidad de generalización.

La próxima figura muestra la situación idealizada de lo dicho anteriormente. Se observa que en un determinado punto la red comienza a perder capacidad de generalización como consecuencia del sobreaprendizaje de los patrones de entrenamiento.

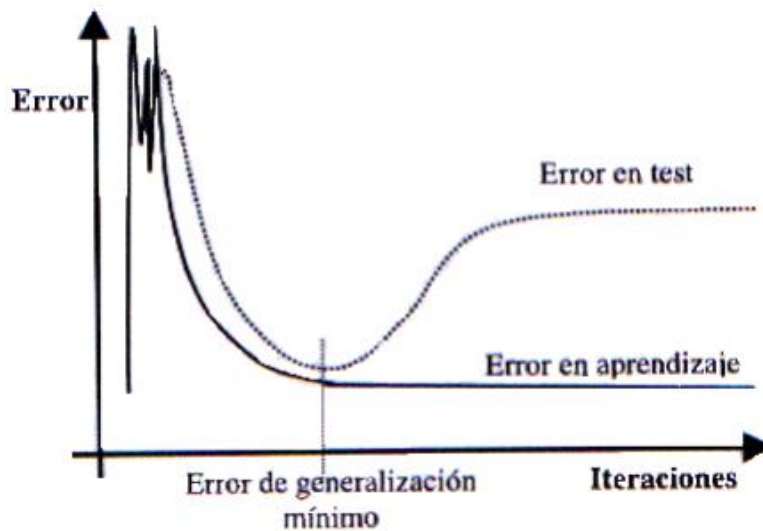


Figura 7. Generalización. Situación idealizada

En la figura 8 se muestra una situación más real del mismo caso. A medida que transcurre el proceso de aprendizaje se obtienen varios mínimos sobre el conjunto de evaluación.

Hay varias técnicas para evitar el sobreaprendizaje como es la parada temprana (early stopping).

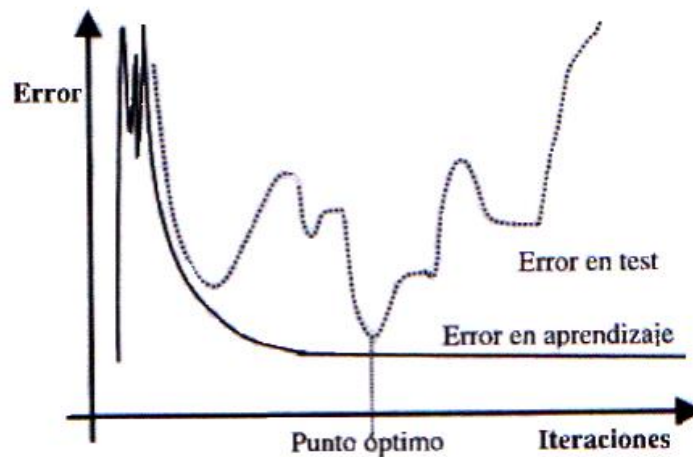


Figura 8. Generalización. Situación real

Hay veces que la capacidad de generalización se pierde por el excesivo uso de neuronas ocultas en la red neuronal. Al hacer este excesivo uso la red tiende a ajustarse con mucha exactitud a los patrones de entrenamiento y no se consigue extraer las características del conjunto (que definen una buena generalización). Este problema se agrava si además los patrones de entrenamiento poseen ruido, ya que la red se va a ajustar a ese ruido.

Métricas de evaluación

En el campo de la inteligencia Artificial y el aprendizaje automático existen diversas herramientas que permiten visualizar el desempeño de un algoritmo de aprendizaje supervisado.

A continuación, se exponen algunos:

Matriz de confusión (12): Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real., o sea en términos prácticos nos permite ver qué tipos de aciertos y errores está teniendo nuestro modelo a la hora de pasar por el proceso de aprendizaje con los datos.



Figura 9. Matriz de Confusión binaria (12)

Exactitud (En inglés Accuracy): es la cantidad de predicciones positivas que fueron correctas.

Se calcula como: $(VP+VN)/(VP+FP+FN+VN)$

El valor predictivo positivo/ tasa de verdaderos positivos/tasa de aciertos/

sensibilidad: Desde un punto de vista médico, es la probabilidad de que, si ha obtenido un resultado positivo en la prueba, realmente tenga la enfermedad.

Se calcula como $VP / VP + FP$.

El valor predictivo negativo /tasa de verdaderos negativos/ **especificidad**: por el contrario, es la probabilidad de que, si ha obtenido un resultado negativo en la prueba, en realidad no tenga la enfermedad.

Se calcula como: $VN/ VN + FN$

Curva ROC: ROC (13) es un acrónimo que viene del inglés Receiver Operating Characteristic (Característica Operativa del Receptor). Es una gráfica que enfrenta la ratio de falsos positivos (eje x) con el ratio de falsos

negativos (eje y). Estas ratios los va obteniendo en función de una serie de umbrales definidos entre 0 y 1. Enfrenta la falsa alarma versus la tasa de éxito.

La tasa de falsos positivos se calcula como el número de positivos verdaderos divididos entre el número de positivos verdaderos y de falsos negativos. Describe cómo de bueno es nuestro modelo prediciendo las clases positivas cuando la salida real es positiva. También se conoce esta tasa como sensibilidad.

La tasa de falsos positivos se calcula como el número de falsos positivos dividido entre la suma de falsos positivos con los verdaderos negativos.

Errores a resolver en modelos de aprendizaje profundo

En un problema de aprendizaje profundo existen una serie de errores que debemos conocer. (14):

e_0 : (Error objetivo). Hemos de fijar un valor de referencia como objetivo y tener en cuenta que alcanzar un error nulo, puede producir problemas de sobreaprendizaje.

e_{BO} : Error óptimo de Bayes Es el mejor error posible para cualquier sistema de aprendizaje automático, que para algunos problemas es equivalente al e_{HL} (Nivel de error humano).

La siguiente figura representa los diferentes tipos de errores que tenemos, así como aquellos que podemos evitar y los que no:

Error	Tipo de error	
e_0		
Diferencia error	Sesgo	Sesgo no evitable
$e_{HL} \approx e_{BO}$		
Diferencia error		Sesgo evitable
e_{train}		
Diferencia error		Varianza
e_{val}		

Figura 10. ¿Qué tipo de error tenemos? - apuntes Deep Learning (14)

- el sesgo evitable es el que obtenemos en nuestra función de pérdidas en el entrenamiento y podemos utilizar diferentes herramientas para poder minimizarlo.
- la varianza que es la diferencia /dispersión entre errores. Diferencia entre el error de entrenamiento y el error de validación

Puede existir que, tras utilizar diferentes herramientas para minimizar la varianza, no sea posible porque **“no tenemos suficientes datos”**.

Recursos externos

Ante un problema de **“no disponer de suficientes datos”** como para obtener un buen rendimiento en un modelo de aprendizaje automático, una posible solución es utilizar recursos externos.

Es importante cuando se utiliza recursos externos que:

- Que sean datos del mismo dominio.
- Ver si existen diferencias entre los datos, a veces inapreciables. Para ello deberemos de comprobar si en nuevo conjunto de entrenamiento es representativo de nuestro conjunto de datos de producción.

El objetivo por alcanzar en esta unión de datos es que:

Error	Tipo de error	
e_0		
Diferencia error		Sesgo no evitable
$e_{HL} \approx e_{BO}$	Sesgo	
Diferencia error		Sesgo evitable
e_{train}		
Diferencia error		Varianza
$e_{train/val}$		
Diferencia error		Datos train no representativos
e_{val}		

Error lo más parecido para ser representativo

Figura 11. Errores tras la unión de bases de datos - apuntes Deep Learning (14)

Para poder conocer si nuestro nuevo conjunto de datos de entrenamiento (tras la unión de datos) es representativo de nuestro conjunto de datos inicial, deberemos realizar la siguiente partición de datos:

- Un conjunto de validación final y testeo que debe estar formado únicamente por datos de producción
- Un conjunto de datos de entrenamiento que estará formado por el nuevo conjunto de los datos más parte de los datos de producción.
- A partir de este nuevo conjunto de datos de entrenamiento, a su vez debemos realizar partición de datos entre entrenamiento y validación.

Decimos que el nuevo conjunto de datos de entrenamiento es representativo de nuestro conjunto de producción cuando la varianza que es la diferencia/dispersión entre errores alcanzado entre los nuevos datos de entrenamiento y los nuevos datos de validación, es lo más similar posible al error que obtenemos al aplicar el nuevo modelo entrenado en los datos de test (solo datos de producción) que reservamos para la validación final.

2.2. REDES DE NEURONAS CONVOLUCIONALES

Desde 2012, uno de los avances más importantes en el aprendizaje profundo es el uso de las redes de neuronas convolucionales para obtener mejores resultados en el reconocimiento de imágenes.

“Las redes de neuronas convolucionales” (CNN) (8) presentan múltiples capas para calcular la salida dado un conjunto de datos. El desarrollo de este modelo comenzó en la década de 1950, donde los investigadores Hubel y Wiesel modelaron la corteza visual animal.

En un artículo en el año 1968, ambos debatieron sus hallazgos, los cuales identificaron tanto células simples como complejas dentro de los cerebros de monos y gatos que ellos estudiaron. Observaron que las células simples tuvieron un rendimiento maximizado. Por otro lado, se observó que el cuerpo receptivo en las células complejas era considerablemente más grande, y sus salidas no se veían afectadas por las posiciones de los bordes dentro del cuerpo receptivo.

Además del reconocimiento de imágenes, para lo cual las redes de neuronas convolucionales fueron originalmente creadas, tienen otras aplicaciones considerables, como dentro de los campos del procesamiento natural del lenguaje y aprendizaje reforzado.

2.3.1. Estructura y propiedades de las CNN

Las redes de neuronas convolucionales son en términos generales, modelos de redes de neuronas artificiales multicapa. De acuerdo con la estructura de la corteza visual animal descrita por Hubel y Weisel, el modelo puede ser interpretado visualmente como se muestra en la siguiente figura.

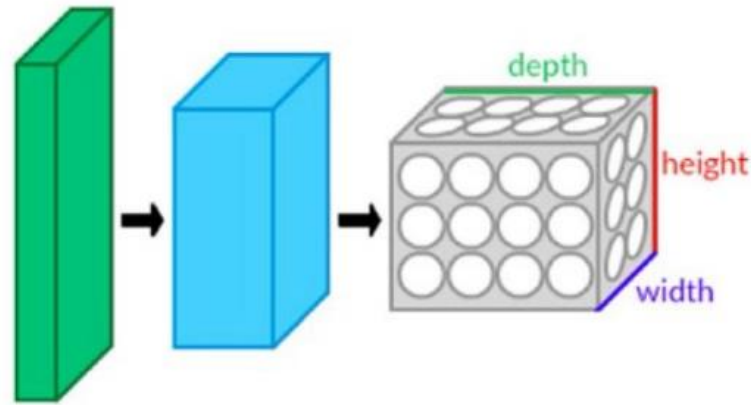


Figura 12. Visualización de una CNN (8)

Cada uno de los bloques representa una capa diferente de la red de neuronas convolucionales, las cuales tienen altura, anchura y profundidad. De izquierda a derecha podemos observar, las capas de entrada, las capas ocultas (convolucional, agrupación y capas de abandono) y las capas completamente conectadas. Después de la última capa, el modelo genera una clasificación.

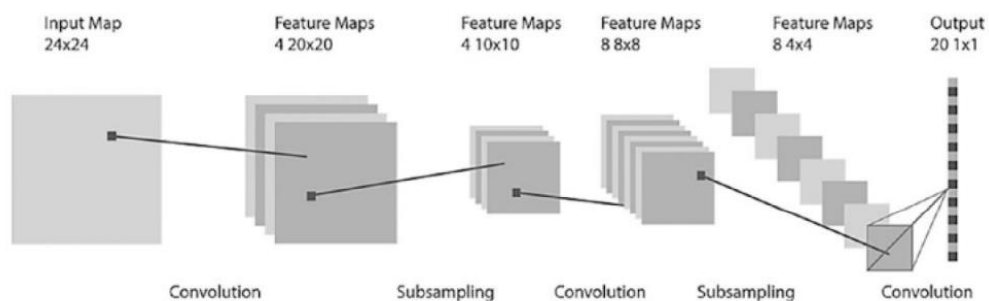


Figura 13. Arquitectura de una CNN (8)

Las capas completamente conectadas imponen la conectividad local entre las neuronas y las capas adyacentes. Como tal, las entradas de las capas ocultas son un subconjunto de neuronas de la capa que precede a dicha capa oculta. Esto asegura que las neuronas del subconjunto de aprendizaje produzcan la mejor respuesta posible. Además, las unidades comparten el mismo peso y sesgo en el mapa de activación, de modo que todas las neuronas que estén en una capa dada están analizando la misma característica.

Tal como podemos ver están formadas por tres tipos de capas:

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

La 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” (15)

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 si mismo sino sus píxeles. Las dimensiones de la matriz de píxeles de una imagen vienen determinadas 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 una capa de entrada está totalmente relacionado con el número de píxeles de la imagen, de manera que una imagen con dimensiones $64 \times 64 \times 3 = 12.288$ tendrá 12.288 neuronas en la capa de entrada. Es importante también normalizar los píxeles antes de que se introduzcan a la red.

2.2.2. La transferencia de aprendizaje

La transferencia de aprendizaje es cuando se utiliza un modelo previamente entrenado, con una gran cantidad de imágenes de manera, que las características aprendidas por dicho modelo pueden ayudarnos en nuestro problema de clasificación, aunque las clases de salida sean totalmente diferentes.

2.2.3. Xception

Xception (16), es una arquitectura que fue propuesta por François Chollet en el año 2017 (el creador de Keras) y lo único que aporta a Inception es que hace las circunvoluciones de forma óptima para que tomen menos tiempo. Esto se logra separando las convoluciones 2D en convoluciones 2 1D (17)-.

Las principales características (18) son:

- Arquitectura de red neuronal convolucional basada enteramente en capas de convolución separables en profundidad.

- Hipótesis fundamental: el mapeo de las correlaciones entre canales y las correlaciones espaciales puede desacoplarse por completo.
- Compuesta por 36 capas convolucionales que forman la base de extracción de características de la red.
- Estructurada en 14 módulos, todos ellos con conexiones lineales residuales a su alrededor, excepto el primer y el último módulo

Este es el esquema de su arquitectura:

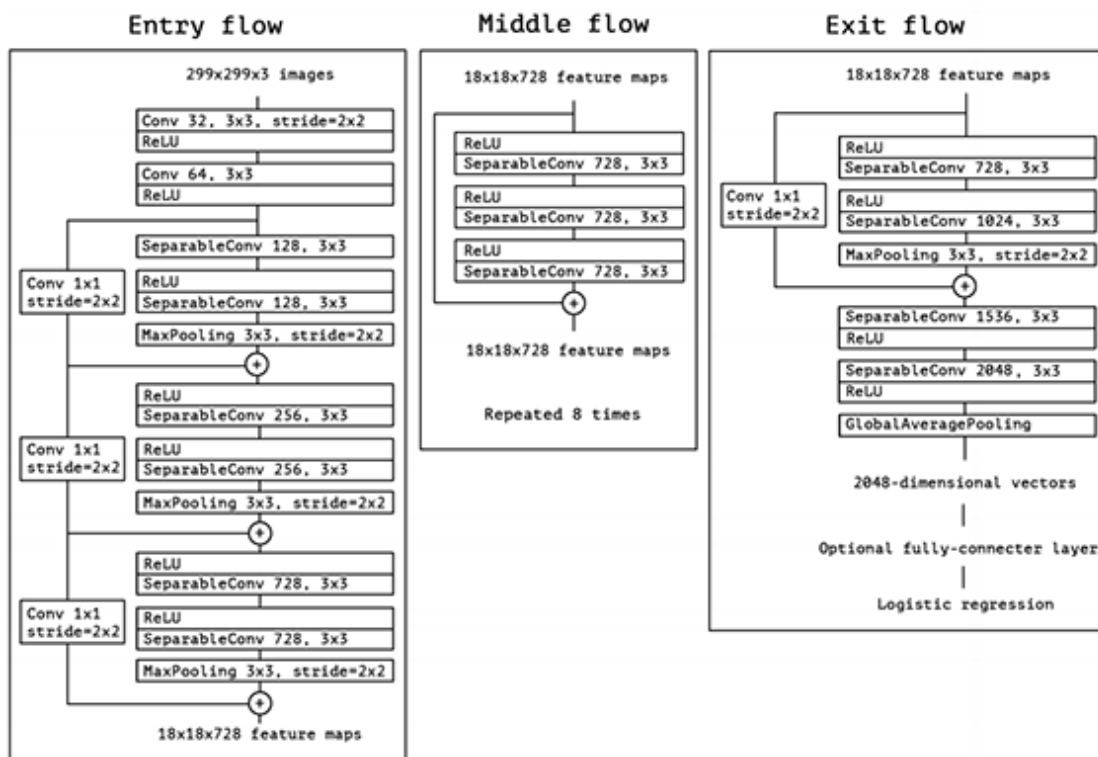


Figura 14. Esquema arquitectura de Xception (16)

En esta arquitectura La hipótesis de partida fue: “las correlaciones espaciales y las que existen entre los canales son lo suficientemente libres para hacer que sea preferible no mapearlas juntas” (19).

Esto significa que, en una CNN tradicional, los estados convolucionales identificaron correlaciones entre el espacio y la profundidad. En la siguiente imagen podemos revisar la operación.

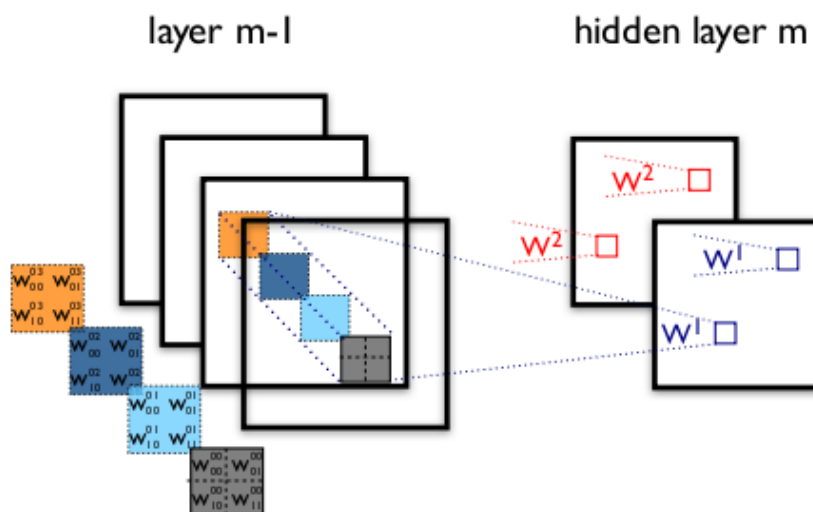


Figura 15. Correlaciones entre el espacio y la profundidad.

El filtro considera simultáneamente una dimensión espacial (cada cuadrado de 2×2 colores) y un cross-channel o dimensión de “profundidad” (la pila de cuatro cuadrados). En la capa de entrada de una imagen, esto es el equivalente de un filtro convolucional que examina un grupo de píxeles de 2×2 que atraviesa los tres canales RGB.

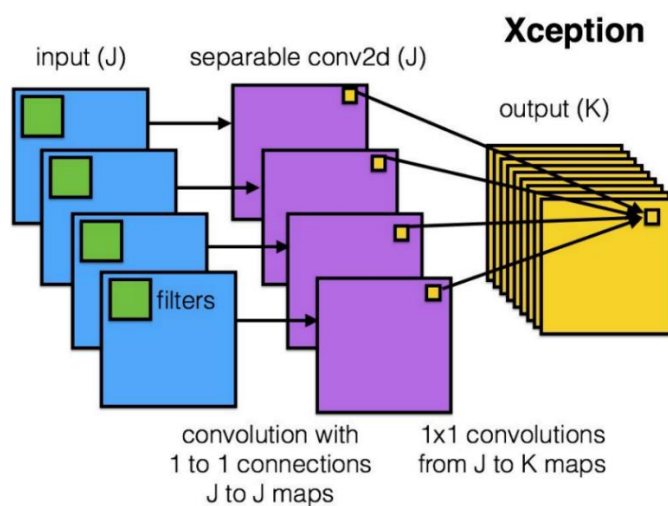


Figura 16. Arquitectura Xception (19)

2.3.1. Otras Arquitecturas de Redes Neuronales Convolucionales

VGG16 y VGG19

“VGG16” (16) es una arquitectura, que fue una de las primeras en aparecer, presentada por Simonyan y Zisserman en 2014 con su artículo titulado “Redes convolucionales muy profundas para el reconocimiento de imágenes a gran escala” (20).

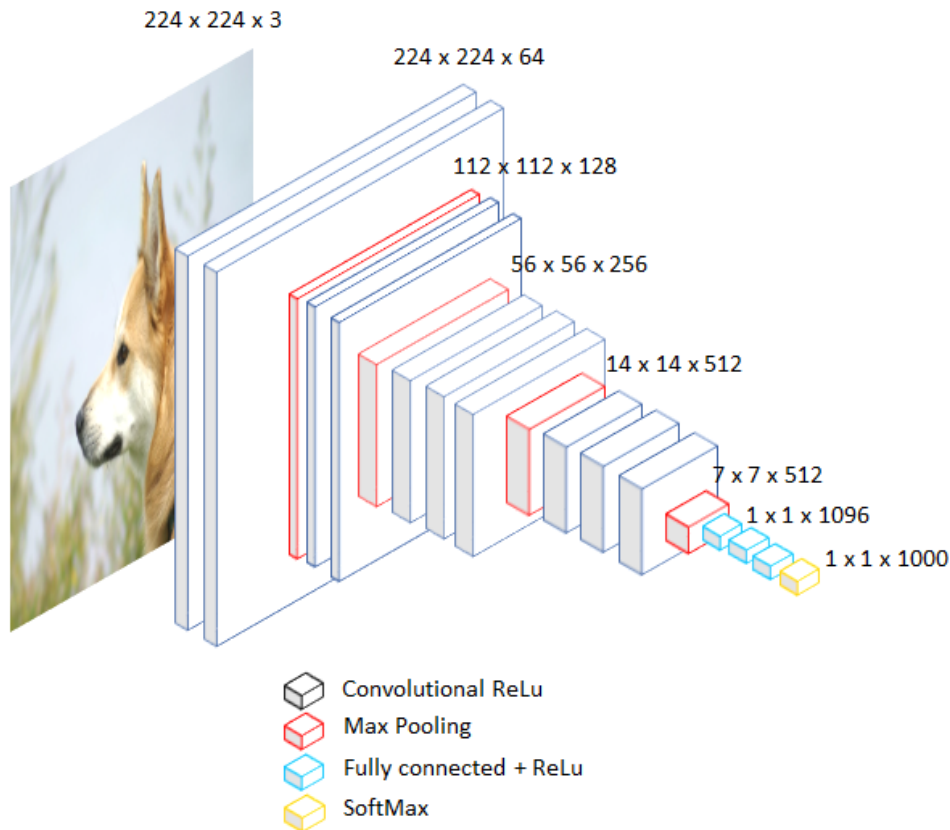


Figura 17. Descripción general de VGG16 (16)

Compuesta por 16 capas de convolución (16), es una arquitectura simple, que utiliza solo bloques compuestos por un número incremental de capas convolucionales con filtros de tamaño 3x3. Además, para reducir el tamaño de los mapas de activación obtenidos, los bloques de agrupación máxima se intercalan entre los convolucionales, reduciendo el tamaño de estos mapas de activación a la mitad. Finalmente, se utiliza un bloque de clasificación, que consta de dos capas densas de 4096 neuronas cada una, y la última capa, que es la capa de salida, de 1000 neuronas

El 16 y el 19 se refieren al número de capas ponderadas que tiene cada red (capas convolucionales y densas, las capas de agrupación no se cuentan).

Sin embargo, esta red tiene un par de desventajas:

- Se tarda demasiado en entrenar.
- Tiene un número muy elevado de parámetros.

Inception V3

Este tipo de arquitectura (16), que fue introducido en 2014 por Szegedy en su artículo "Profundizando en las convoluciones" (21). Utiliza bloques con filtros de diferentes tamaños que luego se concatenan para extraer

entidades a diferentes escalas. El objetivo del bloque de inicio es calcular mapas de activación con convoluciones 1x1, 3x3 y 5x5 para extraer características a diferentes escalas. Luego, simplemente concatena todos estos mapas de activación en uno. Esta arquitectura requiere incluso menos memoria que VGG y ResNet.

AlexNet

“AlexNet” (22), es una arquitectura que utiliza un subconjunto “ImageNet” que consta de alrededor de 1000 imágenes en cada una de las 1000 categorías que tiene. Requiere una dimensionalidad constante de entrada y salida. Las imágenes se reducen a una resolución fija de 256x256. Está compuesta por 5 capas convolucionales y 3 capas completamente conectadas.

GoogleNet

GoogLeNet (23), es una red neuronal convolucional profunda de 22 capas que es una variante de Inception Network. Desarrollada por investigadores de Google. Hoy en día, GoogLeNet se utiliza para otras tareas de visión por computadora, como detección y reconocimiento de rostros, entrenamiento de adversarios, etc.

ResNet

ResNet (24), Microsoft presentó una CNN en la conferencia citada basada en bloques residuales. Fue ganadora del reto ya que bajó la tasa de error hasta el 3,57%. Consiguió con los bloques residuales y las 152 capas tener menos complejidad al entrenar que VGGNet. Con esta nueva arquitectura se batieron los records en clasificación, localización y detección de objetos. (25)

DenseNet

DenseNet (26). En esta arquitectura cada capa obtiene entradas adicionales de todas las capas anteriores y pasa sus propios mapas de características a todas las capas posteriores. Se utiliza la concatenación. Cada capa está recibiendo “conocimiento colectivo” de todas las capas anteriores. (27)

2.3. CLASIFICACIÓN AUTOMATIZADA DE ANOMALÍAS DE LAS RADIOGRAFÍAS DE TÓRAX CON EL USO DE REDES CONVOLUCIONALES PROFUNDAS.

Este apartado se basa en la recopilación de los apartados más destacados de un artículo médico (28) cuyo objeto es obtener algoritmos de clasificación binaria (con anomalía versus sin anomalía) con el uso de redes neuronales convolucionales. El artículo se divide en los siguientes apartados: parte inicial en la que se analiza el desafío, parte en la que se describe el estudio realizado y parte final en la que se presentan las principales conclusiones de este artículo

Una de las principales causas de morbilidad y mortalidad, así como de uso de los servicios de salud en el mundo, son las anomalías cardiotorácicas y pulmonares. (29). El cáncer de pulmón es la principal causa de muerte por cáncer tanto de mujeres como de hombres en los Estados Unidos, y más de 33 millones de estadounidenses padecen una enfermedad pulmonar crónica, según la Asociación Estadounidense del Pulmón. (30).

La radiografía de tórax es el examen radiológico más solicitado por su eficacia en la caracterización y detección de anomalías cardiotorácicas y pulmonares. Se usa tanto en la prevención como en la detección del cáncer de pulmón.

Lo que se necesitaría es que el radiólogo informe a tiempo de cada imagen, pero no siempre es posible, debido a la gran carga de trabajo en muchos grandes centros de salud o también la falta de radiólogos experimentados en áreas menos desarrolladas.

Por lo que disponer de, un sistema automatizado de clasificación de anomalías en la radiografía de tórax sería muy importante ya que permitiría a los radiólogos poder centrarse más en la evaluación de aquellas radiografías con anomalías (patologías de tórax).

El aprendizaje profundo ha tenido un éxito importante en los últimos años. Emerge como herramienta líder de aprendizaje automático en varios campos, como la visión por computadora, el procesamiento del lenguaje natural, el reconocimiento de la voz, el análisis de redes sociales, la bioinformática y el análisis de imágenes médicas. (31).

Las redes neuronales convolucionales toman datos sin procesar (por ejemplo, imágenes) como entrada y realizan una serie de operaciones convolucionales y no lineales para aprender jerárquicamente información rica sobre la imagen, con el fin de cerrar la brecha entre la representación de alto nivel y las características de bajo nivel. En la fase de entrenamiento, ajustan sus valores de filtro (pesos) optimizando ciertas funciones de

pérdida a través de pasos hacia adelante y procedimientos de propagación hacia atrás, de modo que las entradas se mapeen correctamente en las etiquetas de verdad del terreno.

El primer trabajo realizado de este tipo fue con radiografías de tórax para la clasificación de (Tuberculosis pulmonar, detección de nódulos pulmonares) (32).

La reciente publicación de conjunto de datos a gran escala como:

- NIH ChestX-ray 14: es un conjunto de datos de imágenes médicas que comprende 112.120 imágenes de radiografías de tórax de vista frontal de 30.805 (recopilados desde el año de 1992 a 2015) pacientes únicos con las catorce etiquetas de enfermedades comunes extraídas de informes médicos a través de procesamiento de lenguaje natural. PNL. (33). Fuente de los datos en: (34)
- CheXpert: un gran conjunto de datos que contiene 224.316 radiografías de tórax de 65.240 pacientes. (35).
- MIMIC- CXR: El conjunto de datos contiene 377.110 imágenes correspondientes a 227.835 estudios radiográficos realizados en el Beth Israel Deaconess Medical Center en Boston, MA. (36).

Han permitido la realización de muchos estudios de aprendizaje profundo para el diagnóstico automatizado de radiografía de tórax.

El rendimiento de estos algoritmos no es tan bueno como el de los radiólogos para muchas categorías debido principalmente:

- Desequilibrio de clases del conjunto de datos.
- Ruido de la etiqueta causado por el procesamiento del lenguaje natural (NLP). (37)
- Incluso en el etiquetado de las imágenes por diferentes radiólogos hay diferencia de pareceres. (38) Según dataset referenciado sólo en el 41,8% de los casos hay coincidencia de etiqueta de salida por 3 anotadores en el resto no.

Aun así, se puede entrenar una red neuronal convolucional profunda para poder identificar las radiografías de tórax anormales con un rendimiento adecuado y así poder priorizar estos casos, para una rápida revisión y reporte. (31).

- Un estudio reciente (39), presentó una CNN entrenada y probada en la combinación de radiografías anormales ($n = 51,760$, 97.4%) de la base de datos de NIH "ChestX-ray 14" y radiografías normales ($n = 1389$, 2.6%) del hospital de la Universidad de Indiana. Se consiguió un (AUC) de 0,98. (intervalo de confianza (IC) del 95%: (0,97, 0,99)), una sensibilidad del 94,6% y una especificidad del 93,4%.

Aspectos a tener en cuenta:

- las radiografías normales se extrajeron de un hospital, mientras que las anormales fueron de otro hospital debido a la disponibilidad de imágenes y etiquetas, lo que podría sesgar la evaluación. (P. Ej., Al clasificar según diferentes calidades o intensidades, o incluso fabricantes de dispositivos de radiografías diferentes).
- El modelo fue entrenado y probado en radiografías mayoritariamente anormales que era muy poco probable que represente el mundo real.
- Otro estudio más reciente, Annarumma et al. (40) Con una base de datos de 0,5 millones de radiografías de tórax digitales etiquetadas por NLP. En este caso el modelo debía de clasificar el nivel de prioridad (como: crítico, urgente, no urgente y normal). La sensibilidad y la especificidad de este sistema predictivo de anomalías críticas fueron del 65% y el 94%, respectivamente.
 - Se comprobó que las radiografías anormales con hallazgos críticos fueron revisadas antes por los radiólogos (2.7 versus 11.2 días en promedio) por lo que con la ayuda de los modelos automatizados se mejoró la práctica real de estos casos.

2.3.1. Descripción del estudio realizado

Se presenta descripción del estudio realizado en el artículo referenciado (28):

- Se valora el rendimiento de las CNN en la tarea de clasificar las radiografías de tórax normales frente las anormales.
- Se restringe las comparaciones entre los algoritmos y los radiólogos a la clasificación basada en imágenes.
- Se utilizan varias arquitecturas de CNN:
 - AlexNet (22),
 - VGG (41),
 - GoogLeNet (23),
 - ResNet (24)y
 - DenseNet (26).
- Se entrenaron y validaron los datos en los conjuntos de train, validation y test según las etiquetas de los radiólogos participantes en el estudio y el consenso de tres radiólogos certificados por la junta, previamente.
- Se utilizo curvas ROC, AUC, matriz de confusión para la evaluación de los modelos.
- (42), presentó en su día un sistema similar entrenado y probado en las radiografías de su institución.
 - Lograron un AUC de 0.96 en la tarea de clasificación normal versus anormal.

- compararon (1) el impacto de diferentes arquitecturas CNN para la clasificación binaria, (2) el efecto del entrenamiento desde cero y el entrenamiento previo, (3) las diferencias entre el radiólogo asistente (que leyó el escaneo original y redactó el informe de texto) y las etiquetas de consenso del radiólogo, y
- **Concluyo que es útil combinar la predicción del modelo con la lectura del radiólogo asistente.**
- En base a la referencia (42) , se realizaron evaluaciones con más arquitecturas, se realizaron evaluaciones con datos externos con el fin de mejorar la generalización entrenada.

Finalmente, **Los resultados indican que las redes neuronales profundas alcanzan una precisión a la par con los radiólogos experimentados.**

A continuación, se presentan los principales resultados para diferentes conjuntos de datos analizados:

Resultado Rendimiento del modelo en el conjunto de datos de NIH "ChestX-ray 14"

- Se utilizaron como el estándar de referencia de la "verdad fundamental" las etiquetas de consenso de tres radiólogos certificados por la junta de EE.UU. (La mayoría de los votos del radiólogo n.º 1, n.º 2 y n.º 3).
- El conjunto de datos es: ChestX-ray 14 (33), utilizando imágenes con una resolución 256x256 píxeles.
- Se entrenaron en las diferentes arquitecturas de CNN recogidas en la Descripción de este estudio.
- Todas las arquitecturas de CNN lograron AUC superior a 0,96, lo que demuestra un buen desempeño para esta tarea de clasificación binaria.
- El método de aprendizaje por transferencia (pesos CNN preentrenados en ImageNet (43) superó a los modelos entrenados desde cero (pesos CNN inicializados aleatoriamente) ($P < 0.05$ (rango [0.004, 0.047]) para todos los modelos CNN) con un conjunto de entrenamiento de tamaño moderado de unas 8500 imágenes.
- El valor predictivo positivo (VPP), que indica la probabilidad de que la radiografía sea anormal cuando la predicción es positiva, fue del 92,84%; el valor predictivo negativo (VPN), que indica la probabilidad de que la radiografía sea normal cuando la predicción es negativa, fue del 96,52%

Se muestra a continuación imágenes con los principales resultados:

Clasificación de anomalías automatizada de radiografías de tórax utilizando redes neuronales convolucionales profundas

Modelos	AUC	Sensibilidad (%)	Especificidad (%)	PPV (%)	VAN (%)	Puntuación F 1	Precisión (%)
AlexNet (P)	0,9741 ± 0,0050	94,18 ± 0,47	87,70 ± 0,56	87,66 ± 0,61	94,10 ± 0,41	0,9091 ± 0,0057	90,85 ± 0,48
AlexNet (C)	0,9684 ± 0,0043	92,65 ± 0,45	87,99 ± 0,41	87,94 ± 0,57	92,68 ± 0,38	0,9023 ± 0,0052	90,25 ± 0,45
VGG16 (P)	0,9797 ± 0,0039	94,03 ± 0,36	90,74 ± 0,41	90,56 ± 0,45	94,14 ± 0,43	0,9226 ± 0,0038	92,34 ± 0,40
VGG16 (S)	0,9742 ± 0,0044	93,42 ± 0,40	91,46 ± 0,46	91,18 ± 0,50	93,63 ± 0,46	0,9228 ± 0,0040	92,41 ± 0,42
VGG19 (P)	0,9842 ± 0,0036	97,09 ± 0,39	87,99 ± 0,35	88,42 ± 0,41	96,97 ± 0,43	0,9255 ± 0,0035	92,41 ± 0,33
VGG19 (S)	0,9757 ± 0,0054	94,49 ± 0,59	88,86 ± 0,49	88,90 ± 0,56	94,46 ± 0,47	0,9161 ± 0,0048	91,59 ± 0,50
ResNet18 (P)	0,9824 ± 0,0043	96,50 ± 0,36	92,86 ± 0,48	92,84 ± 0,55	96,52 ± 0,30	0,9463 ± 0,0041	94,64 ± 0,45
ResNet18 (S)	0,9766 ± 0,0034	96,63 ± 0,41	85,09 ± 0,33	85,97 ± 0,47	96,39 ± 0,36	0,9099 ± 0,0034	90,70 ± 0,38
ResNet50 (P)	0,9837 ± 0,0048	96,94 ± 0,50	88,42 ± 0,61	88,78 ± 0,73	96,83 ± 0,39	0,9268 ± 0,0055	92,56 ± 0,54
ResNet50 (S)	0,9775 ± 0,0057	94,32 ± 0,54	90,59 ± 0,66	90,43 ± 0,75	94,42 ± 0,44	0,9233 ± 0,0059	92,40 ± 0,60
Inicio-v3 (P)	0,9866 ± 0,0041	97,38 ± 0,35	87,57 ± 0,48	88,11 ± 0,55	97,26 ± 0,27	0,9250 ± 0,0051	92,33 ± 0,42
Inception-v3 (S)	0,9796 ± 0,0034	95,08 ± 0,32	89,58 ± 0,35	89,58 ± 0,42	95,08 ± 0,23	0,9225 ± 0,0047	92,25 ± 0,37
DenseNet121 (P)	0,9871 ± 0,0057	97,40 ± 0,53	87,55 ± 0,68	88,09 ± 0,74	97,27 ± 0,33	0,9251 ± 0,0056	92,34 ± 0,56
DenseNet121 (S)	0,9801 ± 0,0044	95,10 ± 0,38	90,01 ± 0,49	90,00 ± 0,61	95,11 ± 0,27	0,9248 ± 0,0041	92,49 ± 0,44

Figura 18. Métricas de rendimiento de clasificación para diferentes arquitecturas de CNN en la base de datos de NIH "ChestX-ray 14" (28)

Las predicciones del modelo de CNN se compararon con las etiquetas de consenso de tres radiólogos certificados por la junta.

Área bajo la curva *AUC*, valor predictivo positivo de *PPV* (o precisión), valor predictivo negativo de *VPN*. P: los pesos del modelo se inicializaron a partir del modelo preentrenado de ImageNet. S: inicialización aleatoria de pesos del modelo, es decir, entrenamiento desde cero.

Curvas (ROC) para diferentes arquitecturas CNN pre-entrenadas en ImageNet versus radiólogos en el conjunto de datos NIH "ChestX-ray 14."

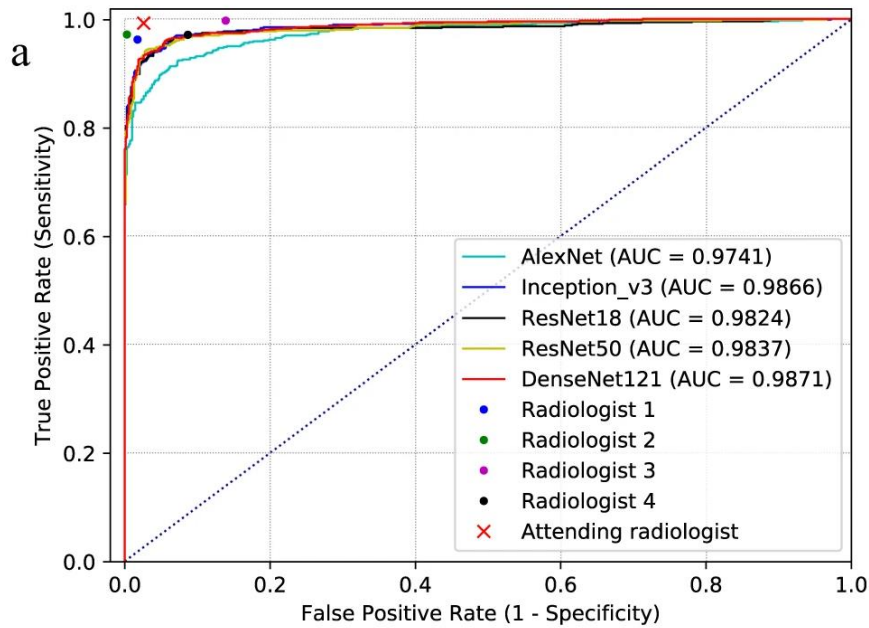


Figura 19. Curva ROC modelos CNN **(a)** Etiquetas elegidas por la mayoría de los radiólogos como el estándar de referencia de verdad del terreno (28).

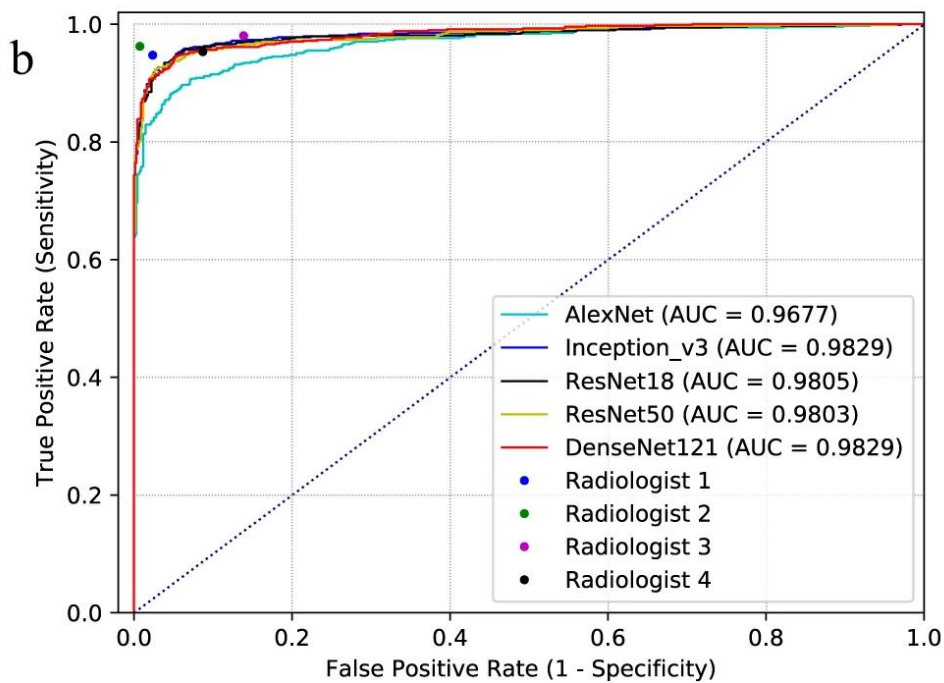


Figura 20. Curva ROC modelos CNN **(b)** Etiquetas derivadas de informes de texto como estándar de referencia de verdad sobre el terreno (28).

Los niveles de rendimiento de los radiólogos se representan como puntos únicos (o una cruz para el radiólogo asistente que redactó el informe de radiología). AUC Área bajo la curva.

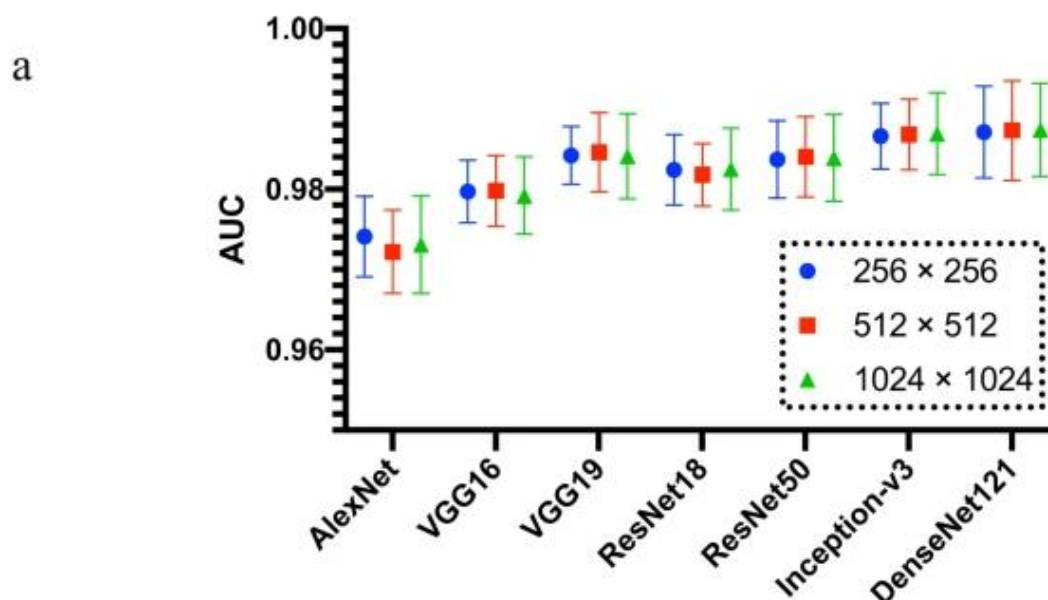
Comparaciones del rendimiento del modelo y diferentes radiólogos

Figura 21. Rendimiento de diferentes arquitecturas de CNN con diferentes tamaños de imagen de entrada en el conjunto de datos "ChestX-ray 14" (28).

Las AUC logradas con modelos entrenados usando un tamaño de imagen de entrada de 256×256 , 512×512 o 1024×1024 píxeles no fueron significativamente diferentes. Las barras de error representan las desviaciones estándar de los valores medios.

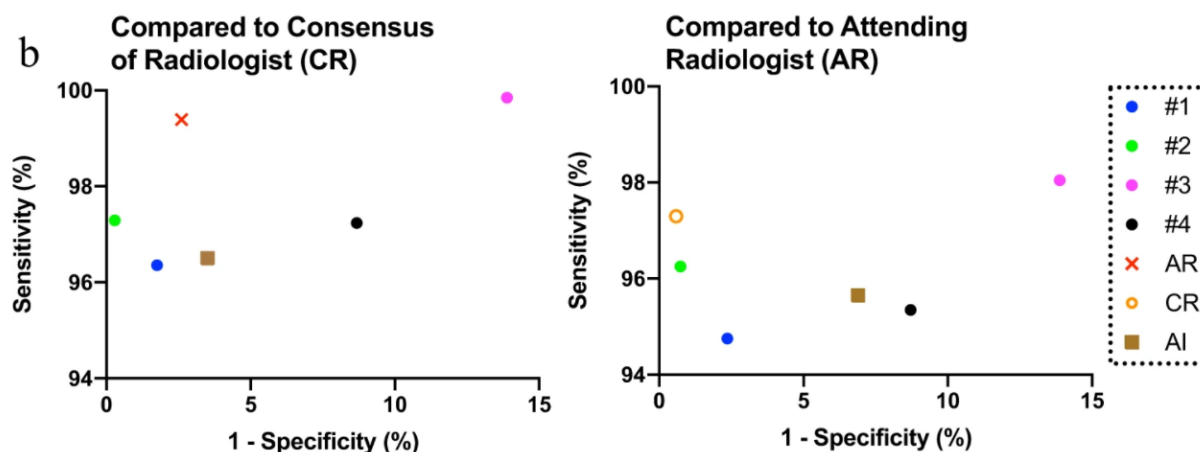


Figura 22. Tasa de verdaderos positivos (sensibilidad) y tasa de falsos positivos (especificidad 1) de diferentes radiólogos ($n.^\circ 1$, $n.^\circ 2$, $n.^\circ 3$ y $n.^\circ 4$) frente a diferentes etiquetas de verdad del terreno (28)

A la izquierda se muestran comparaciones de desempeño cuando se establece el consenso de los radiólogos como una verdad fundamental. La derecha muestra comparaciones cuando se establecen etiquetas del radiólogo asistente como verdad fundamental. AR: radiólogo asistente, CR: consenso de radiólogos (voto de la mayoría de tres radiólogos certificados por la junta), AI: el modelo de inteligencia artificial (modelo ResNet18 CNN que se muestra aquí).

Rendimiento del modelo en el conjunto de datos de desafío de detección de neumonía RSNA

Los datos RSNA se pueden extraer de la siguiente referencia. (44).

En este caso, inicialmente se entrena un clasificador CNN normal versus anormal y se realiza un cross-validation con 7 folds. en 21.152 radiografías de tórax (normal = 6.993, 33.06%; anormal = 14.159, 66.94%). Se utiliza modelo CNN VGG-19.

Con posterioridad se aplica el modelo de datos obtenidos para muestras de datos más reducidas.:

- El modelo se probó en un conjunto de pruebas de 4.532 radiografías (normal = 1.532, 33,80%; anormal = 3.000, 66,20%): El valor predictivo positivo fue del 94,30% y el valor predictivo negativo fue del 78,11%.

En un segundo momento se entrenó clasificador VGG19 de opacidad pulmonar normal versus similar a la neumonía, con cross-validation de 7 folds en 11.652 radiografías de tórax (un subconjunto del primer experimento, donde, Normal=6.993, 60,02%; anormal con opacidad pulmonar= 4.659, 39,98%).

También se realiza una prueba con menor cantidad de datos:

- 2.532 radiografías (normal= 1.532, 60,51%; anormal con opacidad pulmonar= 1.000; 39,49%). El valor predictivo positivo fue del 94,27% y el valor predictivo negativo del 94,98%.

A continuación, se muestran las matrices de confusión:

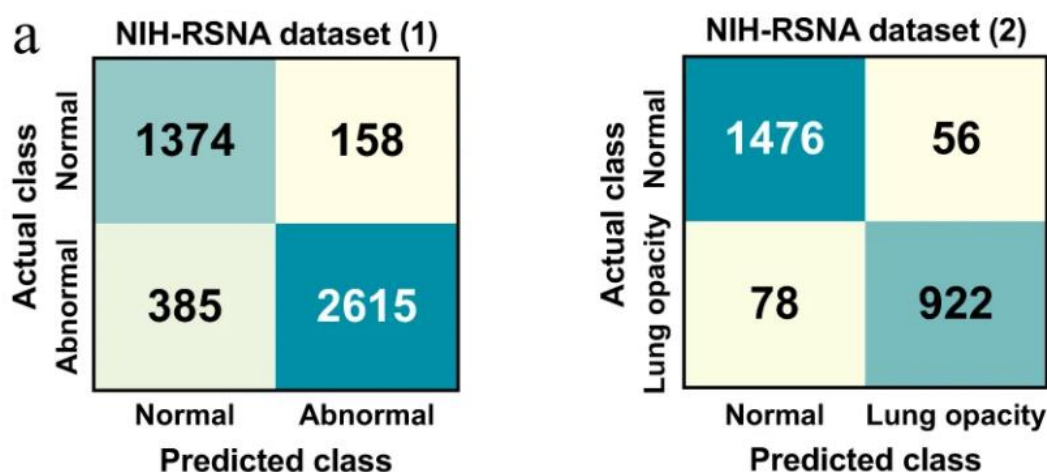


Figura 23. Matrices de confusión para los diferentes conjuntos de datos de NIH-RSNA entrenados (28).

Matrices de confusión del rendimiento del modelo VGG-19 CNN en diferentes conjuntos de datos. Izquierda: conjunto de datos de prueba de

RSNA para clasificación normal versus anormal. Derecha: conjunto de datos de prueba de RSNA para la clasificación de opacidad pulmonar normal frente a lung opacity.

2.3.2. Conclusiones

- La detección automatizada, rápida y confiable de enfermedades basada en la radiografía de tórax es un paso crítico en el flujo de trabajo de la radiología.
- El objetivo del análisis fue la creación de un modelo de clasificación binario, (radiografías de tórax normales versus anormales) con el objetivo de ayudar a alertar a los radiólogos y médicos de posibles hallazgos anormales, sirviendo como un medio de clasificación de listas de trabajo y priorización de informes.
- Se demostró que es capaz de igualar el desempeño de los radiólogos en la tarea de clasificación binaria, en las radiografías de prueba provenientes de la misma institución que las radiografías de tórax de entrenamiento, con un tiempo de inferencia significativamente menor (50s para una red de aprendizaje profundo versus 2,3h para los radiólogos en promedio para 1.344 radiografías de tórax).
- La apariencia y las propiedades estadísticas de las radiografías de tórax se ven afectadas por la tecnología del escáner, los ajustes de adquisición y las técnicas de posprocesamiento.
- En el conjunto de datos de detección de neumonía de la RSNA, los radiólogos anotaban las etiquetas de forma puramente manual basándose únicamente en la apariencia de las imágenes y en el conjunto de datos "ChestX-ray 14", las etiquetas se extrajeron de informes médicos (etiquetado NLP), y los resultados no pueden ser comparables porque: (1) Existía diferencias en cuanto a número de imágenes en los conjuntos de entrenamiento y testeo. (112.120 imágenes versus 21.152 imágenes) (2) existían diferentes proporciones de imágenes normales/anormales tanto en el entrenamiento como en el conjunto de pruebas.
- Un modelo basado en CNN logró un AUC de $0,9824 \pm 0,0043$ (con una precisión de $94,64 \pm 0,45\%$, una sensibilidad de $96,50 \pm 0,36\%$

y una especificidad de $92,86 \pm 0,48\%$) para la clasificación de radiografía de tórax normal frente a anormal.

- El entrenamiento previo, ayuda en el entrenamiento posterior de un conjunto de imágenes de entrenamiento de tamaño moderado de aproximadamente 8.500 imágenes.
- El preentrenamiento podría ser más beneficioso para un conjunto de datos de tamaño moderado (p. Ej., Alrededor de 8.500 imágenes en el conjunto de datos del NIH o 18.000 imágenes) que en un conjunto de datos suficientemente grande (p. Ej., 180.000 imágenes).
- El notable desempeño en la precisión diagnóstica observado en este estudio muestra que las CNN profundas pueden diferenciar con precisión y eficacia las radiografías de tórax normales y anormales, en bases de datos bien balanceados.
- Una crítica común a los modelos de aprendizaje profundo en radiología es que a menudo encuentran **“problemas de generalización”** debido a la gran diferencia entre los dominios de origen y destino. Descubrimos con el uso de ajuste fino esto puede mejorar los resultados. En el proceso de ajuste fino se aprende las características comunes de ambos dominios, y esto permite una mejor inicialización de los parámetros del modelo y como consecuencia un entrenamiento más rápido.
- Este estudio puede permitir futuros estudios de aprendizaje profundo de otras enfermedades torácicas en las que actualmente solo se encuentran disponibles conjuntos de datos más pequeños.

3. PLATEAMIENTO DEL PROBLEMA

En este apartado se realiza una descripción del contexto general que envuelve el problema a resolver. Se detallan los principales problemas que se derivan de las anotaciones realizadas por la propia competición del proyecto. Se define el objetivo final del trabajo fin de máster y se describe las herramientas utilizadas para la resolución del problema.

A modo de descripción del contexto que envuelve este problema cabe decir:

La pandemia causada por el nuevo coronavirus (SARS-CoV-2) ha derivado en nuevos desafíos en la manera que la radiología apoya el trabajo clínico y presta servicios oportunos.

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

Tal como se ha referido en la Ficha Trabajo Fin de Máster del punto 1.1 de este trabajo, COVID-19 se puede diagnosticar mediante la reacción en cadena de la polimerasa (para detectar material genético del virus) y con una radiografía de tórax. Mientras que pueden pasar algunas horas y, a veces, días antes de que se obtengan los resultados de las pruebas moleculares, las radiografías de tórax se pueden obtener en minutos.

Esto ha hecho que las pruebas de imagen hayan adquirido un papel muy importante en la detección de coronavirus, ayudando al diagnóstico, manejo de los pacientes, determinando la gravedad y guiando su tratamiento (45). “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”. Incluso se acepta la radiografía de tórax como método de triaje en algunos escenarios, destacando la capacidad diagnóstica de Covid-19.

Los servicios de radiodiagnóstico se han tenido que enfrentar a un verdadero reto para poder informar el gran volumen de radiografías de tórax realizadas (45) en plena pandemia. Los recursos sanitarios son limitados y surge la necesidad de agilizar los procesos de detección del virus.

Además, la radiografía de tórax, no sólo nos indican si el paciente tiene o no el coronavirus, sino que además también puede indicarnos la gravedad de la afectación de tórax que sufre el paciente, orientando hacia el tratamiento y diagnóstico.

El aprendizaje profundo, mediante el procesamiento y la clasificación de imágenes médicas, será vital para la ayudar a los sanitarios en el diagnóstico médico además de reducir los tiempos de evaluación.

3.1. DEFINICIÓN DEL PROBLEMA Y OBJETIVOS

Uno de los principales problemas a resolver en este trabajo se deriva por la propia metodología utilizada en la clasificación de las anotaciones SIIM (46).

La explicación dada en el foro de discusión de la propia competición expone que tipo de patología se recoge en cada una de las 4 clasificaciones realizadas.

Podemos encontrar en algunos casos una amplia diversidad de patologías (atípica) y al mismo tiempo ciertos casos de (indeterminada) no implica que no exista la posibilidad de que el paciente tenga Covid -19 pero no tenga una radiografía de patrón clásico de neumonía de Covid-19 (Típica).

Las radiografías de tórax (CXR) de este trabajo, se clasificaron utilizando un esquema de clasificación específico, basado en un documento publicado en la siguiente referencia. (47).

La siguiente figura expone una descripción detalla de cada una de las categorías de la clasificación.

Patrón	Hallazgos	Impresión sugerida
Típico	<p>OVE de distribución bilateral y periférica: +/- Focos de condensación +/- Líneas intralobulillares (patrón crazy-paving) ó OVE multifocales con morfología redondeada: +/- Focos de condensación +/- Líneas intralobulillares (patrón crazy-paving)</p>	<p>"Hallazgos tomográficos (frecuentemente reportados) (altamente sugerentes) (clásicos) de/en neumonía viral COVID-19. El diagnóstico diferencial corresponde a neumonía viral por diferente agente (ej: influenza) y neumonía organizante."</p>
Indeterminado	<p>Ausencia de hallazgos típicos y: Presencia de: OVE multifocales, difusas, perihiliares o unilaterales con o sin condensación, sin una distribución periférica ni morfología redondeada. ó Escasas y pequeñas OVE sin una distribución periférica ni morfología redondeada.</p>	<p>"Hallazgos tomográficos posibles de observar en neumonía viral tipo COVID-19, sin embargo, inespecíficos y que pueden ser manifestación de otro proceso infeccioso o no infeccioso."</p>
Atípico	<p>Ausencia de hallazgos típicos e indeterminados y Presencia de: Condensación lobar o segmentaria única, sin OVE. Nódulos centrolobulillares con morfología de árbol en brote. Cavitación pulmonar Engrosamiento septal interlobulillar liso con derrame pleural</p>	<p>"Hallazgos tomográficos atípicos o escasamente reportados en neumonía viral tipo COVID-19. Se sugiere considerar un diagnóstico alternativo para los hallazgos imagenológicos."</p>
Negativo	<p>Ausencia de hallazgos tomográficos sugerentes de neumonía.</p>	<p>"Tomografía computada sin hallazgos sugerentes de neumonía. Nota: Considerar que en fases precoces de enfermedad COVID-19 pueden no observarse alteraciones tomográficas."</p>

Figura 24. Terminología sugerida para el informe estructurado en COVID-19 en tomografía computada, basada en consenso de la Sociedad Norteamericana de Radiología.

Abreviaciones: OVE = opacidades con densidad en vidrio esmerilado (48)

En resumen, tomando como base la figura anterior, las radiografías de tórax se clasifican en una de las cuatro categorías, que son mutuamente excluyentes:

- **Aspecto típico/ patrón Clásico COVID-19**: Opacidades multifocales bilaterales, periféricas, de morfología redondeada, de distribución predominantemente pulmonar inferior.
- **Aspecto indeterminado**: Alteraciones presentes, pero que no cumplen con los criterios de un patrón clásico ni tampoco de un patrón No-COVID-19. Ausencia de hallazgos típicos Y distribución unilateral, central o de predominio pulmonar superior
- **Aspecto atípico**: Neumotórax, derrame pleural, edema pulmonar, consolidación lobar, nódulo o masa pulmonar solitaria, nódulos diminutos difusos, cavidad, entre otra posible enfermedad torácicas incluidas en esta categoría.
- **Negativo**: No hay opacidades pulmonares.

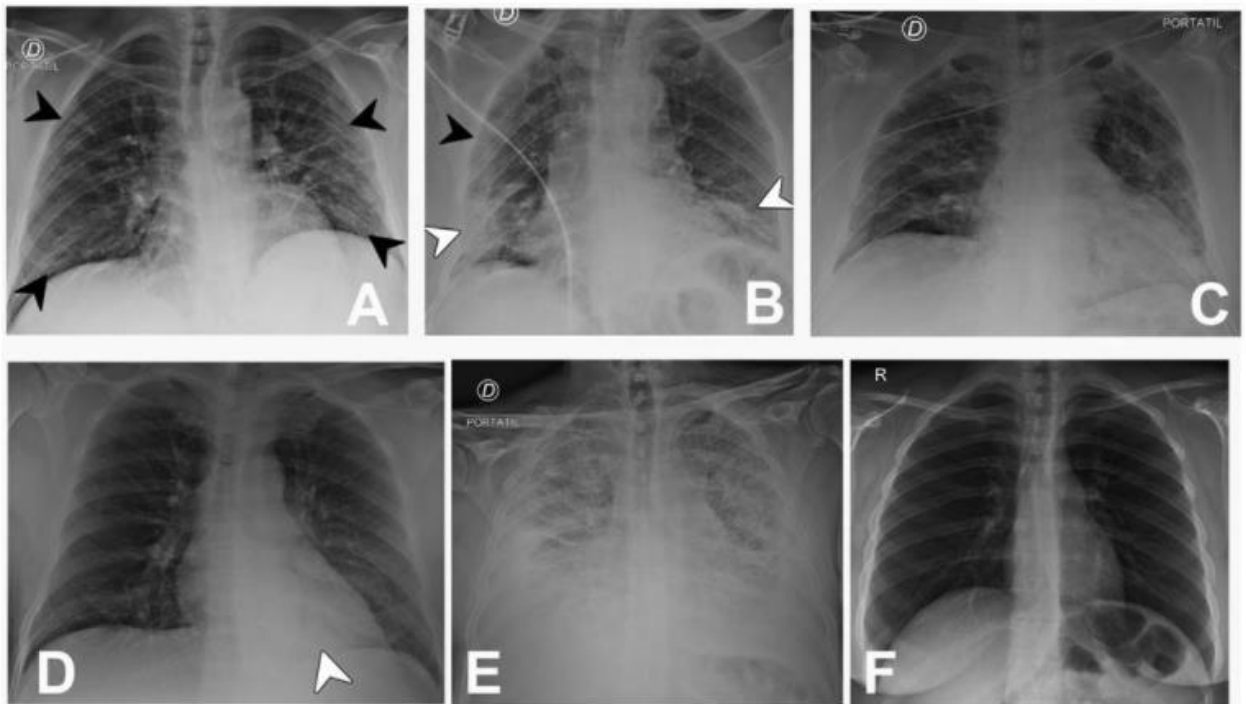


Figura 25. Ejemplos radiografías de tórax de las diferentes categorías (48).

Figura 25 muestra ejemplos de radiografías de tórax para cada patrón del sistema de reporte estructurado BSTI en COVID-19.

- Patrón clásico / típico (A-B): Paciente 1 (A) Opacidades con densidad en vidrio esmerilado bilaterales de distribución periférica e inferior (puntas de flecha negra). Paciente 2 (B): Opacidades bilaterales en vidrio esmerilado (puntas de flecha negra) asociadas a focos de condensación múltiples en la distribución descrita (puntas de flecha blanca)

- Patrón indeterminado (C): Opacidades con densidad en vidrio esmerilado de distribución difusa, sin predominio zonal inferior ni periférico.
- Patrón No-COVID-19/ Atípico (D-E): Paciente (D) con foco de condensación retrocardíaco, unifocal, consistente con neumonía bacteriana. Paciente (E) con signos de edema intersticial y alveolar bilateral, difuso, asociado a derrame pleural bilateral, consistente con insuficiencia cardíaca descompensada.
- Patrón normal (F): Examen sin hallazgos radiológicos sugerentes de neumonía en paciente con COVID-19 confirmado mediante test de PCR.

El propio sistema utilizado por los anotadores puede ser en sí un problema para identificar si una persona tiene o no tiene Covid-19: los anotadores tenían acceso al estado de COVID de cada paciente, pero se les pidió que se adhirieran al sistema de clasificación anterior, independientemente del estado. Por ello, algunos pacientes con COVID negativo tenían radiografías de tórax con apariencia típica. Del mismo modo, algunos pacientes con COVID positivo tenían un aspecto atípico, o eran negativos para la neumonía (sin opacidades pulmonares), porque el sistema de clasificación se basa únicamente en los resultados de la radiografía de tórax.

Otro problema que se detecta en el sistema utilizado por los anotadores es que no se han marcado los recuadros en todos los casos de radiografías con opacidad. Se colocaron recuadros en las opacidades pulmonares, ya sean típicas o indeterminadas. También se colocaron recuadros en algunos hallazgos atípicos, como la consolidación lobar solitaria, los nódulos/masas y las cavidades. **No se colocaron recuadros en los derrames pleurales ni en los neumotórax.** No se colocaron recuadros para la categoría de neumonía negativa.

Otro de los problemas que se encuentra es el elevado número de patologías que recoge la categoría atípica. El no disponer de suficientes imágenes para que un algoritmo de aprendizaje pueda obtener las características específicas de esta clase en la generalización realizada en el entrenamiento, derivará en problemas de rendimiento del modelo.

No disponer de suficientes imágenes como para que se pueda realizar correctamente el proceso de generalización que el algoritmo de aprendizaje obtenide durante el entrenamiento del modelo, es uno de los problemas a resolver, incluso en el resto de las categorías hay pocos datos a excepción de la categoría típica que consta de un mayor número de imágenes.

Otro problema es si no se dispone de clases correctamente balanceadas. El problema de desbalanceo de clases también está directamente relacionado con el buen desempeño del algoritmo de clasificación. Sino es así podemos tener fácilmente problemas de sobreaprendizaje.

Siendo la neumonía por Covid-19 una patología reciente, hay problemas de diferencia de pareceres en el etiquetado de las imágenes por diferentes radiólogos (38) Según dataset referenciado sólo en el 41,8% de los casos hay coincidencia de etiqueta de salida por 3 anotadores en el resto no, este aspecto también se ha de tener en cuenta en el momento de realizar este tipo de análisis, pues también se recogía este problema en el artículo médico referenciado (28).

Las herramientas utilizadas que se detallan en el próximo apartado, también suponen una limitación computacional para el buen desarrollo y rendimiento de un algoritmo de clasificación de una red neuronal convolucional.

Por todo ello en el siguiente subapartado se expone el objetivo del trabajo Fin de Máster.

3.1.1. Objetivos

- Crear una red neuronal convolucional (CNN) que identifique patrones de radiografías con patologías/con anomalías (opacidades) diferenciándolas de radiografías sin patologías/sin anomalías (sin opacidades).
-
- Visualización de los mejores resultados del modelo de la red neuronal convolucional seleccionada.

3.2. HERRAMIENTAS

En este apartado se describe el entorno de trabajo usado, el lenguaje de programación utilizado y el entorno de programación en el cual se desarrolla todo el código de programación de este trabajo Tal como se ha expuesto en el apartado de definición del problema y objetivo, las propias herramientas han supuesto una limitación en la mejora del rendimiento del algoritmo de clasificación objeto de este trabajo

3.2.1. Entorno de trabajo

El trabajo se ha realizado inicialmente en Google Colab, pero por problemas de memoria RAM para algunos procesos y por el tiempo limitado por sesión se termina realizado en mi propio equipo. Mi equipo personal es de marca Lenovo. Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz, con 32 GB de RAM. El sistema operativo es Windows 11 Home Insider Preview.

3.2.2. Lenguaje de programación

El lenguaje de programación elegido para el desarrollo de este proyecto ha sido Python.

Es uno de los lenguajes de programación que mejor conozco para el desarrollo del código necesario y además es considerado uno de los lenguajes más apropiados para el procesamiento de imágenes y la implementación de técnicas de aprendizaje automático. Se debe a que cuenta con iteraciones rápidas permitiendo concentrar los datos y desarrollar algoritmos.

Además, otra ventaja de Python para la Inteligencia Artificial es que permite combinar librerías como NumPy y SciPi, aumentando así los rendimientos de los proyectos. (49)

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

3.2.3. Entorno de Programación

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

Se realizaron las instalaciones necesarias en mi portátil personal para poder crear entorno virtual denominado tensorflow-gpu que me permitía el uso de las librerías necesarias y mayor rendimiento que me permitía el uso de la GPU disponible.

Una vez creado y activado el entorno virtual tensorflow-gpu, se creó 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 a parte del código. En dicho cuaderno se ha ejecutado todo el código para el desarrollo de los algoritmos de redes neuronales convolucionales.

4. DESARROLLO

En este apartado se detalla todo el desarrollo del trabajo fin de máster realizado para obtener un algoritmo de clasificación de una red neuronal convolucional que distinga una radiografía con anomalía de una sin anomalía. **Se ha creado un repositorio en GitHub donde se recoge todo el código realizado en el notebook de trabajo (50)**

Los pasos realizados en el desarrollo de este trabajo han sido los siguiente:

- Conjunto de datos utilizados: de partida se trabaja con los datos de la competición. Las imágenes iniciales están en formato Dicom, por ello se utiliza conjunto de datos publicado donde las imágenes están redimensionadas a formatos jpg/png en varios tamaños. Se detecta la necesidad de datos externos y los datos utilizados se obtienen de la misma plataforma que los datos de la competición.
- Almacenamiento del trabajo: Se detalla los diferentes directorios utilizados.
- Proceso de carga de ficheros/ imágenes y análisis exploratorio EDA de los datos.
- Visualización de las imágenes de las radiografías de tórax con los cuadros que delimitan las anomalías.
- Preprocesamiento de las imágenes previo a su uso en la creación del modelo.
- Proceso de aumento de los datos y partición de los datos.
 - Cuando se usa solo el conjunto de datos de la competición la partición realizada en conjunto de datos de entrenamiento es del (80% del conjunto de imágenes de las radiografías de tórax) y el conjunto de datos de validación es del (20% restante del conjunto de imágenes de las radiografías de tórax)
 - Cuando se hace uso del conjunto de datos externo se reserva un conjunto de datos del 20% de conjunto de datos de la competición para el test final del modelo obtenido. Con el resto de los datos de la unión se realiza partición del 80% para entrenamiento y 20% de los datos para validación.
- Creación de modelos: En este apartado se realiza el entrenamiento y la validación del modelo entrenado. Además de la métrica del Accuracy se utiliza en algunos casos matriz de confusión para el análisis del resultado obtenido.
- Detalle del proceso de carga de la unión de los dos conjuntos de datos (datos de la competición más conjunto de datos externo)
- Partición de los datos del nuevo conjunto de datos tras la unión (80% del total se divide a su vez en 80% para entrenamiento y 20% para validación) y 20% de los datos solo de la competición que se reservan para test final.
- Creación del modelo con los nuevos datos, entrenamiento, validación y test final con los datos ocultos para saber si el nuevo conjunto de datos de entrenamiento es representativo de nuestro conjunto de datos de la competición.

- Mejora de resultados: Reentrenamos nuestro conjunto de datos inicial de la competición con el uso de un modelo preentrenado. El modelo preentrenado es el modelo resultante de la unión de los conjuntos de datos.
- Visualización de los mejores resultados donde se muestra además del accuracy del modelo, la matriz de confusión y una gráfica con la evolución del accuracy y la función de pérdidas del modelo a lo largo de los epochs para datos de entrenamiento y validación.

4.1 CONJUNTO DE DATOS

Para el desarrollo de este trabajo se han utilizado los siguientes conjuntos de datos:

4.1.1. [SIIM-FISABIO-RSNA COVID-19 Detection](#)

“Este conjunto de datos” (51) 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.

Se lanza como una competición de la cual soy partícipe. En esta competencia se debía identificar anomalías de COVID-19 en radiografías de tórax. En particular, categorizará las radiografías como negativas, típicas, indeterminadas o atípicas.

EL organizador líder de esta competición **es la Sociedad de Informática por Imágenes en Medicina (SIIM)**, es la responsable de promover la informática de imágenes médicas a través de la educación, la investigación y la innovación. SIIM se ha asociado con la Fundación para el Fomento de la Salud y la Investigación Biomédica de la Comunitat Valenciana (FISABIO), el Banco de Datos de Imágenes Médicas de la Comunitat Valenciana (BIMCV) y la Sociedad Radiológica de Norteamérica (RSNA) para el lanzamiento de esta competición.

Descripción de las otras dos organizaciones anfitrionas de la competición:

- **FISABIO**, Fundación para el Fomento de la Investigación Biomédica y Sanitaria de la Comunitat Valenciana: entidad científica y sanitaria sin ánimo de lucro, cuya finalidad primordial es fomentar, promover y desarrollar la investigación científica y técnica en salud y biomédica en la Comunitat Valenciana.
- **Sociedad Radiológica de América del Norte (RSNA)**, es una organización sin fines de lucro que representa a 31 subespecialidades radiológicas de 145 países de todo el mundo. RSNA promueve la excelencia en la atención al paciente y la prestación de atención médica a través de la educación, la investigación y la innovación tecnológica.

La base de datos recibe el nombre de: BIMCV-COVID19 fue publicada originalmente por el Banco de Datos de Imagen Médica de la Comunitat Valenciana (BIMCV) en cooperación con la Fundación para la Promoción de la Salud y la Investigación Biomédica de la Comunitat Valenciana (FISABIO) y la Consejería de Innovación., Universidades, Ciencia y Sociedad Digital (Generalitat Valenciana), sin embargo, las imágenes se volvieron a anotar completamente utilizando diferentes tipos de anotaciones. Los usuarios de estos datos deben cumplir con el Acuerdo de uso de investigación de conjunto de datos BIMCV-COVID19 (52) . *Documento de referencia:* (53).

El tamaño total de los datos a descarga es de 128,51GB. Comprende de:

- 7597 imágenes en formato DICOM, dividido en conjuntos de datos train y test. Al ser una competición las 1.263 imágenes de test no están etiquetadas a la espera de la predicción por el participante de la competición. Por lo que se dispone sólo de 6334 imágenes etiquetadas.
- Archivos:
 - train_study_level.csv: los metadatos de nivel de estudio del train, con una fila para cada estudio, incluidas las etiquetas correctas.
 - Columnas:
 - id - identificador de estudio único.
 - Negative for Pneumonia - 1 si el estudio es negativo para neumonía, 0 en caso contrario.
 - Typical Appearance - 1 si el estudio tiene esta apariencia, 0 en caso contrario.
 - Indeterminate Appearance - 1 si el estudio tiene esta apariencia, 0 en caso contrario.
 - Atypical Appearance - 1 si el estudio tiene esta apariencia, 0 en caso contrario.
 - train_image_level.csv: los metadatos de nivel de imagen del train, con una fila para cada imagen, incluidas las etiquetas correctas y los cuadros delimitadores en un formato de diccionario. Algunas imágenes tanto de test como de train tienen varios cuadros delimitadores.
 - Columnas:
 - id - identificador de imagen único.
 - boxes - cuadros delimitadores en formato de diccionario de fácil lectura.
 - label - la etiqueta de predicción correcta para los cuadros delimitadores proporcionados.
 -

4.1.2. [SIIM COVID-19: Resized to 512px JPG](#)

Tras la publicación de resultados de la competición en el foro de discusión, 22 de agosto de 2021, (54) entre los links de destaque, está la publicación

de imágenes de la competición redimensionadas en jpg/png y diferentes tamaños (55). Utilizo las imágenes redimensionadas en jpg de tamaño 512x512píxeles. (56)

Se descargan 152,21 MB con las carpetas de imágenes train, test y archivo meta.csv. El archivo meta.csv se compone de las columnas: image_id, dim0, dim1, split.

4.1.3. [COVID-19 Radiography Database](#)

Conjunto de datos también publicado en la plataforma de Kaggle. (57)

La base de datos 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 Tawsifur Rahmany y su creación se remonta al 29 de marzo de 2021.

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

El tamaño del conjunto de datos es de 775,92MB y está formado por 21.165 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 con opacidad torácica (infección no Covid-19)
- Imágenes de rayos X de tórax normal.

Además de las carpetas de imágenes en formato png diferenciadas para cada una de las categorías en tamaño 299x299, se aportan los siguientes archivos con los metadatos de las imágenes:

- COVID.metadata.xlsx
- Lung_Opacity.metadata.xlsx
- Normal.metadata.xlsx
- Neumonía viral.metadata.xlsx

4.1.4. [SIIM COVID-19: Resized to256px PNG](#)

Para poder realizar la unión de datos SIIM Covi-19 (51) con el conjunto de datos COVID-19 Radiography Database (57), utilizo conjunto de datos redimensionado al mismo formato png y con el mismo tamaño. Fuente: (58)

Se descargan 223,1 MB con las carpetas de imágenes train, test y archivo meta.csv. El archivo meta.csv se compone de las columnas: image_id, dim0, dim1, split.

4.2 ALMACENAMIENTO DE LOS DATOS

Inicialmente se crea como directorio de trabajo una carpeta llamada "TFM"

Dentro de la carpeta de trabajo se crean a su vez tres subcarpetas con la denominación de "Fuentes", "Fuentes_1" y "Fuentes_C"

Una vez descargados el conjunto de datos de Kaggle "SIIM-FISABIO-RSNA COVID-19 Detection" y "SIIM COVID-19: Resize to 512px JPG", se almacenan en la carpeta "Fuentes":

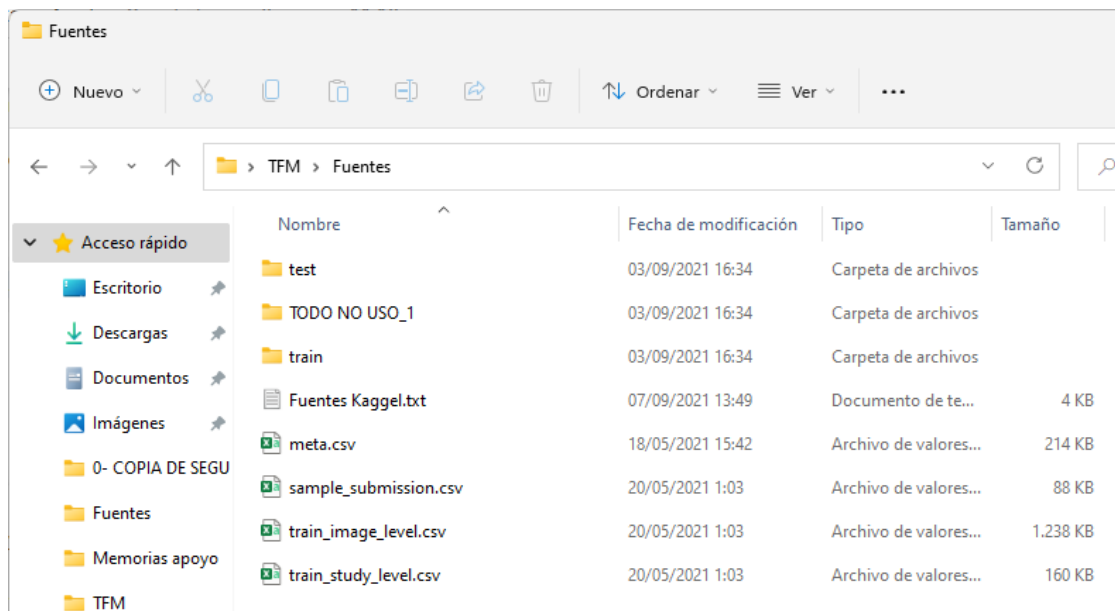


Figura 26. Imagen carpeta Fuentes creada para el almacenamiento de los datos.

Los conjuntos de datos descargados de Kaggle "COVID-19 Radiography Database" y "SIIM COVID-19 Resized to 256px PNG" más los archivos .csv de la carpeta "Fuentes" se almacenan en la carpeta "Fuentes_1".

Dentro del directorio ".TFM/Fuentes_1" de la carpeta se crea a su vez la subcarpeta "train_new" con la unión de todas las imágenes de radiografías de tórax de los conjuntos de datos descargados:

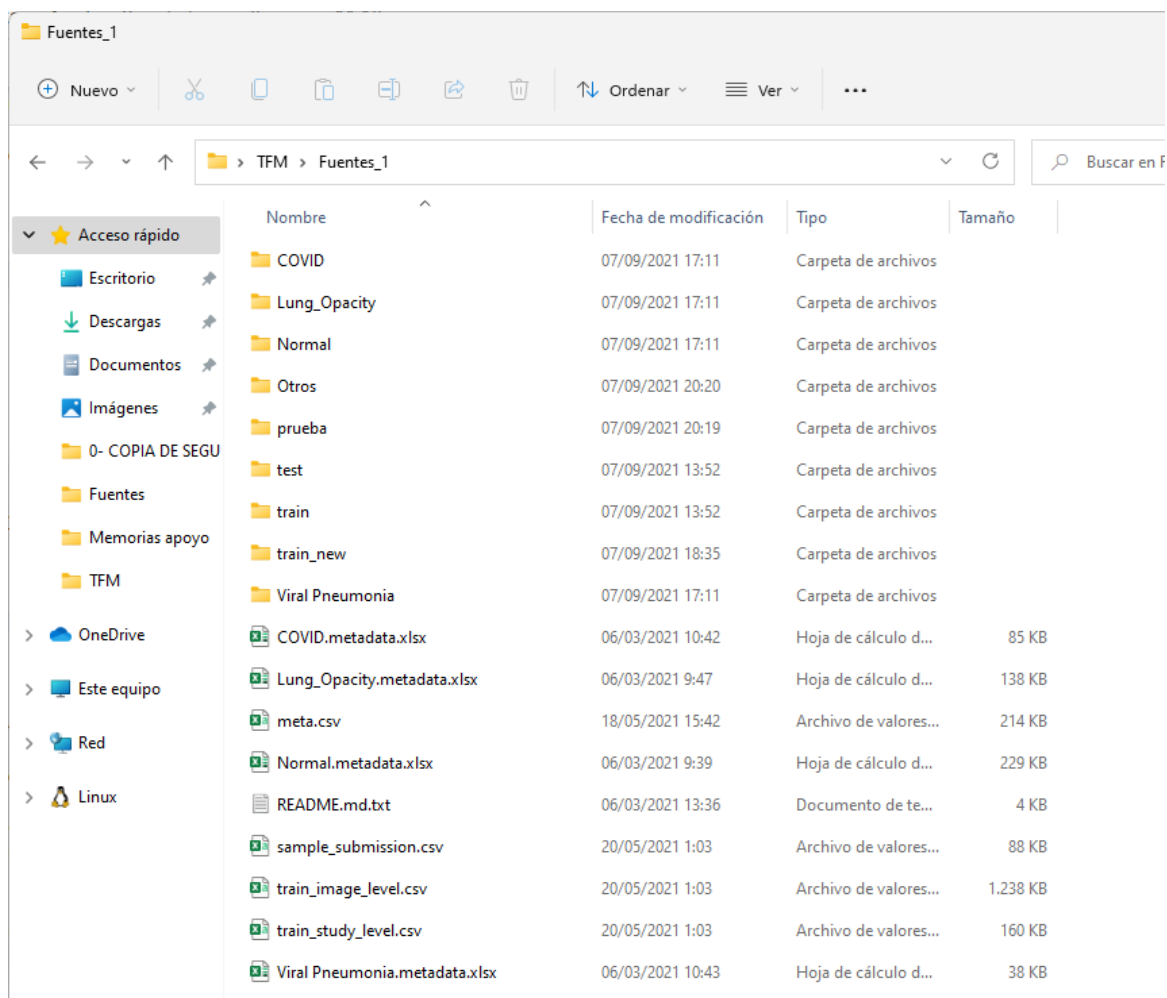


Figura.27. Imagen carpeta Fuentes_1 creada para el almacenamiento de los datos.

EL directorio ".TFM/Fuentes_C" es el que utilizamos para la unión de las bases de datos. Decir que, para poder realizar un modelo utilizando las dos bases de datos es necesario hacer una partición inicial de imágenes train y test de los datos originales de la competición "SIIM COVID-19: Resized to 256px PNG". De esta forma reservamos un conjunto de datos test, que no son entrenados para la validación final del modelo obtenido con la unión de las bases de datos. Utilizo las librerías "os, glob, shutil y pandas" para la creación, movimientos de imágenes entre directorios, copia de directorios y creación.

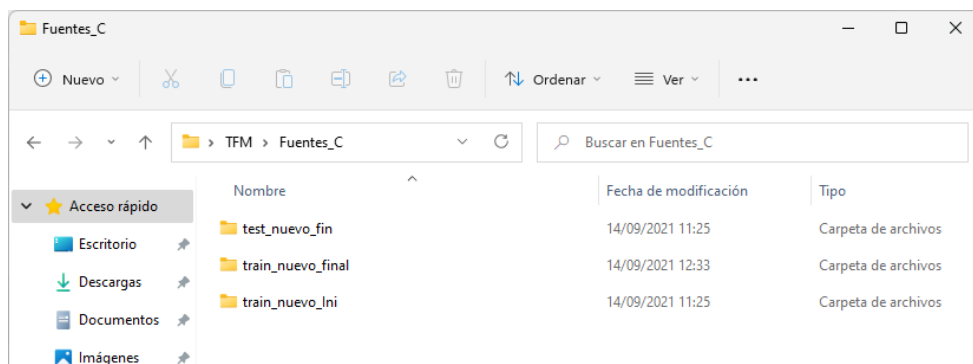


Figura.28. Imagen carpeta Fuentes_C creada para el almacenamiento de los datos modelo Unión BBDD..

4.3 CARGAR LOS DATOS (FICHEROS CSV, CARPETAS FICHEROS DE IMÁGENES) ANÁLISIS EXPLORATORIO Y PREPARACIÓN DE LOS DATOS

4.3.1. Solo con los conjuntos de datos aportados por la Competición

Todo lo que se detalla a continuación se realiza en el punto 2. del notebook del trabajo (50) .

Tras importar las librerías necesarias se procede a un primer análisis exploratorio de los datos o EDA. Es uno de los primeros pasos que hay que utilizar previo a la utilización de las técnicas de aprendizaje automático El objetivo de EDA es saber más de los datos

Para la visualización de las imágenes de las radiografías de tórax, así como el recuento de archivos disponibles en cada una de las carpetas descargadas utilizamos la librería OpenCV importando el módulo cv2. También será necesario la librería matplotlib y numpy.

OpenCV nos permite leer y manipular imágenes, entre muchas otras características. En el próximo código vemos dentro del ciclo for del directorio establecido previamente, la lectura de la imagen, conversión a escala de grises, redimensionamos la imagen a 224*224 píxeles, añadimos a array previamente creado, para finalmente a través de la función len() poder conocer el total de imágenes añadidas al array. La librería matplotlib nos permite realizar el cambio de color y la visualización de la imagen número 10.

```
[ ] import cv2
import numpy as np
import os
import matplotlib.pyplot as plt

ruta_test="./Fuentes/test"
ruta_train="./Fuentes/train"
test_img=[]
train_img=[]
img_size=224

for img in os.listdir(ruta_train):
    img = cv2.imread(os.path.join(ruta_train,img))
    img_gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    img_gray_resize=cv2.resize(img_gray,(img_size,img_size))
    train_img.append([img_gray_resize])

print(len(train_img))
```

6334

```
▶ plt.figure()
plt.imshow(np.squeeze(train_img[10]))
plt.colorbar()
plt.grid(False)
plt.show()
```

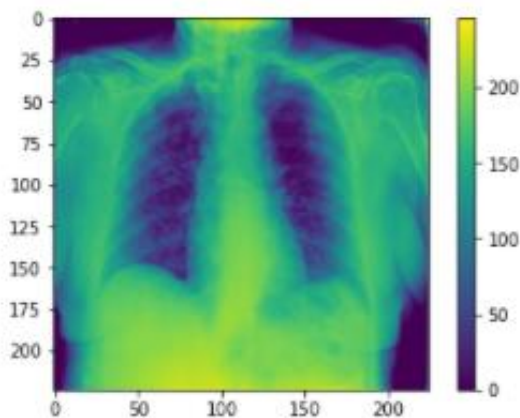


Figura 29. Código notebook TFM: lectura, conversión y visualización imagen 10 del conjunto de datos train – Fuente elaboración propia

Lo mismo se realiza para el conjunto de imágenes de test.

A continuación, se procede a lectura de los archivos .csv proporcionados en el conjunto de datos.

Para cargarlos utilizamos la función de pandas: "pd.read_csv()".

Para cada uno de los archivos train_image_level.csv, train_study_level.csv, sample_submission.csv, realizamos los siguientes pasos:

- Cargamos los datos: con la función `pd.read_csv()`.
- Vemos la estructura del data frame con la función `“.head()”`, utilizamos `“display()”` para realizarlo en una misma línea de código.
- Utilizamos `“.shape()”` para poder ver el total de filas y columnas del data frame.

Se precisa realizar los siguientes cambios en el data frame cargado del archivo `train_study_level.csv`.

- Eliminar la extensión `“_study”` de la columna `“id”`, lo hacemos a través de función `“.str.replace('_study', '')”`
- Renombrar la columna usando `“.rename()”`

Creamos data frame `df_train` con la unión de los data frames creados con `train_image_level.csv` y `train_study_level.csv`, usando la función `“.merge()”`.

En el data frame `df_train` realizamos los siguientes cambios:

- Creamos la columna con la etiqueta de salida a través de la función `“.loc[]”`
- Eliminamos las columnas innecesarias a través de la función `“.drop()”`
- Sustituimos la extensión `‘_image’` por `‘.jpg’` de la columna `‘id’` a través de la función `‘.str.replace()’`.

El último archivo que cargamos es `meta.csv`:

- Lo cargamos al igual que los anteriores con pandas `“pd.read_csv()”`.
- Creamos nueva columna `“id”`, con la unión de la columna `“image_id”` + `“.jpg”`

Por último, para obtener el data frame `df_train` final realizamos la unión con el último dataset cargado, usando de nuevo la función `“.merge()”`

```
[ ] df_train = df_train.merge(df_size, on='id')
df_train.head(5)
```

	id	boxes	label	StudyInstanceUID	study_label	image_label	image_id	dim0	dim1	split
0	000a312787f2.jpg	[{"x": 789.28836, "y": 582.43035, "width": 102... opacity 1 789.28836 582.43035 1815.94498 2499...		5776db0cec75	typical	opacity	000a312787f2	3488	4256	train
1	000c3a3f293f.jpg	NaN	none 1 0 0 1 1	ff0879eb20ed	negative	none	000c3a3f293f	2320	2832	train
2	0012ff7358bc.jpg	[{"x": 677.42216, "y": 197.97662, "width": 867... opacity 1 677.42216 197.97662 1545.21983 1197...		9d514ce429a7	typical	opacity	0012ff7358bc	2544	3056	train
3	001398f4ff4f.jpg	[{"x": 2729, "y": 2181.33331, "width": 948.000... opacity 1 2729 2181.33331 3677.00012 2785.33331		28dddc8559b2	atypical	opacity	001398f4ff4f	3520	4280	train
4	001bd15d1891.jpg	[{"x": 623.23328, "y": 1050, "width": 714, "he... opacity 1 623.23328 1050 1337.23328 2156 opaci...		dfd9fdd85a3e	typical	opacity	001bd15d1891	2800	3408	train

```
[ ] print(df_train.shape)

(6334, 10)
```

Figura 30. Código notebook TFM: data frame final `df_train`.

Tras la creación del data frame final, procedemos al recuento y visualización de la etiqueta de salida `“study_label”`. Punto 2.1 del notebook de trabajo.

Para poder realizar este apartado utilizamos la librería de visualización `plotly` y `seaborn` además de utilizar la función `“.groupby().size()”` para agrupar por la columna de la etiqueta de salida y ordenar por tamaño.

```
[ ] df_train.groupby(['study_label']).size().reset_index(name='Total_clases')
```

	study_label	Total_clases
0	atypical	483
1	indeterminate	1108
2	negative	1736
3	typical	3007

```
[ ] f, ax = plt.subplots(figsize=(7, 3))
sns.countplot(y="study_label", data=df_train, color="blue")
```

<AxesSubplot:xlabel='count', ylabel='study_label'>

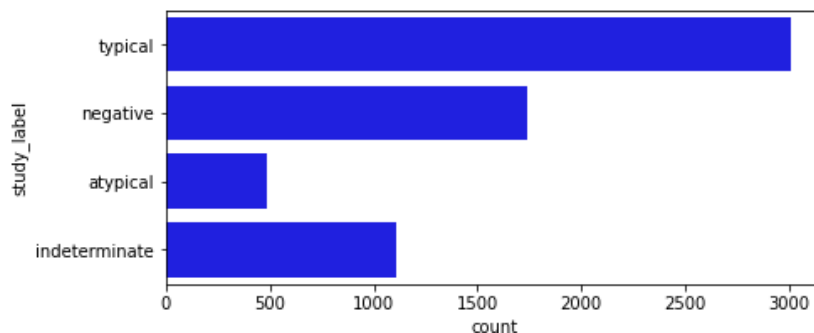


Figura 31. Código Notebook TFM. Recuento etiqueta de salida "study_label".

Ya podemos ver que contamos con pocos datos, solo son 6.334 imágenes de radiografías de tórax y además las clases están totalmente desbalanceadas (8% atípica, 17% indeterminada, 27% negativa y 47% típica) por lo que deberemos recurrir a diferentes técnicas para poder subsanar este problema.

Esto nos lleva a la situación de plantearnos otras posibles soluciones alternativas. En base al artículo médico (28) detallado en el punto 2.3 del Estado de Cuestión de la Memoria del Trabajo Fin de Máster, se concluye que las redes neuronales convolucionales pueden ser de gran ayuda en la creación de modelos binarios, para la detección de anomalías o no anomalías en las radiografías de tórax. Por lo que procedemos a realizar este análisis en nuestros datos.

Realizamos recuento y visualización de la etiqueta de salida "image_label". Punto 2.2 del notebook de trabajo. Que nos determina si la radiografía de tórax tiene opacidad o no.

```
[ ] df_train.groupby(['image_label']).size().reset_index(name='Total clases')
```

	image_label	Total clases
0	none	2040
1	opacity	4294

```
[ ] f, ax = plt.subplots(figsize=(7, 3))
sns.countplot(y='image_label', data=df_train, color="blue")
```

<AxesSubplot:xlabel='count', ylabel='image_label'>

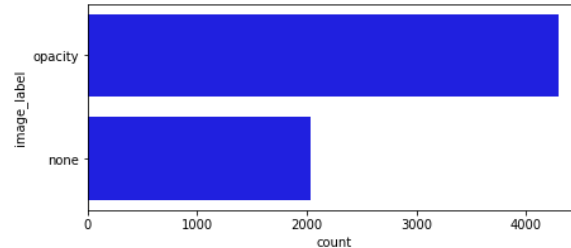


Figura 32. Código Notebook TFM. Recuento etiqueta de salida "image_label".

Se detecta que hay un error en la columna "image_label", ya que si vemos en el recuento realizado por la columna de "study_label", la clasificación de sin opacidad es de (negative=1.736 casos) mientras que aquí sin opacidad (none=2.040) por lo que hay un error de 304 casos con opacidad que estarían mal clasificados. Además, tal como se explica en el problema derivado por el propio sistema utilizado por los anotadores de la competición punto 3.1 de la Memoria del Trabajo Fin de Máster, existen patologías atípicas (los derrames pleurales y en los neumotórax, en las que no se colocaron recuadros y se consideran en la columna "image_label" sin opacidad) por lo que el error está en estas patologías de categoría atípica.

Por ello se crea una columna tomando como referencia "study_label" en la que será opacidad todas las categorías a excepción de "negative ". Para la creación de esta nueva columna, que llamo "label_opacity" utilizo la función condicional de numPy junto con where."np.where()". Punto 2.3 notebook de trabajo.

```
[ ] df_train.groupby(['label_opacity']).size().reset_index(name='Total clases')
```

	label_opacity	Total clases
0	none	1736
1	yes	4598

```
[ ] f, ax = plt.subplots(figsize=(7, 3))
sns.countplot(y='label_opacity', data=df_train, color="blue")
```

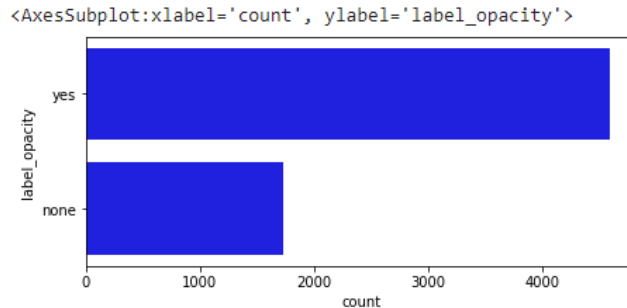


Figura 33. Código Notebook TFM. Recuento etiqueta de salida "label_opacity".

Vemos que en este caso también los datos están desbalanceados, pues solo se dispone de un 27% de los datos sin patología /sin opacidad.

Otra posible solución de clasificación binaria es considerar realizar una agrupación de datos diferenciado si la radiografía de tórax tiene neumonía de Covid-19 típica o no la tiene, para ello creamos nueva etiqueta de salida denominada: "label_n_Covid_19".

En este caso creamos una función para que por fila identifique la etiqueta de la columna "study_label" y cruce con un diccionario que creamos previamente. Posteriormente a través la función ".apply()" aplicamos la función de cruce creada y realizamos recuento y visualización. Punto 2.4. del notebook de trabajo.


```
[ ] def map_values(row, values_dict):
    return values_dict[row]

values_dict = {'typical': 'Yes', 'negative': 'None', 'indeterminate': 'None', 'atypical': 'None'}
df_train['label_n_Covid_19'] = df_train['study_label'].apply(map_values, args = (values_dict,))
```

```
[ ] df_train.groupby(['label_n_Covid_19']).size().reset_index(name='Nuevas clases')
```

	label_n_Covid_19	Nuevas clases
0	None	3327
1	Yes	3007

```
[ ] f, ax = plt.subplots(figsize=(7, 3))
sns.countplot(y='label_n_Covid_19', data=df_train, color="blue")
```

<AxesSubplot:xlabel='count', ylabel='label_n_Covid_19'>

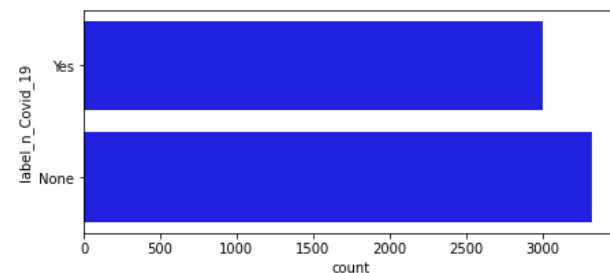


Figura 34. Código Notebook TFM. Recuento etiqueta de salida "label_n_Covid_19U".

En este caso, los datos están más balanceados (47% con neumonía de Covid-19 típica versus 53% que no tienen neumonía de Covid-19 típica). El problema es que dentro del 53% de imágenes sin neumonía de Covid-19 hay casos con multitud de patologías e incluso casos sin patologías. En este caso las diferencias han de ser muy importantes para que con estos datos el modelo creado pueda extraer características precisas en la fase de generalización del entrenamiento que permita al modelo obtener un buen rendimiento.

Como vemos que existe un número reducido de imágenes y es posible que la situación de clasificación binaria con neumonía de Covid-19 típica vs resto no obtenga el resultado esperado, también se realiza esta última clasificación. Esta clasificación se deduce de la explicación expuesta en la descripción de las patologías que recogen cada categoría descrita en el resumen del punto 3.1. Definición del problema y objetivos” de la Memoria del trabajo fin de máster. De la lectura del resumen del sistema utilizado por los anotadores de la competición se puede extraer una clasificación con tres salidas posibles: Negativo (sin opacidad), Neumonía típica (Con opacidad bilateral: recuadros en ambos lados de la radiografía), esta opacidad bilateral también se puede ver el siguiente apartado 4.4 “Visualización de las imágenes train con cajas que identifican anomalías” – neumonía de Covid-19) y Otros (contemplado los casos de Atípicos e indeterminados que tienen alguna opacidad, en el apartado 4.4. casualmente estos casos tienen una opacidad unilateral). Punto 2.6 del notebook de trabajo.

```
[ ] df_train.groupby(['label_opacity_3l']).size().reset_index(name='Nuevas clases')
```

	label_opacity_3l	Nuevas clases
0	None	1736
1	Yes	3007
2	otros	1591

```
[ ] f, ax = plt.subplots(figsize=(7, 3))
sns.countplot(y='label_opacity_3l', data=df_train, color="blue")
```

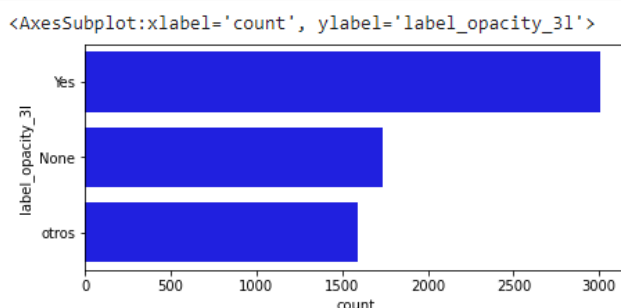


Figura 35. Código Notebook TFM. Recuento etiqueta de salida "label_opacity_3l".

Seguimos teniendo clases desbalanceadas: la clase mayoritaria es con neumonía de Covid-19 con un 47%, clase negativa con un 27% y conjunto de datos de indeterminados más atípicos que denominamos otros con un 25%. Con esta partición se permitirá realizar una mejor clasificación de las clases típicas y negativa (se compara opacidad bilateral y sin opacidad) pero hemos de ver si tenemos suficientes imágenes para la correcta clasificación de la tercera categoría.

4.3.2. Unión de bases de datos de los 4 conjuntos de datos presentados en el punto 4.1 Conjunto de datos de la Memoria del Trabajo Fin de Máster.

Todo lo que se detalla a continuación se recoge en punto 7 y 8 del notebook de trabajo (50).

Para hacer esta unión se ha creado nueva carpeta de fuentes que hemos denominado "Fuentes_C". En esta tenemos todos los elementos necesarios para poder realizar la carga de los datos, previo a la partición de los datos iniciales de la competición, en la que debemos reservar un 20% de los datos SIIM COVID-19 para realizar la validación final de los datos de test.

Los pasos realizados se describen a continuación:

Primero se cargan todos los datos en un único data frame (Punto 7 Notebook – Trabajo que nos permitirá crear las columnas con las etiquetas de salida):

- Repetimos todo el proceso detallado en el apartado anterior “Solo con los conjuntos de datos aportados por la Competición”
- Posteriormente, cargamos los metadatos de la nueva base de datos, en este caso los archivos tienen formato “.xlsx” por lo que para la lectura de los datos también utilizaremos pandas, pero en este caso: “.pd.read_excel()”.
- En los cuatro archivos .xlsx hemos de hacer el mismo proceso:
 - Lectura y carga con “.pd.read_excel()”.
 - Modificar columna “File Name”, añadiendo la extensión “.png”
 - Renombrar la columna “File Name” a “id” para posteriormente poder realizar la unión de data frames.
 - Crear nueva etiqueta de salida con nombre “study_label”
 - Quedarnos finalmente con las columnas que necesitamos.
 - Tras realizar estos cambios procedemos a la unión de data frames con el uso de la función “.concat()” y nuestro data frame final tendrá el nombre de “df_new”.

```
[ ] df_new=pd.concat([df_train_1_final, df_covid,df_LO,df_NORMAL,df_VIRALNEU], axis=0)
df_new
```

	id	study_label
0	000a312787f2.png	typical
1	000c3a3f293f.png	negative
2	0012ff7358bc.png	typical
3	001398f4ff4f.png	atypical
4	001bd15d1891.png	typical
...
1340	Viral Pneumonia-1341.png	VIRALNEU
1341	Viral Pneumonia-1342.png	VIRALNEU
1342	Viral Pneumonia-1343.png	VIRALNEU
1343	Viral Pneumonia-1344.png	VIRALNEU
1344	Viral Pneumonia-1345.png	VIRALNEU

27499 rows x 2 columns

Figura 36. Código Notebook TFM. Data frame unión bases de datos “df_new”.

Realizamos un recuento por etiqueta de salida "study_label"

```
[ ] df_new.groupby(['study_label']).size().reset_index(name='Total clases')
```

	study_label	Total clases
0	LO	6012
1	NORMAL	10192
2	VIRALNEU	1345
3	atypical	483
4	covid_19	3616
5	indeterminate	1108
6	negative	1736
7	typical	3007

```
[ ] f, ax = plt.subplots(figsize=(7, 3))
sns.countplot(y="study_label", data=df_new, color="blue")
```

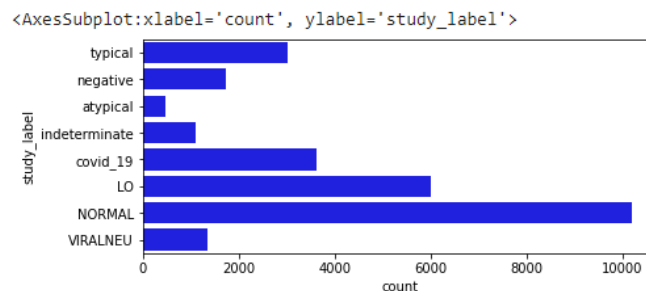


Figura 37. Código Notebook TFM. Recuento y Visualización "study_label" del nuevo conjunto de datos tras la unión de data frames.

Por lo que vemos tenemos un total de 27.499 imágenes con un total de 8 categorías posibles y con datos muy desbalanceados.

Esto nos lleva a un planteamiento binario en el que la clasificación puede ser:

- **Con opacidad (con anomalía) o sin opacidad (sin anomalía)**
- **Con neumonía de Covid-19 o sin ella.**

Procedemos a realizar la creación de las nuevas etiquetas de salida y la visualización final agrupada:

- '2L_opacity_no_opacity' es columna creada que clasifica con opacidad o sin ella.
- '2L_covid_no_covid' es la columna creada que clasifica con neumonía de Covid-19 o sin ella.

Por último, se procede a la creación “del nuevo conjunto de datos train” (unión BBDD) y conjunto de datos test oculto (20% del conjunto de los datos inicial de la competición) – Punto 8 Notebook Trabajo)

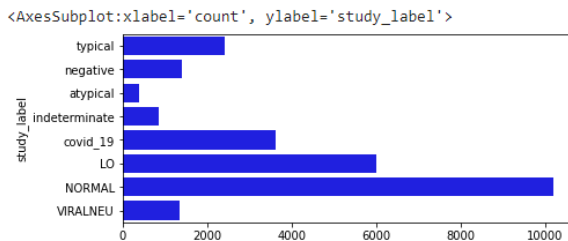
- Tenemos el data frame “df_new” con todos los datos para hacer la unión de datos con función “merge()”, tras hacer la partición de imágenes.
- Tras la creación de directorios y movimientos de imágenes explicado en el punto 4.2 de la Memoria del trabajo, utilizamos la función “os.listdir()” para conocer el número real de imágenes que vamos a entrenar. Con la función “.len()” vemos que contamos con 26.232 imágenes.
- A través de la función de pandas “pd.DataFrame()”, convertimos la lista de archivos del directorio en un data frame.
- Con la función “merge()” unimos al id el resto de las columnas con las etiquetas de salida necesarias. Tengo que subsanar un error, porque no me junta las etiquetas de salida de las 10.192 imágenes con la categoría “Normal”. Lo soluciono sustituyendo los nulos por los valores correspondientes, utilizando la función “.fillna()”
- Finalmente, con el uso de “groupby()” vemos el total por clases de salida.

TFM: SIIM-FISABIO-RSNA COVID-19 Detection – Detección de anomalías en radiografías de tórax

```
[ ] df_new_train_final.groupby(['study_label']).size().reset_index(name='Total clases')
```

	study_label	Total clases
0	LO	6012
1	NORMAL	10192
2	VIRALNEU	1345
3	atypical	386
4	covid_19	3616
5	indeterminate	866
6	negative	1394
7	typical	2421

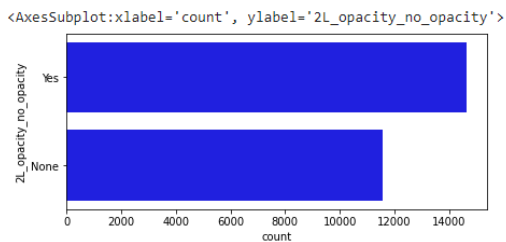
```
[ ] f, ax = plt.subplots(figsize=(7, 3))
sns.countplot(y="study_label", data=df_new_train_final, color="blue")
```



```
df_new_train_final.groupby(['2L_opacity_no_opacity']).size().reset_index(name='Total clases')
```

	2L_opacity_no_opacity	Total clases
0	None	11586
1	Yes	14646

```
[ ] f, ax = plt.subplots(figsize=(7, 3))
sns.countplot(y="2L_opacity_no_opacity", data=df_new_train_final, color="blue")
```



```
[ ] df_new_train_final.groupby(['2L_covid_no_covid']).size().reset_index(name='Total clases')
```

	2L_covid_no_covid	Total clases
0	None	20195
1	Yes	6037

```
[ ] f, ax = plt.subplots(figsize=(7, 3))
sns.countplot(y="2L_covid_no_covid", data=df_new_train_final, color="blue")
```

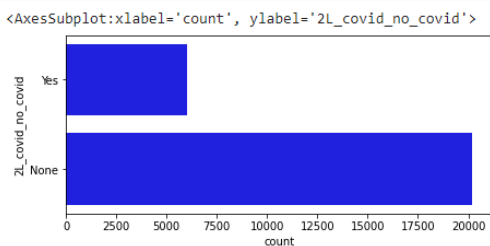


Figura 38. Código Notebook TFM. Recuento y Visualización etiquetas de salida train de la unión bases de datos final (sin el conjunto de datos oculto para test).

En base a los resultados tan desbalanceados al crear la agrupación binaria con neumonía de covid-19 (23% de los datos) versus sin neumonía de Covid-19 (77% de los datos) se opta por la opción en la que los datos están más balanceados. Se realiza la creación de un modelo binario con salida: "con opacidad / con anomalía (56% de los datos)" versus "sin opacidad/ sin anomalía (44% de los datos)" Aunque los datos siguen estando desbalanceados, es el menor desbalanceo de datos que tenemos y además se cuenta con un mayor número de datos, para que el modelo pueda realizar una mejor generalización en la fase de entrenamiento.

- Se carga del mismo modo los datos de test para poder realizar la validación final.
- Finalmente se cuenta con los siguientes totales de imágenes por categoría: atypical:97, indeterminate:242, negative:342, typical:586.
- Para este subconjunto de datos test, el total de imágenes en la clasificación binaria es de: Sin opacidad:342 y con opacidad:925.

4.4 VISUALIZACIÓN IMÁGENES TRAIN CON CAJAS QUE IDENTIFICAN LAS ANOMALÍAS

En este apartado se pretende realizar la visualización de un conjunto de 20 imágenes con sus recuadros que delimitan la opacidad provocada por la anomalía. Este apartado se realiza en el punto 3 del Notebook de trabajo. (50)

Para la visualización de las imágenes de radiografías de tórax del conjunto train, con opacidades y con cajas de recuadro de la opacidad, utilizamos un bucle for y para cargar la imagen utilizamos la librería OpenCV con uso del módulo "cv2.imread()".

En este caso visualizamos un total de 20 imágenes en (5 columnas x 4 filas) con la etiqueta de salida

Por cada imagen se añade:

- El recuadro con ".add_patch(Rectangle())"
- El título con ".set_title()"

```

n = 20
train_dir = './Fuentes/train'
fig, axs = plt.subplots(4, 5, figsize=(20,20))
fig.subplots_adjust(hspace=.2, wspace=.2)
axs = axs.ravel()
for i in range(n):
    img = cv2.imread(os.path.join(train_dir, df_train['id'][i]))
    axs[i].imshow(img)
    if type(df_train['boxes'][i]) == str:
        boxes = literal_eval(df_train['boxes'][i])
        for box in boxes:
            axs[i].add_patch(Rectangle((box['x']*(512/df_train['dim1'][i]), box['y']*(512/df_train['dim0'][i]), box['width']*(512/df_train['dim1'][i]), box['height']*(512/df_train['dim0'][i])))
    else:
        axs[i].set_title(df_train['study_label'][i])
        axs[i].set_title(df_train['study_label'][i])

```

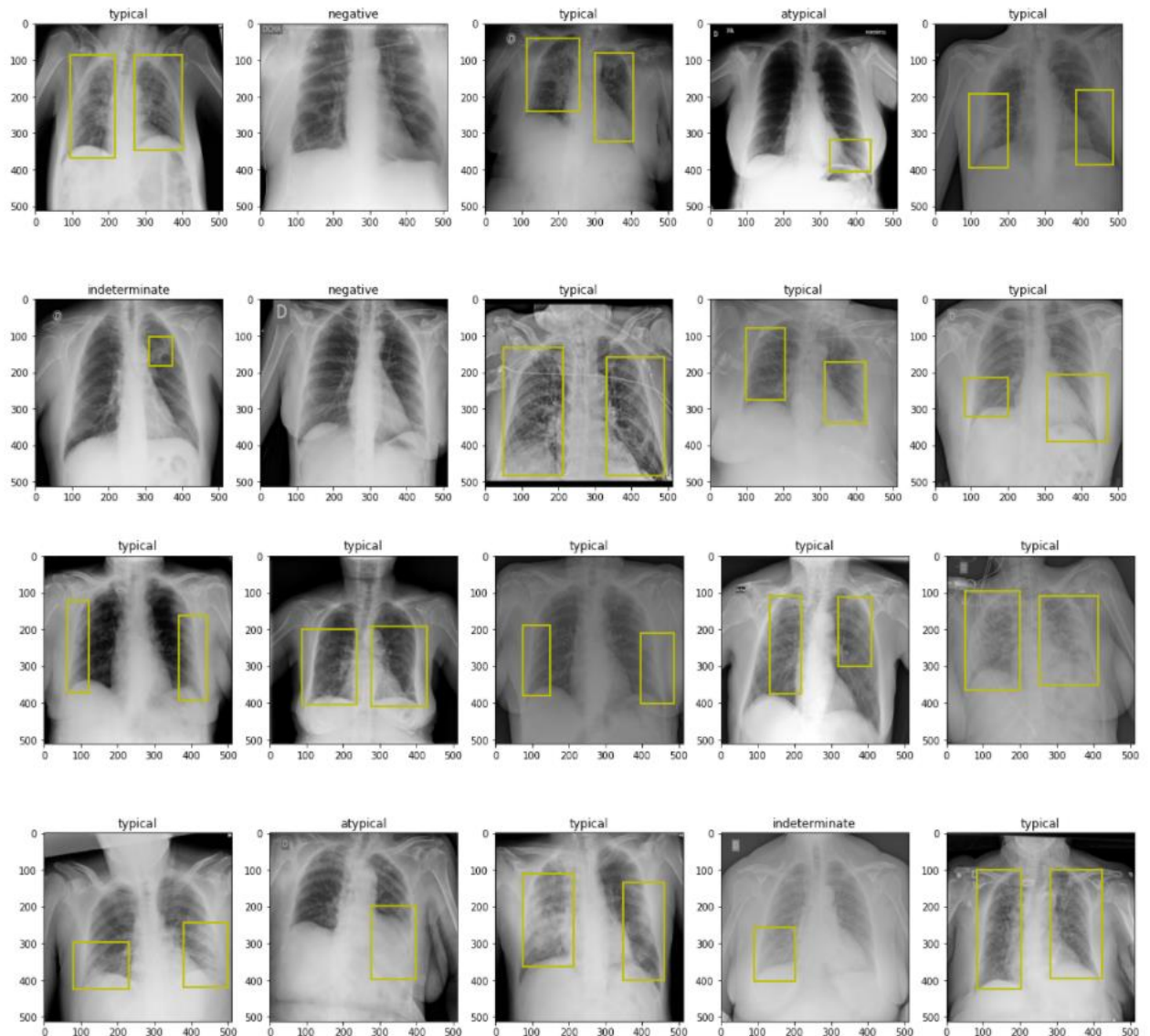


Figura 39. Código Notebook TFM. Visualización imágenes radiografías de tórax conjunto train con cajas que marcan las opacidades.

Lo que podemos ver en estos ejemplos es que en todas las imágenes con neumonía de Covid-19/ típica son bilaterales los recuadros, mientras con en las patologías atípicas e indeterminadas los recuadro son unilaterales.

4.5 PREPROCESAMIENTO DE LOS DATOS

Para poder obtener un buen rendimiento en un modelo de una red de neuronas convolucional es importante que las imágenes utilizadas dispongan de la mejor calidad posible. Es por esta razón por la que realizamos un preprocesamiento de las imágenes de las radiografías de tórax disponibles.

Para realizar el preprocesamiento de las imágenes se ha utilizado el módulo "exposure" de scikit-image. (59)

Se probaron distintas opciones, como "exposure.adjust_gamma()" y "exposure.adjust_log()" de este módulo, pero provocaban distorsiones en las imágenes que podían generar un ruido nada ventajoso para la generalización a realizar en el proceso de entrenamiento.

Finalmente se optó solo por utilizar "skimage.exposure.equalize_hist()". Devuelve la imagen después de la ecualización del histograma: "mejora una imagen con bajo contraste, distribuye los valores de intensidad más frecuentes, en la imagen. La imagen ecualizada tiene una función distribución acumulativa aproximadamente lineal. Esta opción tiene la ventaja de que no precisa parámetros" (60)

Esto se recoge en el punto 4 del notebook de trabajo (50). Como podemos ver en la figura próxima mejora la imagen resultante.

```
[ ] def preprocess_image(img):
    equ_img = exposure.equalize_hist(img)
    return equ_img

im= cv2.imread('./Fuentes/train/000a312787f2.jpg')
im2 = preprocess_image(im)
res = np.concatenate((im/255, im2), axis=1)
plt.imshow(res)
plt.show()
```

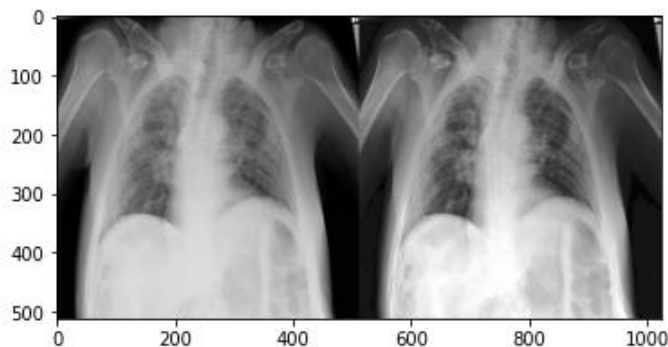


Figura 40. Código Notebook TFM. Visualización imágenes radiografías de tórax tras aplicar función de preprocesamiento.

Otro aspecto para tener en cuenta es que los modelos a crear con redes de neuronas convolucionales pueden funcionar mejor si la entrada del modelo es un vector de imágenes y un vector con las etiquetas de salida. Para poder realizar esta conversión se realiza el siguiente preprocesamiento de imágenes.

Utilizamos la librería "keras.preprocessing" (61) e importamos el módulo "image".

Creamos un ciclo for para realizar el preprocesamiento de todas las imágenes de la carpeta "train".

Dentro del ciclo for se realiza lo siguiente:

- Lectura de la imagen con "image.load_img()".
- Transformar las imágenes en un array: para ello utilizamos la función "img_to_array()".
- Redimensionar las imágenes: Para el primer conjunto de datos contamos con imágenes de dimensiones 512x512 píxeles. Necesitamos redimensionarlas a 299x299 para que el coste computacional sea menor, lo hacemos a través de la función "image.smart_resize()".
- Normalización de las imágenes: los valores de los píxeles se encuentran entre 0 y 255. Por lo que para normalizar dividimos entre 255. De esta manera convertimos los datos en una distribución dentro del intervalo de 0 a 1, que nos puede ayudar a que nuestro algoritmo funcione más rápido.

Finalmente:

- Para convertir las listas de imágenes y etiquetas de salida en un vector: Es necesario realizarla la conversión de imágenes y etiquetas de salida a un array, ya que el formato listas no es aceptado como entrada para los modelos de CNN. Para ello utilizamos la función de pandas np.array()
- Realizar One-Hot-Encoding de la etiqueta de salida: Esta codificación consiste en convertir las etiquetas de salida en una clasificación binaria de 0 y 1. Esta conversión se ha de realizar tanto en etiquetas de train como de validación. Para ello se utiliza de la librería de "sklearn.preprocessing" el módulo "LabelBinarizer ()", se utiliza la función: "LabelBinarizer().fit_transform()".

4.6 AUMENTO Y PARTICIÓN DE DATOS

4.6.1. Aumento de datos:

Tal como se ha detectado en el análisis exploratorio de los datos "EDA" se dispone de pocos datos y además las categorías están desbalanceadas. Este hecho puede ocasionar problemas en el rendimiento del modelo que creemos. El principal problema que se puede originar es el sobreaprendizaje. Para evitar el sobreaprendizaje en la fase de entrenamiento realizamos aumento de los 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 le ayuda a generalizar el modelo y reducir el sobreajuste.

La herramienta utilizada para realizar el aumento de los datos es de la librería "tensorflow.keras.preprocessing.image" el módulo "ImageDataGenerator" (62)

Cuando se realiza la aplicación de un número elevado de transformaciones, los resultados no son positivos.

Finalmente, las transformaciones que se aplican son las que paso a detallar:

- `horizontal_flip = True`. Voltea aleatoriamente la mitad de las imágenes de forma horizontal.
- `zoom_range= 0.1`. Se amplía o aleja la imagen en un rango de -10% y +10%.
- `shear_range=0.05`. Realizar transformaciones de corte de manera aleatoria.
- `brightness_range=[0.8, 1.1]`. Rango para elegir aleatoriamente un valor de cambio de brillo.
- `validation_split=0.2`. Partición de los datos, solo cuando se precise.

4.6.2. Partición de los datos:

Dividiremos nuestro conjunto de datos en dos subconjuntos:

- Conjunto de entrenamiento: constituye las imágenes que se usarán para ajustar y entrenar el modelo.
- Conjunto de datos de validación: imágenes utilizadas para evaluar el modelo final. Dichas imágenes no forman parte del entrenamiento y por lo tanto nos proporcionan una evaluación imparcial ofreciendo el error real que ha cometido el modelo.

Cuando se realiza la unión del conjunto de datos inicial de la competición con conjunto de datos externo, los subconjuntos utilizados son:

- Nuevo conjunto de datos de entrenamiento: la función es la misma que se detalla anteriormente, pero estará formado por imágenes de ambos conjuntos de datos.
- Nuevo conjunto de datos de validación: con la misma función que la detallada con anterioridad y además también ha de estar compuesto por imágenes de ambos conjuntos de datos.
- Conjunto de datos Test final: En este caso está compuesto por un conjunto de imágenes que solo han de ser del conjunto de producción. Imágenes solo del conjunto inicial de la competición. Este conjunto de datos nos servirá tanto para la validación final de modelo alcanzado por la unión de los datos como además nos indicará si el nuevo conjunto de datos de entrenamiento es representativo del conjunto de datos inicial. Para saber si esto es así la precisión del modelo de este conjunto ha de ser muy similar a la alcanzada por el nuevo conjunto de datos de validación.

Se realizan las siguientes particiones:

En los modelos en los que la entrada de datos es una carpeta de imágenes formato jpg/png y etiqueta de salida columna de un data frame:

- En este caso la partición de los datos y la aplicación del aumento de los datos solo en el conjunto train lo aplicamos gracias al método `".flow_from_dataframe()"`.
- La partición de los datos la marcamos dentro las transformaciones de la función `"ImageDataGenerator()"`.

En los modelos cuya entrada es un vector creado con las imágenes y otro con las etiquetas de salida:

- Utilizamos el módulo de `"sklearn.model_selection"` la función `"train_test_split()"`.
- Después para poder aplicar el aumento de los datos en esta partición de los datos de entrenamiento utilizamos el método `".flow()"`

En ambos procesos estamos realizando una partición de los datos de la carpeta de train: 80% para el entrenamiento y 20% para la validación (No tenemos datos test ya que al ser datos de una competición no dan las etiquetas de salida de los datos de test).

Finalmente contamos con las siguientes particiones:

Conjunto de datos iniciales competición:

De las 6334 imágenes de la carpeta train: tenemos 5.068 imágenes para el entrenamiento y 1266 imágenes para la validación tras la partición.

Unión de las 2 Conjuntos de Datos:

Reservamos inicialmente 1267 imágenes de la carpeta train del conjunto inicial de datos de la competición para la validación final, conformarían el denominado "Conjunto de datos Test final".

Nos queda una carpeta del nuevo train con 26.232 imágenes. Con la partición del 80% para "Nuevo conjunto de entrenamiento" y un 20% para "Nuevo conjunto de validación", nos ofrece un total de 20.986 imágenes para entrenamiento y 5.246 para la validación.

4.7 CREACIÓN DE MODELOS

En este apartado se realizan un total de 12 modelos de redes neuronales convolucionales. Se recoge en el punto 6 del Notebook de Trabajo (50).

La realización de cada uno de los modelos tiene una finalidad y **el objetivo del siguiente modelo es siempre la búsqueda de un mejor rendimiento sobre los modelos anteriores, así como subsanar los problemas obtenidos en los anteriores desarrollos.**

A continuación, se detalla un breve resumen de cada uno de los modelos realizados, describiendo: el número de etiquetas de salida, la finalidad y la precisión, así como si existe algún problema en el rendimiento final del mismo:

Subapado.	Nº de etiquetas de salida	Finalidad	Accuracy (Precisión en datos de validación) / problema
4.7.1	4 etiquetas	Se pretende conocer cuál es el rendimiento de un modelo de redes de neuronas convolucionales creado con un total de 9 capas: 1 de entrada 7 ocultas más la capa de salida. En el análisis EDA de los datos se ha detectado el problema de que tenemos pocos datos y además las categorías están desbalanceadas por ello se aplica el aumento de datos en el entrenamiento	45,97% Sobreaprendizaje
4.7.2	4 etiquetas	Dado que el modelo anterior no es suficiente para poder obtener un buen rendimiento con los datos de partida y conocemos que el rendimiento puede mejorar con la aplicación de transfer learning, realizamos esta mejora con la aplicación de transfer learning del Modelo Xception, manteniendo	1º Entrenamiento: 51,74% 2º Entrenamiento: 59,32% Sobreaprendizaje

		aumento de datos en el entrenamiento.	
4.7.3	4 etiquetas	Otra posible mejora del rendimiento en los modelos de las redes de neuronas convolucionales se puede obtener cuando las imágenes de entrada y la etiqueta de salida son un vector por ello se realiza este preprocesamiento de los datos y se mantiene el transfer learning con el modelo Xception, así como el aumento de datos en el entrenamiento.	1º Entrenamiento: 57,93% 2º Entrenamiento: 63,30% Sobreaprendizaje
4.7.4	4 etiquetas	La finalidad en estos modelos es conocer si la aplicación del aumento de datos en el entrenamiento es efectiva o por el contrario no está aportando una mejora en el rendimiento del modelo. Por ello aplicamos la base del modelo anterior, pero sin el aumento de datos.	1º Entrenamiento: 53,91% 2º Entrenamiento: 59,34% Mayor Sobreaprendizaje versus modelo anterior.
4.7.5	4 etiquetas	Se pretende ver si cuando se hace la carga del Modelo Xception con los pesos propios del modelo nos ayudan en la obtención de un buen rendimiento o por el contrario podemos obtener un mejor rendimiento si entrenamos el modelo Xception desde cero. Por ello realizamos modelo tomando como base los datos del modelo del apartado 4.7.3 pero entrenamos desde cero el modelo Xception	1º Entrenamiento: 47,91% 2º Entrenamiento: 47,91% Sobreaprendizaje
4.7.6	4 etiquetas	Tomando como referencia el artículo médico detallado en el apartado 2.3 de la Memoria del Trabajo fin de máster, existen otros modelos de transfer learning como muy buenos rendimientos en la clasificación binaria de radiografías de tórax. Uno de estos modelos es el VGG16. La finalidad en este modelo es conocer el rendimiento de la aplicación de este en estos datos. También se aplica el aumento de datos en el entrenamiento.	1º Entrenamiento: 18,88% 2º Entrenamiento: 45,97% Sobreaprendizaje
4.7.7	4 etiquetas	Dado que el modelo anterior no es suficientemente robusto para la obtención de un buen resultado con nuestros datos de partida, en base al mismo artículo se utiliza el modelo de transfer learning que ofrece el mayor rendimiento en el estudio realizado. Es el modelo Inception V3. También se aplica aumento de los	1º Entrenamiento: 26,20% 2º Entrenamiento: 26,20%

		datos en el entrenamiento.	
4.7.8	3 etiquetas	Se ha detectado en los anteriores modelos que no contamos con suficientes datos para poder obtener un modelo con un buen rendimiento. Por ello la finalidad de este modelo es ver si podemos mejorar el rendimiento si se reduce la clasificación a 3 etiquetas de salida basándonos en la clasificación realizada tras la lectura del resumen del sistema utilizado por los anotadores de la competición descrito en el punto 3.1 de la Memoria del trabajo y la agrupación resultante que se describe en el punto 4.3 de la Memoria. Se utiliza la base del mejor modelo que es con la utilización de transfer learning Xception y con aumento de los datos en entrenamiento.	1º Entrenamiento: 54,90% 2º Entrenamiento: 62,95% Sobreaprendizaje
4.7.9	2 etiquetas	Tras la aplicación del mejor modelo con tres etiquetas y comprobar que tenemos un bajo rendimiento con sobreaprendizaje, hemos de realizar un modelo donde se reduzca la clasificación a modelo binario. La primera clasificación estudiada es la que se considera que el modelo podrá realizar una mejor generalización en la fase de entrenamiento. Se compara radiografías con opacidad versus radiografías sin opacidad. El modelo ha de ser lo suficientemente potente para poder clasificar con los pocos datos y además clases muy desbalanceadas. Por ello se aplica transfer learning Xception y aumento de datos en el entrenamiento.	1º Entrenamiento: 74,96% 2º Entrenamiento: 80,17% Sobreaprendizaje
4.7.10	2 etiquetas	La finalidad es comprobar el rendimiento del modelo en la clasificación binaria de radiografías con neumonía de Covid-19 típica versus el resto de las categorías. Las clases están más balanceadas en cuanto a número de imágenes, pero el compendio de patologías y no patologías comprendidas en la clase negativa de neumonía de Covid-19 es muy diversa, por ello hemos de utilizar los mejores modelos y con aumento de datos en entrenamiento. Se utiliza inicialmente el modelo de transfer learning Xception.	1º Entrenamiento: 67,69% 2º Entrenamiento: 55,45% Sobreaprendizaje
4.7.11	2 etiquetas	En la búsqueda de un mejor rendimiento en la clasificación	1º Entrenamiento:

		binaria de neumonía de Covid-19 típica versus sin ella. Se utiliza el mejor modelo del artículo médico Inception V3, con aumento de datos en el entrenamiento	88,79% SESGO
4.7.12	Conclusión: En los 12 modelos anteriores no podemos considerar ningún resultado como óptimo para el problema que nos ocupa. Aunque se han utilizado diferentes recursos para la mejora del rendimiento de los modelos, tenemos un problema de pocos datos y además otro problema de datos desbalanceados. Por esta razón se ha buscado otras soluciones. Es por esta razón por la que se pasa a utilizar un conjunto de datos externo, como posible mejora del rendimiento.		

El detalle de todo el código se recoge en el punto 6 del Notebook de trabajo (50).

A continuación, se detallan los subapartados descritos en la tabla superior con un mayor detalle de los diferentes recursos utilizados en cada uno de los modelos. Cada subapartado consta de: la finalidad/justificación descrita, tabla descriptiva del modelo, métrica de evaluación y conclusión

De forma generalizada se utiliza en todos los modelos los siguientes parámetros:

- Función de pérdidas=" categorical_crossentropy"
- Métrica del modelo: "accuracy".
- El entrenamiento se realiza con:
 - batch_size=16
 - img_size=299

4.7.1. Modelo con red neuronal convolucional de 9 capas con aumento de datos

Finalidad: comprobar el rendimiento obtenido por una red neuronal convolucional creada con 9 capas. Este será el punto de partida del estudio.

N.º notebk.	Datos de Entrada	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
6.1	<p>Se utilizan los datos de la competición.</p> <p>La entrada es carpeta de imágenes train del directorio: ./Fuentes/train y data frame con columna de etiqueta de salida.</p> <p>Total, imágenes: 5068 para train 1266 para valid</p> <p>Con Data Augmentation</p>	<p>Se crea una CNN utilizando padding tipo samme con 8 capas + la capa de salida:</p> <ul style="list-style-type: none"> - 1 capa con 64 filtros de dimensión 7 y max pool de dimensión 2. El input_shape= [299,299,3]. - 1 capa con convolución de 128 filtros de dimensión 3 y max pool de dimensión 2. - 1 capa con convolución de 256 filtros de dimensión 3 y max pool de dimensión 2. - 1 capa tipo flatten. - 1 capa tipo dense de dimensión 128 y función de activación relu. - 1 capa Dropout a 0.5. - 1 capa tipo dense de dimensión 64 y función de activación relu. - 1 capa Dropout a 0.5. - La capa de salida: capa tipo dense de dimensión 4 y función de activación sigmoidea: "softmax". 	<ul style="list-style-type: none"> - optimizer= nadam - epochs=10. 	<p>4 clases de salida:</p> <ul style="list-style-type: none"> - atypical. - Indeterminate. - negative - typical 	45,97% de accuracy en datos de validación.	<p>Tengo Sobreaprendizaje.</p> <p>En la matriz de confusión puedo ver que clasifica todo con el valor de la clase mayoritaria que es la neumonía de Covid-19 – typical.</p> <p>Se queda atascado el entrenamiento con el accuracy de val en 45,97% no tiene sentido seguir entrenando.</p>

Métrica de evaluación:

```
| Test Accuracy: 0.4597156398104265  
Confusion matrix: tf.Tensor(  
[[ 0  0  0 89]  
 [ 0  0  0 239]  
 [ 0  0  0 356]  
 [ 0  0  0 582]], shape=(4, 4), dtype=int32)
```

Figura 41. Código Notebook TFM. Métrica del modelo 6.1 del Notebook.

Conclusión:

Tenemos sobreaprendizaje, clasifica todo con el valor de la clase mayoritaria que es la neumonía de Covid-19 – typical. Esta red neuronal convolucional no es apropiada para el conjunto de datos que tenemos ya que son pocos y además están desbalanceados por lo que para poder obtener un mejor rendimiento hemos de utilizar una red neuronal convolucional con mayor rendimiento y más robusta. Es por ello, por lo que se proceda a la utilización de un **transfer learning**.

4.7.2. Modelo con transfer learning – cargamos Xception, con aumento de datos

Finalidad: Mejora del rendimiento del modelo anterior con la utilización de transfer learning Xception.

El modelo previamente entrenado elegido es la arquitectura Xception. La arquitectura Xception se detalla en el punto 2.2.3. de la Memoria del trabajo Fin de Máster. Esta red ha sido entrenada con más de un millón de imágenes de la base de datos ImageNet (63) y es capaz de clasificar en 1.000 categorías. Por lo que la red ha aprendido representaciones ricas en características para una amplia gama de imágenes. **Cabe decir que el dominio del problema a resolver “patologías en radiografías de tórax” no es el mismo para el cual ha sido entrenada esta red, aun así, se ha comprobado que el resultado es positivo y además no se ha encontrado otra red neuronal convolucional preentrenada para la clasificación objeto de este trabajo.**

N.º notebk.	Datos de Entrada	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
6.2 y 6.3	<p>Se utilizan los datos de la competición.</p> <p>La entrada es carpeta de imágenes train del directorio:./Fuentes/train y data frame con columna de etiqueta de salida.</p> <p>Total, imágenes: 5068 para train 1266 para valid</p> <p>Con Data Augmentation</p>	<p>Se utiliza transfer learning, se importa el modelo Xception.</p> <ul style="list-style-type: none"> - Utilizamos los pesos de la propia red: weights="imagenet" - No se incluye la capa de salida, ya que he de cambiarla por mi capa de salida: include_top=False. - Se Añade al modelo base una capa Global Average Pooling. - Se añade la capa de salida, con una capa dense de dimensión 4 y función de activación sigmoidea: "softmax". 	<p>1º Entrenamiento</p> <ul style="list-style-type: none"> - optimizer= SGD(lr=0.2, momentum=0.9, decay=0.001) - epochs=5. - base_model.trainable:False. <p>2º Entrenamiento. Ajuste Fino cambio:</p> <ul style="list-style-type: none"> - base_model.trainable:True - Se importa de Keras, modulo keras.callbacks.EarlyStopping . Me permite detener el entrenamiento si la función de pérdidas de validación aumenta con un patience=10. - optimizer=Adam(lr=0,001). - epochs=100 	<p>4 clases de salida:</p> <ul style="list-style-type: none"> - atypical. - Indeterminate - negative - typical 	<p>1º Entren.:</p> <p>51,74% de accuracy en datos de validación.</p> <p>2º Entren.:</p> <p>59,32% de accuracy en datos de validación.</p>	<p>Se observan indicios de Sobreaprendizaje en los epochs 12 y 22.</p> <p>En la matriz de confusión se puede ver que las dos clases con mayor peso:</p> <ul style="list-style-type: none"> - negative. - typical. <p>Son las que clasifica mejor, sin embargo, las otras dos tienen malos resultados.</p>

Métrica de evaluación:

```
Test Accuracy: 0.5932069510268563
Confusion matrix: tf.Tensor(
[[ 21   5  14  49]
 [ 38   8  64 129]
 [ 32  20 242  62]
 [ 36  12  54 480]], shape=(4, 4), dtype=int32)
```

Figura 42. Código Notebook TFM. Métrica del modelo 6.3 del Notebook.

Conclusión:

Tenemos sobreaprendizaje con este modelo, tal como se puede ver en la matriz de confusión clasifica mejor las dos clases con mayor peso: negative y typical, sin embargo, las otras dos tienen malos resultados. Se ha de buscar una mejora del rendimiento y una posible solución es comprobar si el rendimiento del modelo mejora si se modifican los datos de entrada de este.

4.7.3. Modelo con transfer learning Xception con datos de entrada y etiqueta de salida convertidos a un vector, con aumento de datos

Finalidad: Tal como se ha detallado en el apartado “4.5 Preprocesamiento de los datos” de la Memoria del Trabajo, los modelos de las redes neuronales convolucionales pueden tener un mejor rendimiento cuando la entrada de los datos es un vector. Este modelo tiene como finalidad comprobar si esto mejora el resultado hasta ahora obtenido.

Además, se modifican las transformaciones del aumento de los datos, incluyendo: width_shift_range=0.1, height_shift_range=0.1, vertical_flip=True, y quitando shear_range = 0.1, brightness_range = [0.8, 1.1]. De esta forma se verifica si las transformaciones realizadas hasta el momento son las correctas.

N.º notebk.	Datos de Entrada	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
6.4	<p>Se utilizan los datos de la competición.</p> <p>La entrada de los datos es un vector tanto para las imágenes como para la etiqueta de salida. (One-hot-encoder)</p> <p>Total, imágenes: 5068 para train 1266 para valid</p> <p>Con Data Augmentation</p>	<p>Se utiliza transfer learning se importa el modelo Xception.</p> <p>como modelo base:</p> <ul style="list-style-type: none"> - Igual que el modelo 6.3 y 6.4. - El único cambio en este modelo antes de la capa de salida se incluye: - Una capa Batch Normalization con función de activación “relu”. - Una capa flatten antes de la capa de salida dense con dimensión 4. 	<p>1º Entrenamiento</p> <ul style="list-style-type: none"> - optimizer= Nadam(lr=0.001) - epochs=10. - base_model.trainable:False. <p>2º Entrenamiento, Ajuste Fino cambio:</p> <ul style="list-style-type: none"> - base_model.trainable:True - Se importa de Keras, modulo keras.callbacks.EarlyStopping - patience=10. - optimizer=Nadam(lr=0,001). - epochs=100 	<p>4 clases de salida:</p> <ul style="list-style-type: none"> - atypical. - Indeterminate - negative - typical 	<p>1º Entren.:</p> <p>57,93% de accuracy en datos de validación.</p> <p>2º Entren.:</p> <p>63,30% de accuracy en datos de validación.</p>	<p>El primer entrenamiento se comporta bien sin sobreaprendizaje</p> <p>Sin embargo, en el segundo entrenamiento tenemos más Sobreaprendizaje Con bajadas muy pronunciadas en los epochs 3, 5, 9, 12, 14, 18 y 24. Por lo que tengo más sobreaprendizaje que en el modelo 6.3</p>

Métrica de evaluación:

```
80/80 [=====] - 12s 150ms/step - loss: 0.9532 - accuracy: 0.6330  
[0.9532450437545776, 0.6329913139343262]
```

Figura 43. Código Notebook TFM. Métrica del modelo 6.4 del Notebook.

Conclusión:

En este caso en el primer entrenamiento se comporta mejor, pero **en el segundo entrenamiento** tiene bajadas constantes en el rendimiento del modelo en los datos de validación, por lo que tengo **mayor sobreaprendizaje**. Hay bajadas muy pronunciadas en los epochs 3, 5, 9, 12, 14, 18 y 24.

En este modelo también se puede ver que el optimizer hasta ahora utilizado que mejor me funciona en el primer entrenamiento es Nadam, con un learning rate de 0,001., Además la conversión de los datos a un vector es positiva en el primer entrenamiento, pero no ofrece grandes mejoras en el segundo entrenamiento ya tenemos rápidamente indicios de sobreaprendizaje en el tercer epoch.

4.7.4. Modelo con transfer learning Xception con datos de entrada y etiqueta de salida convertidos a un vector, SIN aumento de datos

Finalidad: Se pretende comprobar si el aumento de los datos nos está ayudando o no en la mejora del rendimiento del modelo, por ello se realiza de nuevo modelo tomando como base el punto 6.4 del notebook de trabajo, pero sin data augmentation.

N.º notebk.	Datos de Entrada	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
6.5	IGUAL 6.4 SIN Data Augmentation	IGUAL 6.4	IGUAL 6.4	IGUAL 6.4	1º Entren.: 53,91% de accuracy en datos de validación. 2º Entren.: 59,34% de accuracy en datos de validación.	Además de reducir el accuracy. En este caso tengo un Sobreaprendizaje muy pronunciado.

Métrica de evaluación:

```
80/80 [=====] - 12s 147ms/step - loss: 1.9779 - accuracy: 0.5935
[1.9778562784194946, 0.5935280323028564]
```

Figura 44. Código Notebook TFM. Métrica del modelo 6.5 del Notebook.

Conclusión:

Se concluye como era lógico que el aumento de los datos realizado nos ayuda a mejorar un poco el resultado.

4.7.5. Modelo con transfer learning Xception (DESDE CERO) con datos de entrada y etiqueta de salida convertidos a un vector, con aumento de datos

Finalidad: Se pretende comprobar si se mejora el resultado del modelo cuando tras cargar el modelo Xception, se entrena la red desde el principio. Se aprovecha la arquitectura robusta de este modelo y se entrena desde el inicio todas las capas del modelo. La intención es ver si obtiene un mejor rendimiento del modelo sin cargar los pesos preentrenados del mismo en el modelo base.

Además, se establecen cambios en las transformaciones del aumento de los datos realizados en el modelo 6.4, se elimina: `vertical_flip=True` y se añade: `brightness_range= [0.9, 1.1]`.

N.º notebk.	Datos de Entrada	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
6.6	Igual 6.4.	Igual 6.4. -	1º Entrenamiento - optimizer= Nadam(lr=0.1) Empiezo con un learning rate más elevado. - epochs=Aumento 25 . - base_model.trainable: True 2º Entrenamiento, Ajuste Fino cambio: Igual que el modelo 6.4.	Igual 6.4.	1º Entren.: 47,91% de accuracy en datos de validación. 2º Entren.: 47,91% de accuracy en datos de validación.	Se queda atascado el accuracy de validación en 47,91% desde el 2º epoch en el primer entrenamiento. En el segundo entrenamiento no avanza de 47,91%

Métrica de evaluación:

```
80/80 [=====] - 72s 897ms/step - loss: 1.2099 - accuracy: 0.4791
[1.209938645362854, 0.47908446192741394]
```

Figura 45. Código Notebook TFM. Métrica del modelo 6.6 del Notebook.

Conclusión: Como se puede ver el resultado es peor en este modelo que cuando se cargan los pesos del modelo Xception preentrenados.

4.7.6. Modelo con transfer learning – cargamos VGG16, con aumento de datos

Finalidad: Tomando como referencia el artículo médico detallado en el punto 2.3 de la Memoria de este trabajo “Clasificación automatizada de anomalías de las radiografías de tórax con el uso de redes convolucionales profundas.” (28) y en base a la figura “Figura 21. Rendimiento de diferentes arquitecturas de CNN con diferentes tamaños de imagen de entrada en el conjunto de datos “ChestX-ray 14””, se decide utilizar dos modelos que tiene buen rendimiento en la clasificación de anomalías en las radiografías de tórax. Se elige el modelo VGG16 con menor coste computacional y el modelo Inception V3 que es el modelo con mayor rendimiento. (Las barras de error de la figura se recuerda representan las desviaciones estándar de los valores medios). Este último modelo se utilizará en el siguiente apartado.

El siguiente modelo VGG16, se toma también como referencia el código utilizado en otro trabajo de “Detección y clasificación de células de la sangre periférica usando aprendizaje profundo” (64)

N.º notebk.	Datos de Entrada	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
6.7	<p>Se utilizan los datos de la competición.</p> <p>La entrada es carpeta de imágenes train del directorio: ./Fuentes/train y data frame con columna de etiqueta de salida.</p> <p>Total, imágenes: 5068 para train 1266 para valid</p> <p>Con Data Augmentation</p>	<p>Se utiliza transfer learning y se importa el modelo VGG16 como modelo base:</p> <ul style="list-style-type: none"> - Se Utilizan los pesos de la propia red: weights="imagenet" - No se incluye la capa de salida, ya que he de cambiarla por mi capa de salida: include_top=False. - Se añade al modelo base una capa Flatten y una capa dense previa con dimensión 256 y activación: "relu". - Finalmente, la capa de salida, con una capa dense de dimensión 4 y función de activación sigmoidea: "softmax" 	<p>1º Entrenamiento</p> <ul style="list-style-type: none"> - optimizer= Adam (learning_rate=1e-4) - epochs=5. - base_model.trainable=False. <p>2º Entrenamiento.</p> <ul style="list-style-type: none"> - optimizar= se prueban varios Adam, Nadam, RMSprop - base_model.trainable=True - Se importa de Keras, modulo keras.callbacks.EarlyStop ping - patience=10. - epochs=100 	<p>4 clases de salida:</p> <ul style="list-style-type: none"> - atypical. - Indeterminate - negative - typical 	<p>1º Entren.:</p> <p>18,88% de accuracy en datos de validación.</p> <p>2º Entren.:</p> <p>45,97% de accuracy en datos de validación.</p>	<p>Se queda atascado en 45,97% de accuracy en datos de validación. Clasifica la clase mayoritaria.</p>

Métrica de evaluación: En este caso se toma como referencia los datos del propio entrenamiento del modelo. Se puede ver en el resultado del último epoch que el accuracy de los datos de validación es de 45,97%

Conclusión: Como se ha podido comprobar el modelo VGG16 no es suficiente para aportar un buen rendimiento para nuestro conjunto de datos, ya que el accuracy alcanzado en los datos de validación es tan sólo de un 45,97%.

4.7.7. Modelo con transfer learning – cargamos Inception V3, con aumento de datos

Finalidad: Tal como se ha expuesto en la finalidad descrita en el apartado 4.7.6, en este caso pasamos a utilizar el modelo Inception V3, que es el que tiene un mejor rendimiento en el artículo médico (28). Se pretende comprobar si es de utilidad para la mejora del rendimiento de nuestro resultado hasta ahora obtenido.

N.º <small>notebk.</small>	Datos de Entrada	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
6.8	<p>Se utiliza los datos de la competición.</p> <p>La entrada de los datos es un vector tanto para las imágenes como para la etiqueta de salida. (One-hot-encoder)</p> <p>Total, imágenes: 5068 para train 1266 para valid</p> <p>Con Data Augmentation</p>	<p>Se utiliza transfer learning y se importa el modelo InceptionV3.</p> <ul style="list-style-type: none"> - Se utilizan los pesos de la propia red: weights="imagenet" - No se incluye la capa de salida, ya que he de cambiarla por mi capa de salida: include_top=False - Antes de la capa de salida se incluye: - Una capa Batch Normalization con función de activación "relu". - Una capa flatten antes de la capa de salida dense con dimensión 4. 	<p>1º Entrenamiento</p> <ul style="list-style-type: none"> - optimizer= Nadam(lr=0.001) - epochs=10. - base_model.trainable:False. <p>2º Entrenamiento, Ajuste Fino cambio:</p> <ul style="list-style-type: none"> - base_model.trainable:True - Importo de Keras, modulo keras.callbacks.EarlyStopping - patience=5, lo bajo. -optimizer=Nadam(lr=0,001). - epochs=100 	<p>4 clases de salida:</p> <ul style="list-style-type: none"> - atypical. - Indeterminate - negative - typical 	<p>1º Entren.:</p> <p>26,20% de accuracy en datos de validación.</p> <p>2º Entren.:</p> <p>26,20% de accuracy en datos de validación.</p>	<p>El primer entrenamiento se atasca en 26,20% y en el segundo tiene subidas y bajada en el rango 47,91% y 26,20%</p>

Métrica de evaluación: En este caso se toma como referencia los datos del propio entrenamiento del modelo. En el segundo entrenamiento tiene subidas y bajada en el rango 47,91% y 26,20%, finalmente concluye con 26,20%

Conclusión:

Es el peor de los resultados obtenidos.

Estos modelos pueden tener un buen rendimiento para clasificaciones binarias, o con conjunto de datos más elevados o incluso con conjuntos de datos más moderados, pero con clases bien balanceadas, pero para nuestro conjunto de datos el modelo que nos ofrece el mejor resultado es el modelo Xception. Y con el modelo Xception para 4 clases de salida, el mejor rendimiento y con indicios de sobreaprendizaje, se obtiene en el modelo 6.2 +6.3 de sólo un 59,32% de accuracy en datos de validación.

Dado que tenemos pocos datos y además las categorías están muy desbalanceados hemos de buscar otras opciones que nos ofrezcan mejores resultados. Por lo que en los próximos modelos se realizan modelos con una clasificación reduciendo el número de etiquetas de salidas.

4.7.8. Modelo con transfer learning – cargamos Xception, con aumento de datos – 3 etiquetas de salida

Finalidad: La primera opción que se estudia para la mejora del rendimiento es la creación de un modelo con una clasificación con tres clases de salida. Esta clasificación se expone en el punto “4.7 Cargar los datos (archivos .csv, carpetas archivos de imágenes) análisis exploratorio y preparación de los datos” de la Memoria de este trabajo Fin de Máster, y se deriva de la conclusión tras la lectura la metodología de clasificación utilizada en las anotaciones SIIM. Se Utiliza la clasificación: Negativo (sin opacidad), Neumonía típica (opacidad bilateral – neumonía Covid-19) y Otros (con los casos de Atípicos e indeterminados que tienen alguna opacidad). Para la creación del modelo se utiliza transfer learning del Modelo Xception, ya que es el mejor resultado hasta ahora obtenido.

N.º notebk.	Datos de Entrada	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
6.9	<p>Se utilizan los datos de la competición.</p> <p>La entrada es carpeta de imágenes train del directorio: ./Fuentes/train y data frame con columna de etiqueta de salida.</p> <p>Total, imágenes: 5068 para train 1266 para valid</p> <p>Con Data Augmentation</p>	<p>Se utiliza transfer learning y se importa el modelo Xception.</p> <ul style="list-style-type: none"> - Se Utilizan los pesos de la propia red: weights="imagenet" - No se incluye la capa de salida, ya que he de cambiarla por mi capa de salida: include_top=False. - Se Añade al modelo base una capa Global Average Pooling y otra Batch Normalization. - Se añade la capa de salida, con una capa dense de dimensión 3 y función de activación sigmoidea: "softmax". 	<p>1º Entrenamiento</p> <ul style="list-style-type: none"> - optimizer= Nadam(lr=0,001) - epochs=10. - base_model.trainable:False. <p>2º Entrenamiento, Ajuste Fino cambio:</p> <ul style="list-style-type: none"> - base_model.trainable:True - Se importa de Keras, modulo keras.callbacks.EarlyStopping. Me permite detener el entrenamiento si la función de pérdidas de validación aumenta con un patience=10. - optimizer=Nadam(lr=0.001) - epochs=100 	<p>3 clases de salida:</p> <ul style="list-style-type: none"> - Yes (typical) - None (Negative) - otros (atypical e indeterminate) 	<p>1º Entren.:</p> <p>54,90% de accuracy en datos de validación.</p> <p>2º Entren.:</p> <p>62,95% de accuracy en datos de validación.</p>	<p>En este caso al observar la matriz de confusión se clasifica con una tasa de aciertos del 78% la clase Negative, con un 86% la típica (Neumonía de Covid-19) pero el problema es con la tercera clase que tan sólo tiene una tasa de acierto del 6% (Otros). En este caso son pacientes con otras patologías. El modelo los clasifica en un 60% como typical y en un 34% sin patologías cuando sí las tienen.</p>

Métrica de evaluación:

```
Test Accuracy: 0.6295418641390206  
Confusion matrix: tf.Tensor(  
[[279  65  12]  
 [ 74 499   9]  
 [111 198  19]], shape=(3, 3), dtype=int32)
```

Figura 46. Código Notebook TFM. Métrica del modelo 6.9 del Notebook.

Conclusión:

Tal como se suponía en el momento de realizar esta agrupación, el modelo aprende a generalizar en el entrenamiento en las dos categorías mayoritarias, con tasas de aciertos del 78% para la clase Negative, con un 86% para la clase típica (Neumonía de Covid-19) pero el problema lo encontramos con la tercera clase que tan sólo tiene una tasa de acierto del 6% (Otros). Como se anticipaba en el momento de realizar esta clasificación contamos con pocos datos y además muy desbalanceados en esta última categoría Recordamos que son radiografías con una amplia diversidad de patologías. El modelo los clasifica en un 60% como typical y en un 34% sin patologías cuando sí las tienen.

Además, las bajadas en el rendimiento del modelo del 2º entrenamiento, en los epochs 4 y 6 en los datos de validación nos inducen a pensar que podemos tener indicios de sobreaprendizaje. Esto nos lleva a la conclusión de que seguimos teniendo pocos datos y las clases están muy desbalanceadas por ello a partir de este punto nuestro objetivo ha de ser menos ambicioso. Hemos de realizar modelos con una clasificación binaria.

4.7.9. Modelo con transfer learning – cargamos Xception, con aumento de datos – 2 etiquetas de salida: Con anomalía versus sin anomalía

Finalidad: Se busca un mayor rendimiento en un modelo de clasificación tomando como base al artículo médico. (28). Se considera que el modelo puede obtener una mejor generalización en el entrenamiento cuando se utiliza una clasificación binaria de con opacidad y sin opacidad. Se busca un modelo que detecte o no cualquier tipo de anomalía en una radiografía de tórax. En este caso se parte con un problema de datos muy desbalanceados: Sin opacidad 1.736 imágenes de radiografías de tórax (27%) frente a 4.598 radiografías de tórax con opacidad (73%). (Tras corregir el error del punto 2.2 etiqueta "image_label", utilizando punto 2.3 de notebook "label_opacity").

N.º notebk.	Datos de Entrada	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
6.10	<p>Se utilizan los datos de la competición.</p> <p>La entrada es carpeta de imágenes train del directorio: ./Fuentes/train y data frame con columna de etiqueta de salida.</p> <p>Total, imágenes: 5068 para train 1266 para valid</p> <p>Con Data Augmentation</p>	<p>Se utiliza transfer learning y se importa el modelo Xception.</p> <ul style="list-style-type: none"> - Se utilizan los pesos de la propia red: weights="imagenet" - No se incluye la capa de salida, ya que he de cambiarla por mi capa de salida: include_top=False. - Se añade al modelo base una capa Global Average Pooling y otra Batch Normalization. - Se añade la capa de salida, con una capa dense de dimensión 2 y función de activación sigmoidea: "softmax". 	<p>1º Entrenamiento</p> <ul style="list-style-type: none"> - optimizer= Nadam(lr=0,001) - epochs=10. - base_model.trainable:False. <p>2º Entrenamiento. Ajuste Fino cambio:</p> <ul style="list-style-type: none"> - base_model.trainable:True - Se importa de Keras, modulo keras.callbacks.EarlyStopping. Nos permite detener el entrenamiento si la función de pérdidas de validación aumenta con un patience=10. - optimizer=Nadam(lr=0.001) - epochs=100 	<p>2 clases de salida:</p> <ul style="list-style-type: none"> - Yes (typical, atypical e indeterminate) - None (Negative) 	<p>1º Entren.:</p> <p>74,96% de accuracy en datos de validación.</p> <p>2º Entren.:</p> <p>80,17% de accuracy en datos de validación.</p>	<p>Finalmente tenemos sobreaprendizaje en el segundo entrenamiento con subidas y bajadas del accuracy de validación en el rango de [54,19%, a 81,04%]</p> <p>Nos quedaríamos con el accuracy del primer entrenamiento de un 74,96%</p>

Métrica de evaluación:

```
Test Accuracy: 0.80173775671406  
Confusion matrix: tf.Tensor(  
[[226 130]  
 [121 789]], shape=(2, 2), dtype=int32)
```

Figura 47. Código Notebook TFM. Métrica del modelo 6.10 del Notebook.

Conclusión: Las bajadas pronunciadas de rendimiento en el 2º entrenamiento con los datos de validación en los epochs 4, 8, 11y 15 nos inducen a pensar que tenemos indicios de sobreaprendizaje. Nuestro mejor resultado hasta ahora se obtendría en este caso con el primer entrenamiento con un 74,96% de accuracy en datos de validación. Aun así, no se puede considerar un resultado óptimo para la solución de este problema. Pasamos a comprobar cuál es el rendimiento del modelo con clasificación binaria de neumonía de Covid-19 típica y sin ella.

4.7.10. Modelo con transfer learning – cargamos Xception, con aumento de datos – 2 etiquetas de salida: Con neumonía de Covid-19 típica versus sin ella.

Finalidad: Finalmente se comprueba cuál es el rendimiento de un modelo de clasificación binaria para las clases con neumonía de Covid-19 (typical) versus sin neumonía de Covid-19 (resto de las clases). Punto 2.4 del Notebook del trabajo. Aunque de partida parece que los datos están más balanceados: Con neumonía de Covid-19 (typical)=3.007 imágenes de radiografías de tórax y Sin neumonía de Covid-19 típica (atypical, indeterminate y negative) =3.327 imágenes de radiografías de tórax. Hay una gran diversidad dentro de la categoría sin neumonía de Covid-19 típica, llegando a juntar diversas patologías y casos sin patologías.

N.º notebk.	Datos de Entrada	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
6.11	<p>Se utilizan los datos de la competición.</p> <p>La entrada es carpeta de imágenes train del directorio: ./Fuentes/train y data frame con columna de etiqueta de salida.</p> <p>Total, imágenes: 5068 para train 1266 para valid</p> <p>Con Data Augmentation</p>	<p>Se utiliza transfer learning y se importa el modelo Xception.</p> <ul style="list-style-type: none"> - Se utilizan los pesos de la propia red: weights="imagenet" - No se incluye la capa de salida, ya que he de cambiarla por mi capa de salida: include_top=False. - Se añade al modelo base una capa Global Average Pooling. - Se añade la capa de salida, con una capa dense de dimensión 2 y función de activación sigmoidea: "softmax". 	<p>1º Entrenamiento</p> <ul style="list-style-type: none"> - optimizer= SGD(lr=0,2) - epochs=10. - base_model.trainable:False. <p>2º Entrenamiento, Ajuste Fino cambio:</p> <ul style="list-style-type: none"> - base_model.trainable:True - Se importa de Keras, modulo keras.callbacks.EarlyStopping. Nos permite detener el entrenamiento si la función de pérdidas de validación aumenta con un patience=5 - optimizer=Nadam(lr=0.001) - epochs=100 	<p>2 clases de salida:</p> <ul style="list-style-type: none"> - Yes (typical) - None (Negative, atypical e indeterminate) 	<p>1º Entren.:</p> <p>67,69% de accuracy en datos de validación.</p> <p>2º Entren.:</p> <p>55,45% de accuracy en datos de validación.</p>	<p>1º entrenamiento clasifica sobre todo la categoría mayoritaria sin neumonía de covid-19 típica con una tasa de acierto del 83% y tan sólo un 49% de tasa de acierto de los casos con neumonía de Covid-19 típica.</p> <p>2º entrenamiento, a pesar de reducir patience a 5, hay sobreaprendizaje, lo clasifica prácticamente todo como SIN neumonía de Covid-19 - típica</p>

Métrica de evaluación:

1º entrenamiento: Matriz de confusión

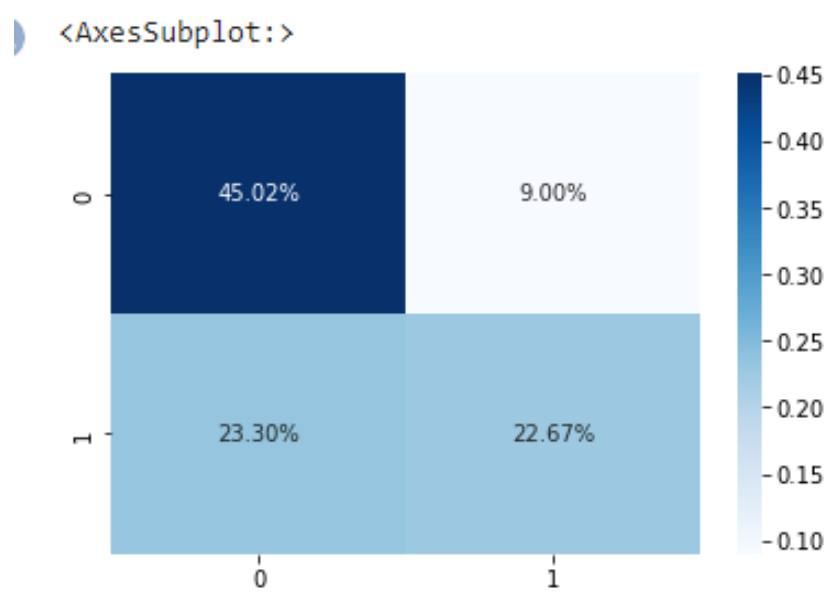


Figura 48. Código Notebook TFM. Métrica del modelo 6.11 del Notebook- primer entrenamiento.

2º entrenamiento: Matriz de confusión

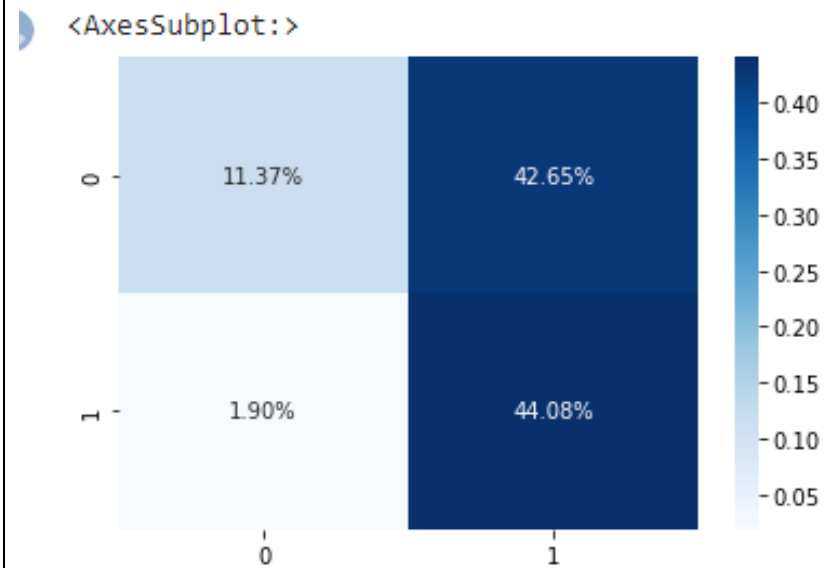


Figura 49. Código Notebook TFM. Métrica del modelo 6.11 del Notebook- segundo entrenamiento.

Conclusión: Tal como se ha descrito en el apartado observaciones de la tabla de este modelo el primer entrenamiento clasifica mejor la categoría mayoritaria con el 53% de las imágenes de radiografías de tórax, SIN neumonía Covid-19 -típica, con una tasa de acierto del 83% y tan sólo un 49% de tasa de acierto en los casos CON neumonía de Covid-19-típica. **Mientras que, en el segundo entrenamiento, a pesar de reducir patience a 5, hay sobreaprendizaje,** lo clasifica prácticamente todo como Sin neumonía de Covid-19 -típica.

4.7.11. Modelo con transfer learning – cargamos InceptionV3, con aumento de datos – 2 etiquetas de salida: Con neumonía de Covid-19 típica versus sin ella.

Finalidad: En este último modelo se pretende comprobar si el rendimiento mejora al utilizar transfer learning del modelo que ofrecía el mejor rendimiento en el artículo médico (28), "Figura 21. Rendimiento de diferentes arquitecturas de CNN con diferentes tamaños de imagen de entrada en el conjunto de datos "ChestX-ray 14". El modelo utilizado es InceptionV3. Se modifica también la entrada de los datos y etiqueta de salida como un vector y también se cambia la función de pérdidas a "sparse_categorical_crossentropy".

N.º notebk.	Datos de Entrada	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
6.12	<p>Se utilizan los datos de la competición.</p> <p>La entrada de los datos es un vector tanto para las imágenes como para la etiqueta de salida. (One-hot-encoder)</p> <p>Total, imágenes: 5068 para train 1266 para valid</p> <p>Con Data Augmentation</p>	<p>Se utiliza transfer learning y se importa el modelo InceptionV3 como modelo base:</p> <ul style="list-style-type: none"> - Se utiliza weights="imagenet" y include_top=False. - Se añade al modelo base una capa Global Average Pooling, otra Batch Normalization, una capa Flatten con función de activación relu. - Se añade la capa de salida, con una capa dense de dimensión 2 y función de activación sigmoidea: "softmax". 	<p>1º Entrenamiento</p> <ul style="list-style-type: none"> - optimizer= "RMSprop" - epochs=10. - base_model.trainable:False. 	<p>2 clases de salida:</p> <ul style="list-style-type: none"> - Yes (typical) - None (Negative, atypical e indeterminate) 	<p>1º Entren.:</p> <p>88,79% de accuracy en datos de validación.</p>	<p>En este caso tengo un problema de <u>SESGO</u></p>

Métrica de evaluación:

Evolución del accuracy en el modelo

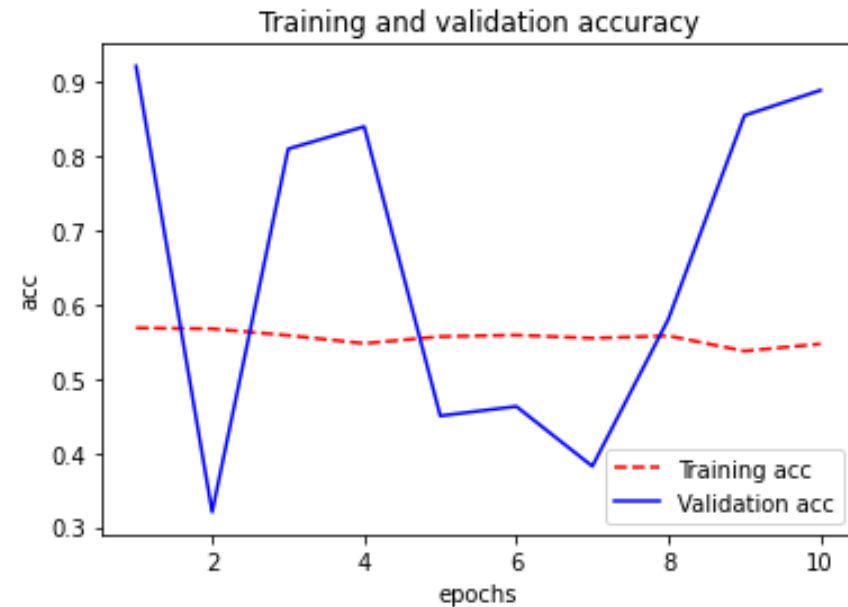


Figura 50. Código Notebook TFM. Métrica del modelo 6.12 del Notebook

Evolución de la función de pérdidas



Figura 51. Código Notebook TFM. Métrica del modelo 6.12 del Notebook

Conclusión: En este modelo tenemos un problema de sesgo, tal como se puede apreciar en las gráficas de las métricas. Se producen resultados erróneos de forma sistemática, en este caso tenemos saltos continuados en los rangos de valores del 92% de accuracy en validación a 32%, del 83% al 38%..., mientras que en los datos de entrenamiento se mantiene en rangos de (54-56%). El problema de sesgo se puede deber a que tenemos un modelo demasiado simple para el problema a resolver.

4.7.12. Conclusión tras la ejecución de los 12 modelos anteriores

Se han realizado diferentes modelos, en la búsqueda continuada de una mejora en el rendimiento del modelo de clasificación para un conjunto de imágenes de radiografías de tórax con diversas patologías y sin ellas. Un conjunto de datos reducido en número y con categorías muy desbalanceadas.

En la búsqueda de una menor diferencia en la varianza (subapartado “Errores a resolver en modelos de aprendizaje profundo” del apartado 2.1.2 de la Memoria del trabajo fin de máster) se han aplicado ayudas como:

- Data Augmentation
- Transfer learning con la utilización del Modelo Xception.
- Hemos añadido capas ocultas dentro de nuestra red neuronal convolucional: Batch normalization y Dropout cuyo objetivo es la regularización y con ello reducir la varianza del sistema.
- También se ha utilizado Early Stopping junto con ajuste fino para solucionar el mismo problema.

El mejor resultado se ha obtenido con una clasificación binaria de con opacidad versus sin opacidad, con la utilización de transfer learning del modelo Xception (No se encontró ninguna red entrenada con radiografías y al utilizar esta red entrenada con imagenet ayudaba ligeramente en los resultados) y con aumento de datos en el entrenamiento, pero solo se alcanza una precisión del 74% en el conjunto de validación en un primer entrenamiento (subapartado 4.7.9). Este resultado no se puede considerar óptimo como solución al problema planteado. Si se pretende mejorar el resultado hemos de aumentar nuestro conjunto de datos de estudio **Por estas razones, se concluye que la mejora en el rendimiento se podrá obtener si se utiliza un conjunto de datos externo.** Solo se podrá mejorar el resultado del rendimiento con el uso de un conjunto de datos externo, si el nuevo conjunto de datos de entrenamiento es representativo de nuestro conjunto de datos de producción (conjunto de datos de la competición inicial).

4.8 USO CONJUNTO DE DATOS EXTERNO

En este apartado se detallan cuales han de ser los aspectos relevantes que debemos cumplir cuando se realiza el uso de un conjunto de datos externo. Aspectos que debemos garantizar con el fin de que el nuevo conjunto de datos que obtengamos tras la unión de nuestro conjunto de datos de producción (Datos de la competición inicial) más el nuevo conjunto de datos externo que se utilice sea representativo y por lo tanto nos pueda servir de ayuda en la obtención una de mejora del rendimiento del modelo de clasificación a obtener.

En el apartado se detallan cuáles serán los nuevos totales de imágenes de radiografías de tórax que tendremos en cada una de las particiones de datos necesarias para la obtención del modelo.

Se expone la forma y librerías utilizadas para la carga de datos, agrupación de categorías de salida y partición final de directorios de imágenes con el fin de poder ocultar un 20% de las imágenes del conjunto de datos inicial que nos servirán para la validación final del modelo obtenido.

Los aspectos relevantes que en el uso de un conjunto de datos externo debemos cumplir son:

- Que sean datos del mismo dominio: En nuestro caso tomamos el conjunto de datos "Covid-19 Radiography Database" descrito en el punto 4.1 Conjunto de datos de la Memoria del Trabajo fin de Máster. En principio se trata al igual que nuestro conjunto de datos de partida de radiografías de tórax, con diversa clasificación, encontrándose entre la clasificación radiografías de tórax con neumonía de Covid-19. Con ello aumentamos hasta 27.999 imágenes de radiografías de tórax.
- Ver si existen diferencias entre los datos, a veces inapreciables:
 - Formatos de las imágenes y tamaños de estas: En este caso lo resolvemos ya que tanto el nuevo conjunto de datos como los datos iniciales están en formato png con dimensión 299x299 pixeles.
 - Calidades de las radiografías: Este es un problema que en el caso de las radiografías de tórax puede pasar cuando las radiografías no son tomadas por el mismo hospital, e incluso por la misma máquina. Puede haber diferencias en las mismas que hagan que ambos conjuntos de datos no se deban unir para la resolución de un problema.

Para poder conocer si nuestro nuevo conjunto de datos de entrenamiento (tras la unión de datos) es representativo de nuestro conjunto de datos inicial, deberemos realizar la siguiente partición de datos:

- Un conjunto de validación final y testeo que debe estar formado únicamente por datos de producción, en nuestro caso son los de la competición inicial. (Reservaremos un 20% de las imágenes de la carpeta de train). Total= 1266 imágenes

- Un conjunto de datos de entrenamiento que estará formado por el nuevo conjunto de los datos más parte de los datos de producción. En nuestro caso será el 80% restante de imágenes de la carpeta train + todo el conjunto de imágenes del conjunto de datos externo. Total= 26.232 imágenes
- A partir de este nuevo conjunto de datos de entrenamiento, a su vez debemos realizar partición de datos entre entrenamiento y validación. (80%-20%) para nuestro caso. Total= 20.986 imágenes train y 5.246 imágenes test.

El objetivo por alcanzar en esta unión de datos es que la varianza que es la diferencias/dispersión entre los errores alcanzados entre los nuevos datos de entrenamiento y los nuevos datos de validación ha de ser lo más similar posible al error que obtenemos al aplicar el nuevo modelo entrenado en los datos de test (solo datos de producción) que reservamos para la validación final. Explicación expuesta en el subapartado "Recursos externos" del punto 2.1.2 de la Memoria del Trabajo Fin de Máster.

Tal como se ha expuesto en el punto "4.3 Cargar los datos (ficheros csv, carpetas ficheros de imágenes) análisis exploratorio y preparación de los datos" en el subapartado "Unión de bases de datos de los 4 conjuntos de datos presentados en el punto 4.1 Conjunto de datos de la Memoria del Trabajo Fin de Máster" se ha procedido a la carga de datos atendiendo a las pautas de cómo se ha de realizar una unión de un conjunto de datos de producción más un conjunto de datos externo.

Primero unimos todos los datos de los conjuntos de datos:

Punto 7 del Notebook de trabajo (50): Finalmente se carga data frame con todos los datos y se crean las etiquetas de salida binarias:

- Con opacidad (con anomalía) / Sin opacidad (sin anomalía)
- Con neumonía de Covid-19 (typical + Covid) /Sin neumonía de Covid-19 (Resto de categorías)

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 27499 entries, 0 to 1344
Data columns (total 4 columns):
#   Column                Non-Null Count  Dtype
---  -
0   id                     27499 non-null  object
1   study_label            27499 non-null  object
2   2l_opacity_no_opacity  27499 non-null  object
3   2l_covid_no_covid      27499 non-null  object
dtypes: object(4)
memory usage: 2.1+ MB
```

Figura 52. Código Notebook TFM. df_new es el data frame resultante tras la unión de todos los conjuntos de datos

Segundo se procede a realizar la partición de datos (1) conjunto validación final /test (sólo conjunto de datos competición) (2) Nuevo conjunto de datos de entrenamiento, que a su vez dividiremos en 80% train y 20% validación (con la unión de datos competición más conjunto externo):

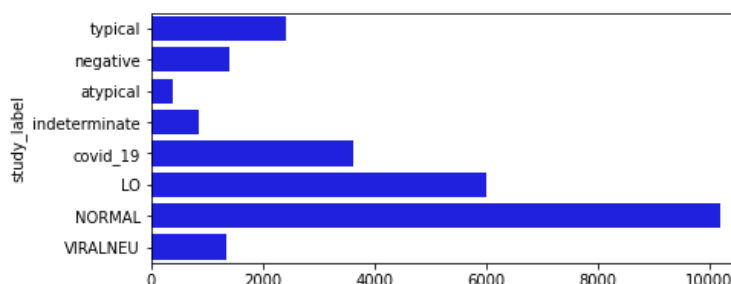
Punto 8 del Notebook de trabajo (50). Se utilizan las librerías:

- "OS": Creación de directorios y listar archivos de los directorios creados.
- "Shutil": Me permite mover 80% de las imágenes del conjunto de producción a la nueva carpeta de entrenamiento y reservar 20% de las imágenes para el test final (conjunto oculto para saber si los datos train de la unión son representativos).
- "GLOB": puedo copiar las carpetas de imágenes en el nuevo directorio train.
- "Pandas": Puedo crear data frame con la lista de archivos de los directorios creados.
- "MERGE": Puedo unir a los "id" de las imágenes las columnas con las etiquetas de salida creadas.

Nuevo conjunto de datos train:

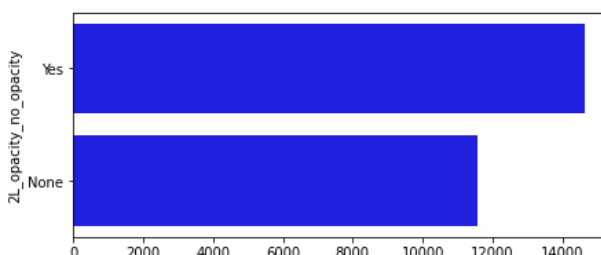
Total imágenes por tipo de patología -columna "study label"

	study_label	Total clases
0	LO	6012
1	NORMAL	10192
2	VIRALNEU	1345
3	atypical	386
4	covid_19	3616
5	indeterminate	866
6	negative	1394
7	typical	2421



Total imágenes para la clasificación binaria con opacidad/con anomalía versus sin opacidad /sin anomalía.

	2L_opacity_no_opacity	Total clases
0	None	11586
1	Yes	14646



Total imágenes para la clasificación binaria con neumonía de Covid-19 versus sin sin neumonía de Covid-19.

	2L_covid_no_covid	Total clases
0	None	20195
1	Yes	6037

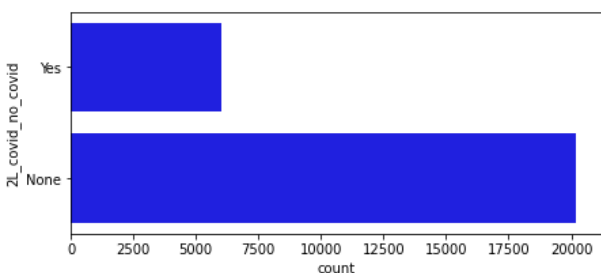


Figura 53. Código Notebook TFM. Recuento y Visualización etiquetas de salida nuevo train tras la unión bases de

datos (sin el conjunto de datos oculto para test final).

La conclusión que se toma tras ver la partición de los datos es la de utilizar como etiqueta de salida en la búsqueda de un modelo que mejore el rendimiento, la clasificación con opacidad versus sin opacidad que tiene los datos más balanceados (56% vs 44% respectivamente). Para la clasificación sin neumonía de Covid-19 versus con ella, los datos están muy desbalanceados (77% versus 23%) y como se ha podido comprobar con nuestros datos el balanceo de las clases es sumamente importante en la realización de esta clasificación

Conjunto de datos para Test final oculto (sólo datos de producción/competición inicial):

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1267 entries, 0 to 1266
Data columns (total 4 columns):
#   Column                Non-Null Count  Dtype
---  -
0   id                    1267 non-null   object
1   study_label           1267 non-null   object
2   2L_opacity_no_opacity 1267 non-null   object
3   2L_covid_no_covid     1267 non-null   object
dtypes: object(4)
memory usage: 49.5+ KB
```

study_label Total clases

0	atypical	97
1	indeterminate	242
2	negative	342
3	typical	586

2L_opacity_no_opacity Total clases

0	None	342
1	Yes	925

Figura 54. Código Notebook TFM. Recuento conjunto de datos oculto para test Final.

Si se compara el conjunto test final con el conjunto inicial de la competición podemos ver que los porcentajes por categoría seleccionados son muy similares, obteniendo una muestra lo más similar a nuestro conjunto de datos de partida.

Conjunto inicial Competición

	study_label	Total_clases
0	atypical	483
1	indeterminate	1.108
2	negative	1.736
3	typical	3.007

Conjunto test final

	study_label	Total_clases	
0	atypical	97	17%
1	indeterminate	242	18%
2	negative	342	16%
3	typical	586	16%

Figura 55. Total, clases seleccionadas "conjunto final test" Porcentaje sobre conjunto inicial.

4.9 Modelos con la unión de los conjuntos de datos: Modelo Xception con Data Augmentation – Clasificación binaria (con opacidad/con anomalía y sin opacidad/sin anomalía).

En este apartado se realizan dos modelos con el objetivo de conseguir mejorar el rendimiento del modelo de clasificación binaria. Se recoge en el punto 9 del Notebook de trabajo (50).

PRIMER MODELO

Finalidad: Se pretende obtener un mejor resultado en el rendimiento de un modelo de clasificación binaria con opacidad versus sin opacidad tras aumentar el conjunto de datos iniciales con la unión de conjunto de datos externo. Se ha conseguido pasar de un conjunto de entrenamiento de 6.334 imágenes de radiografías de tórax a 26.232 imágenes de radiografías de tórax. Además, las categorías de salida en este caso están más balanceadas 56% con opacidad versus 44% sin opacidad. Se aplican las mejores prácticas hasta ahora utilizadas con los mejores rendimientos, se detallan en la siguiente tabla:

N.º notebk.	Datos de Entrada	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
9	<p>Se utiliza la unión de los conjuntos de datos: Total, 26.232 imágenes.</p> <p>La entrada es carpeta de imágenes: "train_nuevo_final" del directorio: ".\Fuentes_C" y data frame: "df_new_train_final" con columna de etiqueta de salida.</p> <p>Total, imágenes: 20.986 para train 5.246 para valid</p> <p>Con Data Augmentation</p>	<p>Se utiliza transfer learning y se importa el modelo Xception. como modelo base:</p> <ul style="list-style-type: none"> - -Se Utiliza los pesos de la propia red: weights="imagenet" - - No se incluye la capa de salida, ya que he de cambiarla por mi capa de salida: include_top=False. - Se añade al modelo base una capa Global Average Pooling y una capa Batch Normalization. - Se añade la capa de salida, con una capa dense de dimensión 2 y función de activación sigmoidea: "softmax". 	<p><u>1º Entrenamiento</u></p> <ul style="list-style-type: none"> - optimizer= Nadam(lr=0,001) - epochs=20 - base_model.trainable:False. 	<p>2 clases de salida:</p> <ul style="list-style-type: none"> - Yes (con opacidad) - None (sin opacidad) 	<p><u>1º Entren.:</u></p> <p>76,67% de accuracy en datos de validación.</p>	<p>En este caso no se produce un proceso de sobreaprendizaje porque no se incrementa la tasa de error según avanzamos.</p> <p>Podemos ver las tasas de acierto de casos negativos y positivos en la siguiente matriz de confusión</p>

Métricas de evaluación:

```
Test Accuracy: 0.7666793747617232
Matrix confusion: tf.Tensor(
[[ 965  429]
 [ 795 3057]], shape=(2, 2), dtype=int32)
```

Matriz Confusión - 1º Entrenamiento**Valid Accuracy: 76,67%**

0 - None	965	429	69%
1 - Yes	795	3057	79%

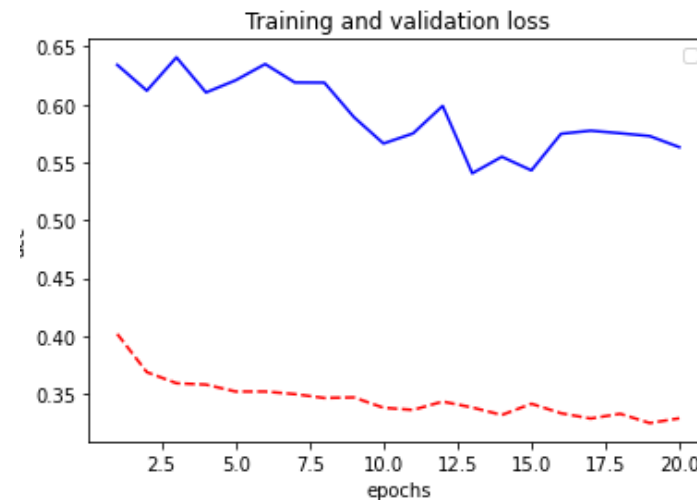
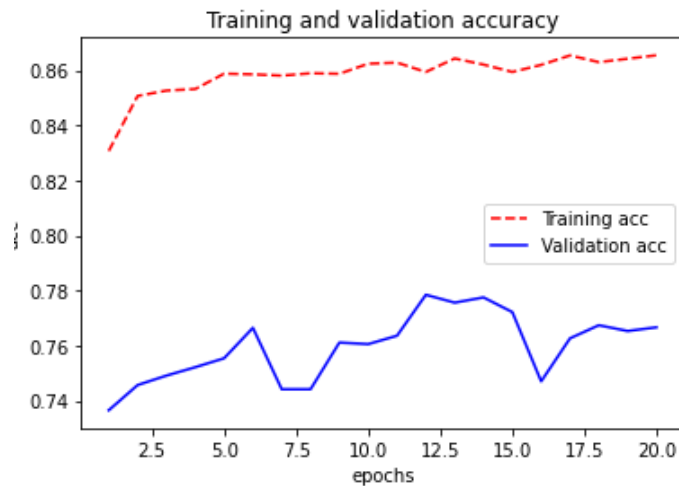
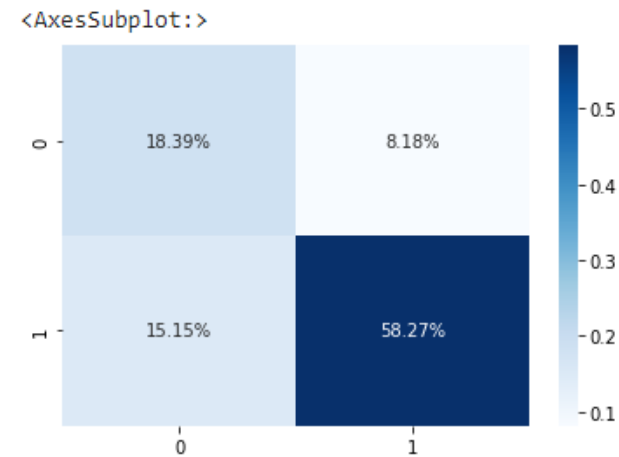
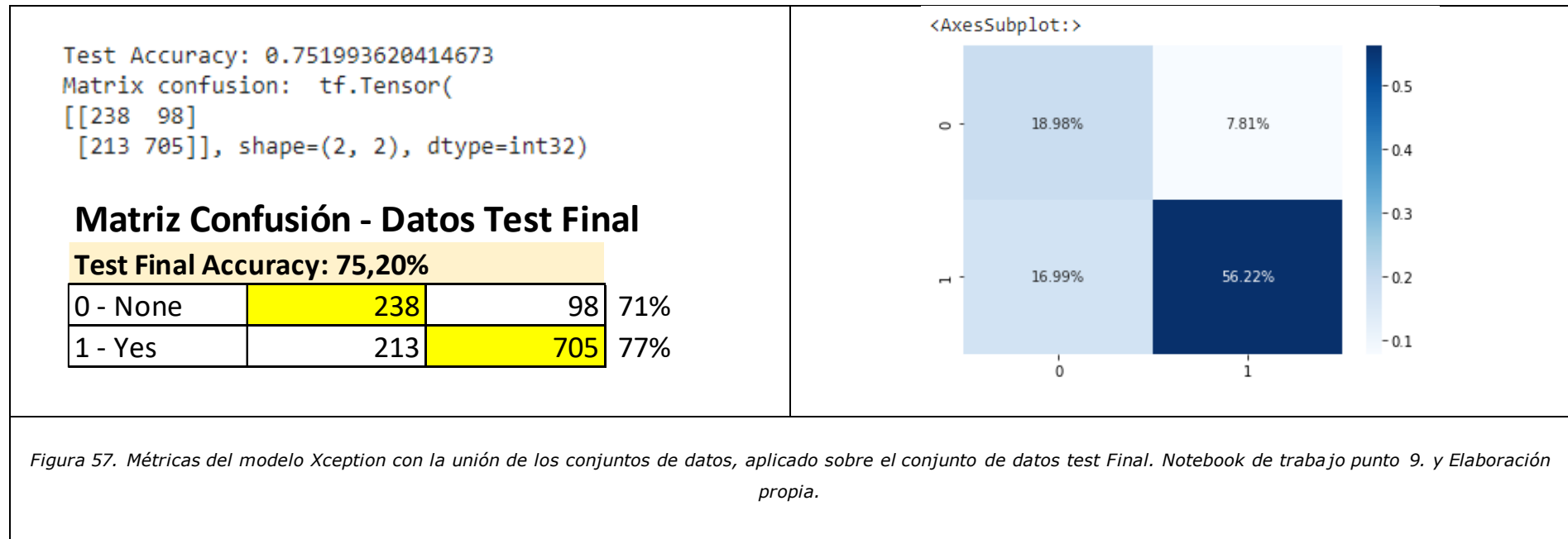


Figura 56. Matriz de confusión del modelo Xception con la unión de los conjuntos de datos – 1º Entrenamiento. Notebook de trabajo punto 9. y Elaboración propia.

Conclusión: Existe una mejora en el accuracy del conjunto de validación. Se alcanza un accuracy en datos de validación de 76,67%. Tal como se puede ver la **tasa de acierto de casos sin opacidad (sin anomalía) es de un 69%**, mientras que la **tasa de acierto de con opacidad (con anomalía) es de un 79%**.

Pero lo que debemos conocer es si podemos considerar que este nuevo conjunto de datos de entrenamiento es representativo del conjunto de datos inicial de la competición. Para poder conocer esto debemos de aplicar el modelo obtenido en el conjunto de Test final. Se presenta a continuación los resultados.

Ahora para saber si el conjunto de datos es representativo debemos de aplicar el modelo sobre el conjunto de datos oculto, Conjunto de datos test (solo del conjunto de datos de producción /Datos de la competición):



Vemos que, aunque baja un poco el resultado (de 76,67% a 75,20%), el comportamiento es muy similar por lo que podemos decir que **“sí es representativo”**, y con este primer entrenamiento **ya conseguimos mejorar los resultados hasta ahora obtenidos. Tenemos una 71% de tasa de acierto en los casos de sin opacidad y un 77% de acierto en los casos con opacidad y hemos pasado de un 74% de tasa de acierto en el conjunto de datos de validación a un 75,20%.**

SEGUNDO MODELO

Finalidad: Con este nuevo modelo se pretende mejorar el resultado anterior, realizando Ajuste Fino. Para ello se realiza un segundo entrenamiento. Entre los diferentes cambios que se aplican se encuentra “layer.trainable=True”

N.º notebk.	Datos de Entrada	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
9.1	<p>Utilizo unión de los conjuntos de datos: Total, 26.232 imágenes.</p> <p>La entrada es carpeta de imágenes: “train_nuevo_final” del directorio: “./Fuentes_C” y data frame: “df_new_train_final” con columna de etiqueta de salida.</p> <p>Total, imágenes: 20.986 para train 5.246 para valid</p> <p>Con Data Augmentation</p>	<p>Utilizo transfer learning e importo el modelo Xception como modelo base:</p> <ul style="list-style-type: none"> - Utilizo los pesos de la propia red: weights=“imagenet” - No incluyo la capa de salida, ya que he de cambiarla por mi capa de salida: include_top=False. - Añado al modelo base una capa Global Average Pooling y una capa Batch Normalization. - Añado la capa de salida, con una capa dense de dimensión 2 y función de activación sigmoidea: “softmax”. 	<p>2º Entrenamiento, Ajuste Fino cambio:</p> <ul style="list-style-type: none"> - base_model.trainable=True - Importo de Keras, modulo keras.callbacks.EarlyStopping . Me permite detener el entrenamiento si la función de pérdidas de validación aumenta con un patience=3(lo reduzco) - epochs=50(lo reduzco) 	<p>2 clases de salida:</p> <ul style="list-style-type: none"> - Yes (con opacidad) - None (sin opacidad) 	<p>1º Entren.:</p> <p>74,17% de accuracy en datos de validación.</p>	<p>En este caso SI tengo sobreaprendizaje.</p> <p>En este caso me clasifica prácticamente todo como la clase de con opacidad.</p>

Métricas de evaluación:

Test Accuracy: 0.7417079679756005
 Matrix confusion: tf.Tensor(
 [[62 1332]
 [23 3829]], shape=(2, 2), dtype=int32)

Matriz Confusión - 2º entrenamiento

Valid Accuracy: 74,17%

0 - None	62	1332	4%
1 - Yes	23	3829	99%

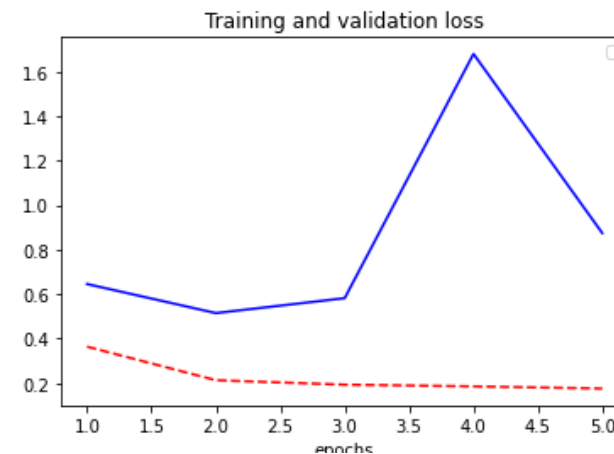
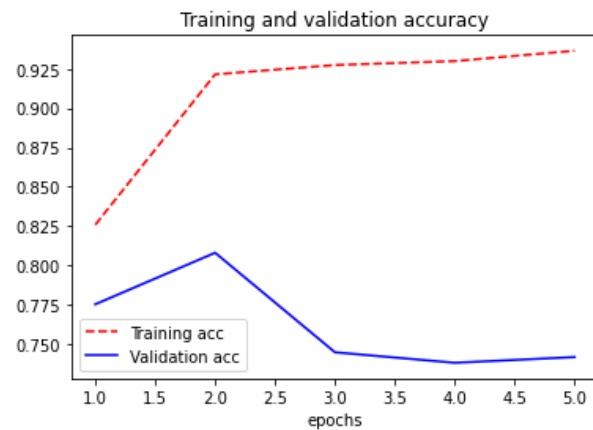
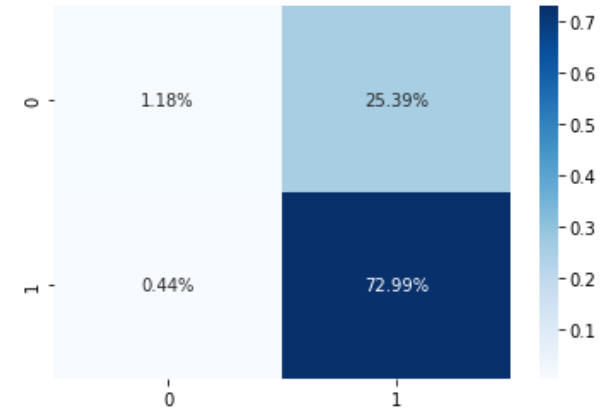
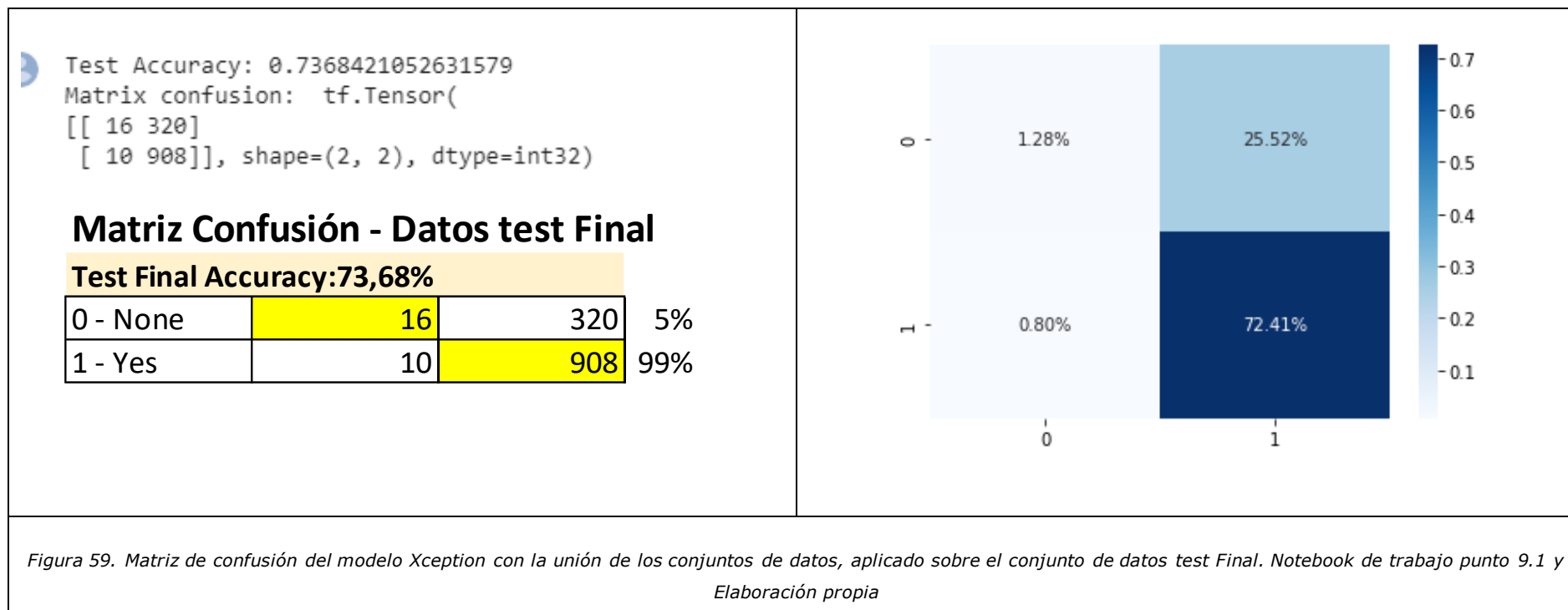


Figura 58. Métricas del modelo Xception con la unión de los conjuntos de datos – 2º Entrenamiento. Notebook de trabajo punto 9.1 y Elaboración propia

Conclusión: Como se puede ver en la matriz de confusión y en las propias gráficas **tenemos sobreaprendizaje**, ya que **baja el accuracy en validación y nos clasifica prácticamente todo como la clase mayoritaria que es con opacidad**.

El mismo comportamiento sucede al aplicar este segundo modelo en el conjunto oculto de los datos de test (conjunto de datos de producción/ datos de la competición inicial):



Finalmente, dado que en el segundo modelo tenemos sobreaprendizaje, **nos quedaríamos como mejor modelo que es el obtenido con el primer entrenamiento: con un 76,67% de accuracy en validación y un accuracy en conjunto de datos test oculto del 75,20% en validación.**

4.10 Mejora de Resultados. Modelo con modelo preentrenado aplicado a solo datos de la competición inicial

Finalidad: Atendiendo a las conclusiones del artículo médico (28), se conoce que “Se puede mejorar el rendimiento de un modelo de clasificación con un conjunto de datos moderado/reducido si se utiliza un modelo previamente preentrenado para el mismo objetivo” En base a esto, se pretende mejorar el resultado, tomando como base el modelo obtenido en el primer entrenamiento de la unión del conjunto de datos. Este será nuestro modelo preentrenado. Se aplica las mejores técnicas utilizadas como es el aumento de datos y **solo se utiliza el conjunto de datos de la competición inicial:**

- El conjunto de datos de entrenamiento: 5.067 imágenes de las radiografías de tórax (Son las mismas imágenes que se han usado en la unión del nuevo conjunto de datos de entrenamiento del modelo preentrenado).
- El conjunto de datos de test: 1.267 imágenes de las radiografías de tórax (Conjunto de datos oculto para la validación final de Test)

N.º notebk.	Datos de Entrada	Modelo	Parámetros de interés	Clasificación	Rendimiento	Observaciones
10	<p>Se utilizan los datos solo de la competición 6.334</p> <p>La entrada para train es carpeta de imágenes: “train_nuevo_ini” del directorio: “./Fuentes_C” y data frame: “df_new_train_final_ini” con columna de etiqueta de salida.</p> <p>Los datos de test se utiliza la carpeta de imágenes: “test_nuevo_fin” del directorio: “./Fuentes_C” y data frame: “df_new_test_final” con columna de etiqueta de salida.</p> <p>5.067 para train 1.267 para Test Final Con Data Augmentation</p>	<p>Utilizo Modelo preentrenado.</p> <p>Cargo los pesos del modelo obtenido en el primer entrenamiento de la unión del conjunto de datos de la competición más el conjunto de datos externo.</p>	<p>Entrenamiento</p> <p>- optimizer= Nadam(lr=0,01)</p> <p>- epochs=15</p> <p>- base_model.trainable: True.</p>	<p>2 clases de salida:</p> <p>- Yes (con opacidad)</p> <p>- None (sin opacidad)</p>	<p>77,83% de accuracy en datos de validación.</p>	<p>En este caso no tengo sobreaprendizaje.</p> <p>Mejoramos los resultados en más de dos puntos porcentual.</p>

Métrica de evaluación:

```
[ ] model_pretrained_2.evaluate(valid_generator_3)

79/79 [=====] - 71s 893ms/step - loss: 0.4651 - accuracy: 0.7783
[0.46506696939468384, 0.7783094048500061]
```

Figura 60. Accuracy test Final. Notebook de trabajo Punto 10.

Conclusión:

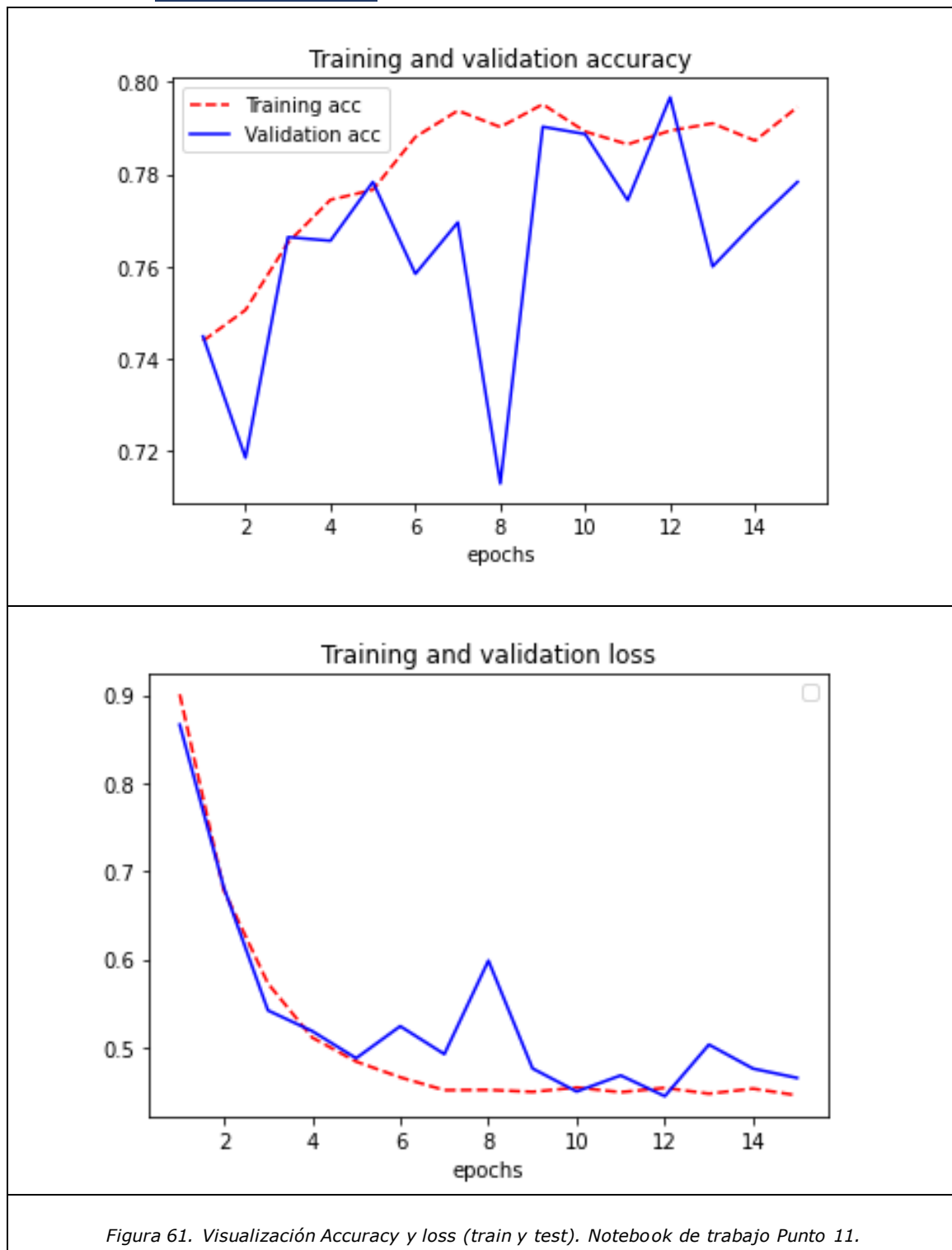
Se consigue mejorar el rendimiento del modelo de clasificación binaria (con opacidad/con anomalía versus sin opacidad/sin anomalía) con los datos de la competición inicial.

En este caso, **pasamos del 75,20% de accuracy en los datos de test** obtenido con el primer modelo entrenado con la unión de las bases de datos, **a un 77,83% de accuracy en los datos de test (más de dos puntos porcentuales).**

4.11 Visualización de los Mejores Resultados

En este apartado se detallan las diferentes métricas de evaluación para validar el modelo con los mejores resultados. Se recoge en el punto 11 Notebook de Trabajo (50).

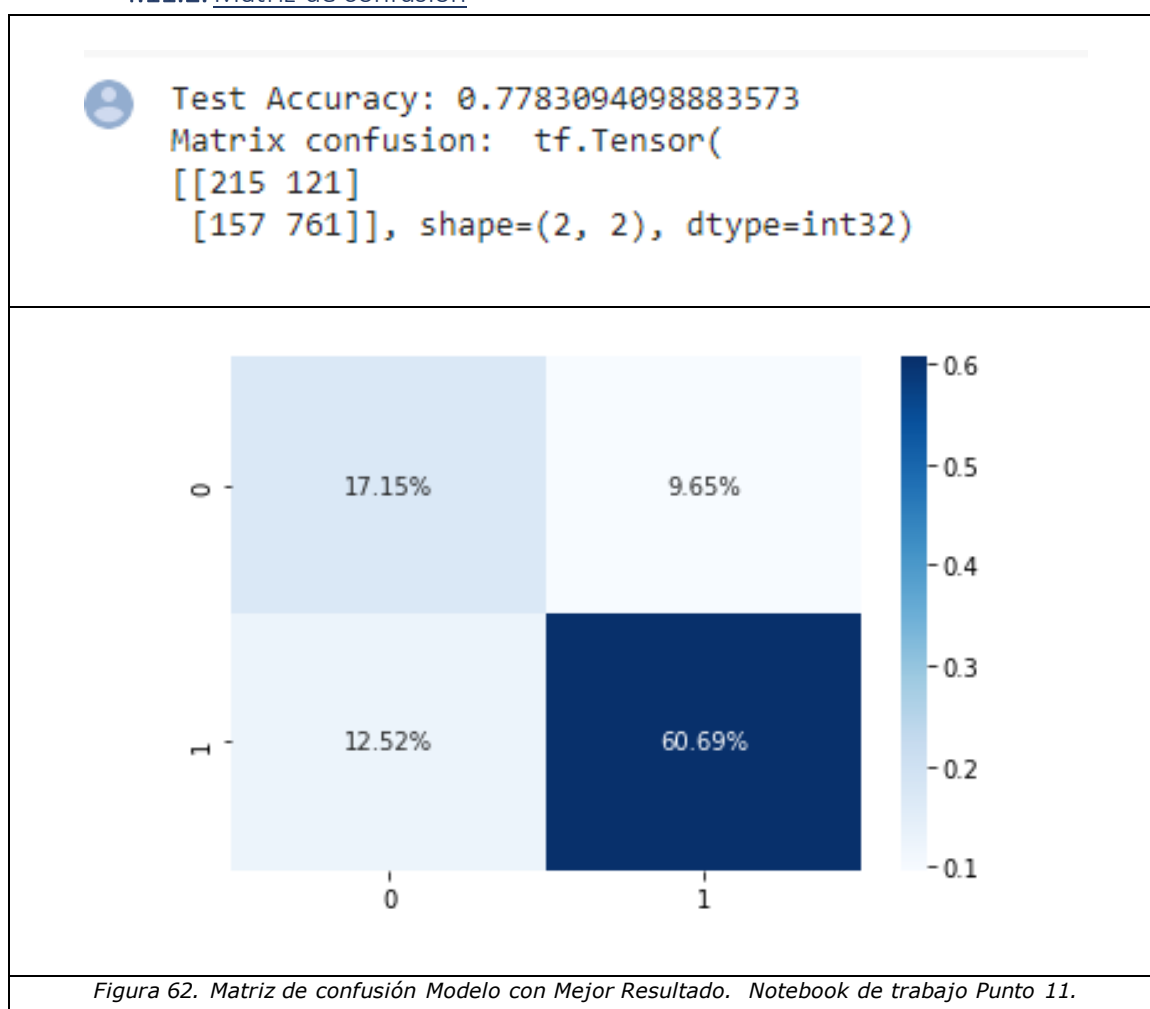
4.11.1. Gráficas con la evolución de las métricas de “accuracy” y “función de pérdidas” tanto en el entrenamiento como en la validación final del subconjunto de test.



En la figura 61, se puede observar si valoramos la métrica de accuracy, que tanto los resultados en train como en test, mantienen una pendiente ascendente, con una bajada más pronunciada en los datos de test en el epoch 8. Si vemos la escala vertical son rangos entre [0,72 y 0,80], por lo que no es una bajada muy pronunciada, y al ser puntual no podemos considerarlo como un problema de sobreaprendizaje.

Este mismo comportamiento nos lo encontramos si valoramos la gráfica de evolución de la función de pérdidas. En este caso tanto en train como en test mantienen la misma pendiente descendente. Con la subida puntual del epoch 8.

4.11.2. Matriz de confusión



En la figura 62, podemos observar, que el 77,83% de accuracy en los datos de test se consiguen con un 60,69% de precisión de verdaderos positivos y un 17,15% de precisión de verdaderos negativos. La diagonal principal mantiene valores más elevados que la diagonal de los casos no acertados. Es importante conocer como evolución la tasa de aciertos de casos con

opacidad y la de los casos sin opacidad. Lo veremos en el próximo apartado.

4.11.3. Evolución tasa de acierto en los datos de validación Test Final

Modelo resultante: 1º Entrenamiento unión bases de datos

Matriz Confusión - Datos test Final

Test Final Accuracy:75,20%

0 - None	238	98	71%
1 - Yes	213	705	77%

Modelo resultante:Tras cargar modelo preentrenado y entrenar de nuevo

Matriz Confusión - Datos test Final

Test Final Accuracy:77,83%

0 - None	215	121	64%
1 - Yes	157	761	83%

Figura 63. Comparación datos matrices de confusión final conjunto de datos Test Final – Fuente propia

En la figura 63, se observa que el aumento de accuracy del modelo en más de dos puntos porcentuales de 75,20% a 77,83%, nos ha permitido mejorar la tasa de acierto de verdaderos positivos (sensibilidad).

Conseguimos aumentar en 6 puntos porcentuales la tasa de acierto de radiografías con opacidad/con anomalía. Pasamos de un 77% de tasa de acierto a un 83% de tasa de acierto (aumento sensibilidad).

Tal como se ha podido comprobar mejoramos los resultados en una precisión final de 77,83% y una tasa de acierto en los casos positivos con anomalía en las radiografías de tórax del 83% (sensibilidad). Aun así, este resultado no se puede considerar como un resultado de éxito desde el punto de vista médico.

5. CONCLUSIONES Y LINEAS FUTURAS

En este apartado se realiza un breve resumen del estudio realizado en la Memoria del trabajo Fin de Máster, se recogen las principales conclusiones y se definen cuales sería las líneas futuras para poder obtener mejores resultados.

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

Uno los proyectos que persiguen este fin es la competición que lanza la plataforma Kaggle, con apoyo de la organizadora líder "la Sociedad de Informática por Imágenes en Medicina (SIIM)" y otras asociaciones, de la cual soy participe.

En este desafío de creación de un modelo de clasificación de patologías, se ofrece un conjunto de datos compuesto por 6.334 radiografías de tórax con patologías muy diversas que atienden a la siguiente clasificación: negativo, típico, atípico e indeterminado.

En el análisis EDA de los datos, ya se puede observar que **son pocos datos**: 6.334 imágenes de radiografías de tórax y **con categorías muy desbalanceadas**: 8% atípica, 17% indeterminada, 27% negativa y 47% típica). Además, la categoría con menor número de imágenes "atípica" es la que comprende un elevado número de patologías (Neumotórax, derrame pleural, edema pulmonar, consolidación lobar, nódulo o masa pulmonar solitaria, nódulos diminutos difusos, cavidad...).

Para poder subsanar el problema detectado de pocos datos y además con categorías desbalanceadas y siempre con la finalidad marcada de obtener el mejor resultado en el rendimiento del modelo final, se aplica:

- Preprocesamiento de las imágenes, data augmentation, capas ocultas de Batch Normalization y Dropout en el modelo, Early Stopping y Transfer learning (Xception).
- **Se realizan un total de doce modelos en la búsqueda continuada de un mayor rendimiento de un modelo de clasificación.**
- Aunque se aplican los modelos de mayor rendimiento del artículo médico referenciado (28), con nuestro conjunto de datos del trabajo, se obtienen resultados con problemas de sobreaprendizaje para el modelo VGG16 y con problema de sesgo cuando se aplica el modelo de InceptionV3. El modelo Inception V3 que daba buenos resultados en el artículo médico, resulta ser un modelo demasiado simple para poder resolver nuestro problema.

Finalmente, Nuestro mejor resultado, solo con los datos de la competición se alcanza para una clasificación binaria de: con opacidad versus sin opacidad (con anomalía versus sin anomalía), aplicando transfer learning con el modelo Xception, con aumento de datos, alcanzando **un rendimiento máximo del 74% de precisión en datos de validación.**

Además, se concluye que para poder mejorar los resultados es necesario aumentar los datos, con el uso de recursos externos.

Se utiliza un conjunto de datos externo de la plataforma Kaggle compuesto por 21.165 imágenes de radiografías de tórax, con la clasificación de neumonía de Covid-19, neumonía viral, otras opacidades torácicas y normal (sin opacidad).

Nuestro **objetivo** tras la unión de los conjuntos de datos será la búsqueda de un algoritmo con una **clasificación binaria** de (con opacidad / con anomalía versus sin opacidad/sin anomalía). Finalmente, se cuenta con un total de 26.232 imágenes de radiografías de tórax. **(56% con opacidad vs 44% sin opacidad), mejorando el balanceo de las clases de salida.**

Se siguen todos los pasos necesarios en la unión de datos iniciales con el conjunto de datos externos, para poder valorar si el conjunto nuevo de datos de entrenamiento es representativo y podemos considerar el modelo resultante como solución a nuestro problema. **Finalmente se demuestra que sí es representativo.**

El resultado obtenido con la unión de los datos y la utilización de transfer learning del modelo Xception con data augmentation, es un modelo cuyo **accuracy en datos de validación** es del **76,67%** y en los **datos de test** (solo conjunto datos iniciales) del **75,20%**, demostrando así que el conjunto de datos de entrenamiento resultante de la unión es representativo de nuestro conjunto de datos iniciales.

Tomando como referencia una de las conclusiones del artículo médico referenciado, el apartado "Clasificación automatizada de anomalías de las radiografías de tórax con el uso de -Redes convolucionales Profundas", **se consigue mejores rendimientos en conjunto de datos de tamaño moderado cuando se utiliza un modelo preentrenado.**

El mejor resultado se alcanza con la utilización del mejor modelo preentrenado con la unión de los dos conjuntos de datos.

En el resultado Final, se alcanza un modelo con un accuracy en datos de test (solo datos iniciales) del 77,83%

Si nos centramos en las **radiografías con patologías, los casos con opacidad**, nuestro resultado final es una **tasa de acierto del 83%**. Se consigue **aumentar 6 puntos porcentuales**, con el último entrenamiento, tras cargar modelo preentrenado.

Líneas futuras

Este resultado no es suficientemente bueno como para considerarse como un caso de producción o un caso de éxito en la obtención de un modelo de clasificación binario de radiografías de tórax.

Es un problema sanitario de gran índole mundial que debe exigir los mejores resultados en los algoritmos de clasificación.

En el artículo médico se refleja un rendimiento superior al 98% en modelos de clasificación binaria para conjunto de datos de mayor dimensión (NIH ChestX-ray 14:112.120 imágenes).

Finalmente se concluye que para la **obtención de un modelo de clasificación con un mayor rendimiento** que el alcanzado **será necesario obtener un conjunto de datos externo mayor y con las clases bien balanceadas**. La dimensión y el balanceo de datos es clave es la buena resolución del problema.

Para poder conseguir una precisión elevada y garantizar que el nuevo conjunto de datos sea representativo de la muestra original, se ha de tener una sensibilidad especial en ver que los datos sean representativos de la cuestión a resolver. El hecho de que la fuente de datos sea única de un hospital, y específicamente de la misma máquina de rayos X, favorece los resultados del algoritmo. Esto unido al número de muestras suficientes y balanceadas.

Sin embargo, como solución de ámbito general, lo ideal será contar con radiografías de diversos hospitales y confirmar que también funciona en este espectro de datos, solo así se confirmará que el modelo ha aprendido a realizar una buena generalización en su fase de aprendizaje y podemos aplicarlo en la validación de nuevos casos futuros. Solo así se podría llegar a considerar que el modelo podría pasar a producción.

La limitación del tiempo, así como la limitación de herramientas utilizadas para la realización de este trabajo, hacen que no sea posible realizar esta mejora propuesta.

6. INDICE DE FIGURAS

FIGURA 1 - COMPONENTES DE UNA NEURONA (3).....	8
Figura 2. Diagrama de una Neurona Artificial (PE)vs Neurona Biológica (3)	9
Figura 3. Funciones de activación habituales (4).....	11
Figura 4. Interconexión entre una neurona presináptica y una neurona postsináptica (4).....	11
Figura 5. Dirección resultante de la función de error con respecto al vector de pesos (4).....	13
Figura 6. Hiperplano resultante de la evaluación de una función de error. (4).....	14
Figura 7. Generalización. Situación idealizada	17
Figura 8. Generalización. Situación real	17
Figura 9. Matriz de Confusión binaria (12)	18
Figura 10. ¿Qué tipo de error tenemos? - apuntes Deep Learning (14)	19
Figura 11. Errores tras la unión de bases de datos - apuntes Deep Learning (14) ..	20
Figura 12. Visualización de una CNN (8)	22
Figura 13. Arquitectura de una CNN (8).....	22
Figura 14. Esquema arquitectura de Xception (16).....	24
Figura 15. Correlaciones entre el espacio y la profundidad.	25
Figura 16. Arquitectura Xception (19)	25
Figura 17. Descripción general de VGG16 (16)	26
Figura 18. Métricas de rendimiento de clasificación para diferentes arquitecturas de CNN en la base de datos de NIH "ChestX-ray 14" (28)	32
Figura 19. Curva ROC modelos CNN (a) Etiquetas elegidas por la mayoría de los radiólogos como el estándar de referencia de verdad del terreno (28).....	33
Figura 20. Curva ROC modelos CNN (b) Etiquetas derivadas de informes de texto como estándar de referencia de verdad sobre el terreno (28).....	33
Figura 21. Rendimiento de diferentes arquitecturas de CNN con diferentes tamaños de imagen de entrada en el conjunto de datos "ChestX-ray 14" (28).....	34
Figura 22. Tasa de verdaderos positivos (sensibilidad) y tasa de falsos positivos (especificidad 1) de diferentes radiólogos (n. ° 1, n. ° 2, n. ° 3 y n. ° 4) frente a diferentes etiquetas de verdad del terreno (28).....	34
Figura 23. Matrices de confusión para los diferentes conjuntos de datos de NIH-RSNA entrenados (28).	35
Figura 24. Terminología sugerida para el informe estructurado en COVID-19 en tomografía computada, basada en consenso de la Sociedad Norteamericana de Radiología. Abreviaciones: OVE = opacidades con densidad en vidrio esmerilado (48).....	40
Figura 25. Ejemplos radiografías de tórax de las diferentes categorías	41
Figura 26. Imagen carpeta Fuentes creada para el almacenamiento de los datos. .	49
Figura.27. Imagen carpeta Fuentes_1 creada para el almacenamiento de los datos.	50

Figura.28. Imagen carpeta Fuentes_C creada para el almacenamiento de los datos modelo Unión BBDD.....	50
Figura 29. Código notebook TFM: lectura, conversión y visualización imagen 10 del conjunto de datos train – Fuente elaboración propia	52
Figura 30. Código notebook TFM: data frame final df_train.	53
Figura 31. Código Notebook TFM. Recuento etiqueta de salida "study_label".	54
Figura 32. Código Notebook TFM. Recuento etiqueta de salida "image_label".	55
Figura 33. Código Notebook TFM. Recuento etiqueta de salida "label_opacity".	56
Figura 34. Código Notebook TFM. Recuento etiqueta de salida "label_n_Covid_19U".	57
Figura 35. Código Notebook TFM. Recuento etiqueta de salida "label_opacity_3l".	58
Figura 36. Código Notebook TFM. Data frame unión bases de datos "df_new".	59
Figura 37. Código Notebook TFM. Recuento y Visualización "study_label" del nuevo conjunto de datos tras la unión de data frames.	60
Figura 38. Código Notebook TFM. Recuento y Visualización etiquetas de salida train de la unión bases de datos final (sin el conjunto de datos oculto para test).....	62
Figura 39. Código Notebook TFM. Visualización imágenes radiografías de tórax conjunto train con cajas que marcan las opacidades.	64
Figura 40. Código Notebook TFM. Visualización imágenes radiografías de tórax tras aplicar función de preprocesamiento.	65
Figura 41. Código Notebook TFM. Métrica del modelo 6.1 del Notebook.	74
Figura 42. Código Notebook TFM. Métrica del modelo 6.3 del Notebook.	76
Figura 43. Código Notebook TFM. Métrica del modelo 6.4 del Notebook.	78
Figura 44. Código Notebook TFM. Métrica del modelo 6.5 del Notebook.	79
Figura 45. Código Notebook TFM. Métrica del modelo 6.6 del Notebook.	80
Figura 46. Código Notebook TFM. Métrica del modelo 6.9 del Notebook.	85
Figura 47. Código Notebook TFM. Métrica del modelo 6.10 del Notebook.	87
Figura 48. Código Notebook TFM. Métrica del modelo 6.11 del Notebook- primer entrenamiento.	89
Figura 49. Código Notebook TFM. Métrica del modelo 6.11 del Notebook- segundo entrenamiento.	89
Figura 50. Código Notebook TFM. Métrica del modelo 6.12 del Notebook	91
Figura 51. Código Notebook TFM. Métrica del modelo 6.12 del Notebook	91
Figura 52. Código Notebook TFM. df_new es el data frame resultante tras la unión de todos los conjuntos de datos.....	94
Figura 53. Código Notebook TFM. Recuento y Visualización etiquetas de salida nuevo train tras la unión bases de datos (sin el conjunto de datos oculto para test final).....	95
Figura 54. Código Notebook TFM. Recuento conjunto de datos oculto para test Final.	96
Figura 55. Total, clases seleccionadas "conjunto final test" Porcentaje sobre conjunto inicial.....	96

Figura 56. Matriz de confusión del modelo Xception con la unión de los conjuntos de datos – 1º Entrenamiento. Notebook de trabajo punto 9. y Elaboración propia.	98
Figura 57. Métricas del modelo Xception con la unión de los conjuntos de datos, aplicado sobre el conjunto de datos test Final. Notebook de trabajo punto 9. y Elaboración propia.	99
Figura 58. Métricas del modelo Xception con la unión de los conjuntos de datos – 2º Entrenamiento. Notebook de trabajo punto 9.1 y Elaboración propia.	101
Figura 59. Matriz de confusión del modelo Xception con la unión de los conjuntos de datos, aplicado sobre el conjunto de datos test Final. Notebook de trabajo punto 9.1 y Elaboración propia.	102
Figura 60. Accuracy test Final. Notebook de trabajo Punto 10.	104
Figura 61. Visualización Accuracy y loss (train y test). Notebook de trabajo Punto 11.	105
Figura 62. Matriz de confusión Modelo con Mejor Resultado. Notebook de trabajo Punto 11.	106
Figura 63. Comparación datos matrices de confusión final conjunto de datos Test Final – Fuente propia.	107

7. REFERENCIAS

1. **Kaggle.** <https://www.kaggle.com/>. [En línea] julio de 2021. <https://www.kaggle.com/c/siim-covid19-detection>.
2. **García, Aranza.** <https://www.kaggle.com/>. [En línea] Agosto de 2021. <https://www.kaggle.com/aranza1972/code?scroll=true>.
3. **Xabier Basogain Olabe - Escuela Superior de Ingeniería de Bilbao, EHU.** LIBRO -XAB. [En línea] https://ocw.ehu.eus/pluginfile.php/40137/mod_resource/content/1/redes_neuro/contenidos/pdf/libro-del-curso.pdf.
4. **Aplicada, Grupo de investigación de topología Computacional y Matemática.** CONCEPTOS BÁSICOS SOBRE REDES NEURONALES. [En línea] <http://grupo.us.es/gtocom/pid/pid10/RedesNeuronales.htm>.
5. **Bertona, Luis Federico.** ENTRENAMIENTO DE REDES NEURONALES BASADO EN ALGORITMOS EVOLUTIVOS. [En línea] 2005. <http://laboratorios.fi.uba.ar/lsi/bertona-tesisingenieriainformatica.pdf>.
6. **Viñuela, Pedro Isasi y León, Inés Galván.** *Las redes neuronales artificiales y sus aplicaciones prácticas*. s.l. : Alhambra, 2004.
7. **Xin Yao - School of Computer Science, University of Binningham, Birmingham, West Midlands, UK.** *Evolving artificial neural networks*. s.l. : IEEE, 1999.
8. **Carrión, Carmelo Bonilla.** TFG - REDES CONVOLUCIONALES. [En línea] 2020. <https://idus.us.es/bitstream/handle/11441/115221/TFG%20DGM%20Bonilla%20Carri%C3%B3n%20Carmelo.pdf?sequence=1&isAllowed=y>.
9. **Gert Cauwenberghs - University of California, San Diego.** A Fast Stochastic Error-Descent Algorithm for Supervised Learning and Optimization. [En línea] 1993. https://www.researchgate.net/publication/2591884_A_Fast_Stochastic_Error-Descent_Algorithm_for_Supervised_Learning_and_Optimization.
10. **Werbos, Paul John.** [En línea] 1994. https://books.google.es/books?id=WdR3OOM2gBwC&printsec=frontcover&dq=PJ+Werbos+-+1994+-+books.google.com&hl=es&sa=X&redir_esc=y#v=onepage&q&f=false.
11. **neuronales, Descripción general de diferentes optimizadores para redes.** [En línea] <https://ichi.pro/es/descripcion-general-de-diferentes-optimizadores-para-redes-neuronales-247308806799555>.
12. **métricas, Juanbarrios.com - La matriz de confusión y sus.** <https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/>. [En línea]
13. **Python?, <https://themachinelearners.com/> - ¿Qué funciona mejor para evaluar un modelo de clasificación en.** https://themachinelearners.com/curva-roc-vs-prec-recall/#Que_es_la_curva_ROC. [En línea]
14. **Ramos, Pablo Criado.** Redes de Neuronas Artificiales. 2021, págs. 91-96.

15. **TensorFlow, Raul E. Lopez Briega - Redes neuronales convolucionales con.** [En línea] 2016. <https://relopezbriega.github.io/blog/2016/08/02/redes-neuronales-convolucionales-con-tensorflow/>.
16. **ICHI.PRO.** [En línea] <https://ichi.pro/es/arquitecturas-de-redes-neuronales-convolucionales-mas-populares-116734478665629>.
17. **profundidad, Xception: Deep Learning con convoluciones separables en.** [En línea] <https://arxiv.org/abs/1610.02357>.
18. **gitbook.io, Stephan osterburg -.** [En línea] <https://stephan-osterburg.gitbook.io/coding/coding/ml-dl/tensorfow/ch3-xception/xception-architectural-design>.
19. **profundas, Joyce Xu - DLI - Guía para arquitecturas de redes.** [En línea] 2018. <https://www.deeplearningitalia.com/guia-para-arquitecturas-de-redes-profundas/>.
20. **escala, Redes convolucionales muy profundas para el reconocimiento de imágenes a gran.** [En línea] <https://arxiv.org/abs/1409.1556>.
21. **convoluciones, Profundizando en las.** [En línea] <https://arxiv.org/abs/1409.4842>.
22. *Clasificación de GE Imagenet con redes neuronales convolucionales profundas. En Los avances en la información neuronales procesan sistemas.* **Krizhevsky, A., Sutskever, I. & Hinton.** 2012, NeurIPS, págs. 1097-1105.
23. *Repensar la arquitectura inicial de la visión por computadora.* **Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. y Wojna, Z.** s.l. : IEEE, 2016. Conferencia IEEE sobre Visión por Computador y Reconocimiento de Patrones 2818–2826.
24. *Aprendizaje residual profundo para el reconocimiento de imágenes.* **Él, K., Zhang, X., Ren, S. y Sun, J.** s.l. : IEEE, 2016. Conferencia IEEE sobre Visión por Computador y Reconocimiento de Patrones 770–778 .
25. **imágenes, Lara Moreno Díaz-Alejo - Análisis comparativo de arquitecturas de redes de neuronas para la clasificación de.** [En línea] <https://reunir.unir.net/bitstream/handle/123456789/10008/Moreno%20D%C3%ADaz-Alejo%2C%20Lara.pdf?sequence=1&isAllowed=y>.
26. *Redes convolucionales densamente conectadas.* **Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, KQ.** s.l. : IEEE, 2017. Conferencia IEEE sobre Visión por Computador y Reconocimiento de Patrones 4700–4708.
27. **Classification), Sik-Ho Tsang - DenseNet — Dense Convolutional Network (Image.** [En línea] 2018. <https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803>.
28. **networks, Automated abnormality classification of chest radiographs using deep convolutional neural.** <https://www.nature.com/npjdigitalmed/>. <https://www.nature.com/articles/s41746-020-0273-z>. [En línea] 14 de Mayo de 2020. <https://www.nature.com/articles/s41746-020-0273-z>.
29. **Collaborators, Authors List - GBD 2015 Mortality and Causes of Death.** Global, regional, and national life expectancy, all-cause mortality, and cause-specific mortality for 249 causes of death, 1980–2015: a systematic analysis for the Global Burden of Disease Study 2015. [https://www.thelancet.com/journals/lancet/article/PIIS0140-6736\(16\)31012-](https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(16)31012-)

1/fulltext. [En línea] Octubre de 2016.
<https://www.sciencedirect.com/science/article/pii/S0140673616310121>.

30. **mortality., American Lung Association. Trends in lung cancer morbidity and.** [En línea] 2014. <https://www.lung.org/assets/documents/research/lc-trend-report.pdf>.

31. **Academy, MM Waldrop - Proceedings of the National.** What are the limits of deep learning? [En línea] 2019.
https://scholar.google.com/scholar_lookup?title=News%20feature%3A%20What%20are%20the%20limits%20of%20deep%20learning%3F&journal=Proc.%20Nat%20Acad.%20Sci.&volume=116&pages=1074-1077&publication_year=2019&author=Waldrop%2CMM.

32. **Paras Lakhani, Baskaran Sundaram.** Deep Learning at Chest Radiography: Automated Classification of Pulmonary Tuberculosis by Using Convolutional Neural Networks. *RSNA - Radiology.* [En línea]
<https://pubs.rsna.org/doi/10.1148/radiol.2017162326>.

33. **Dataset, NIH- National Institutes of Health Chest X-Ray.**
<https://www.kaggle.com/nih-chest-xrays/data>. [En línea]
<https://www.kaggle.com/nih-chest-xrays/data>.

34. **at, The NIH chest radiographs that support the findings of this study are publicly available.** [En línea] <https://nihcc.app.box.com/v/ChestXray-NIHCC>.

35. **Irvin, J. et al. CheXpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In Proceedings of the AAAI Conference on Artificial Intelligence 590–597.** [En línea] 2019.
<https://aaai.org/ojs/index.php/AAAI/article/view/3834/3712>.

36. **MIMIC-CXR, Multiple - Base de datos.** [En línea] 2019.
<https://physionet.org/content/mimic-cxr/2.0.0/>.

37. **L Oakden-Rayner - Academic radiology, 2020 - Elsevier.** Exploring Large-scale Public Medical Image Datasets. [En línea]
<https://www.sciencedirect.com/science/article/abs/pii/S107663321930488X>.

38. **X-rays, External data - 1000 triple-read Covid-19.**
<https://www.kaggle.com/raddar/ricord-covid19-xray-positive-tests>. [En línea]
<https://www.kaggle.com/c/siim-covid19-detection/discussion/240187>.

39. **Yates, E., Yates, L. & Harvey, H. Machine learning-red dot: redes neuronales convolucionales profundas, en la nube y de código abierto en la clasificación de normalidad binaria de la radiografía de tórax.** Clin. Radiol. 73, 827–831. [En línea] 2018.
https://scholar.google.com/scholar_lookup?title=Machine%20learning-red%20dot%3A%20open-source%2C%20cloud%2C%20deep%20convolutional%20neural%20networks%20in%20chest%20radiograph%20binary%20normality%20classification&journal=Clin.%20Radiol.&volume=73&pages=.

40. **Annarumma, M. et al. Automated triaging of adult chest radiographs with deep artificial neural networks.** [En línea] 2019.
https://scholar.google.com/scholar_lookup?title=Automated%20triaging%20of%20adult%20chest%20radiographs%20with%20deep%20artificial%20neural%20networks&journal=Radiology&volume=291&pages=196-202&publication_year=2019&author=Annarumma%2CM.

41. *Redes convolucionales muy profundas para el reconocimiento de imágenes a gran escala.* **Simonyan, K. y Zisserman, A.** s.l. : ICLR, 2015. En Conferencia Internacional sobre Representaciones de Aprendizaje .

42. **Dunnmon, JA y col. Evaluación de redes neuronales convolucionales para la clasificación automatizada de radiografías de tórax. Radiología 290 , 537–544.** [En línea] 2019. https://scholar.google.com/scholar_lookup?title=Assessment%20of%20convolutional%20neural%20networks%20for%20automated%20classification%20of%20chest%20radiographs&journal=Radiology&volume=290&pages=537-544&publication_year=2019&author=Dunnmon%2CJA.

43. **Russakovsky, O. et al. Desafío de reconocimiento visual a gran escala de Imagenet. En t. J. Comput. Vis 115 , 211-252.** [En línea] 2015. https://scholar.google.com/scholar_lookup?title=Imagenet%20large%20scale%20visual%20recognition%20challenge&journal=Int.%20J.%20Comput.%20Vis&volume=115&pages=211-252&publication_year=2015&author=Russakovsky%2CO.

44. **RSNA, Datos.** [En línea] <https://www.kaggle.com/c/rsna-pneumonia-detection-challenge>.

45. **Nacenta, E.Martínez Chamorro A.Díez Tascón L.Ibáñez Sanz S.Ossaba Vélez S.Borrueal.** Diagnóstico radiológico del paciente con COVID-19. [En línea] <https://www.sciencedirect.com/science/article/abs/pii/S003383382030165X?via%3Dihub>.

46. **SIIM, Annotation Grading Methodology -.** [En línea] Junio de 2021. <https://www.kaggle.com/c/siim-covid19-detection/discussion/240250>.

47. **SIIM, Fuente de la metodología de anotación Clasificación.** [En línea] nov de 2020. https://journals.lww.com/thoracicimaging/Fulltext/2020/11000/Review_of_Chest_Radiograph_Findings_of_COVID_19.4.aspx.

48. **Radiología, Sociedad Norteamericana de.** [En línea] https://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0717-93082020000300088#f1.

49. **LovTechnology - El Lenguaje de Programación Python, Pros y Contras.** [En línea] <https://lovtechnology.com/lenguaje-programacion-python-pros-contras/>.

50. **Máster, Aranza García - Repositorio github - Trabajo Fin de.** https://github.com/aranza1972/github_Trabajo_Fin_Master. [En línea] https://github.com/aranza1972/github_Trabajo_Fin_Master.

51. **Detection, SIIM-FISABIO-RSNA COVID-19.** Kaggle.com. [En línea] 2021. <https://www.kaggle.com/c/siim-covid19-detection/data>.

52. **BIMCV-COVID19, Acuerdo investigación de conjunto de datos.** [En línea] <https://bimcv.cipf.es/bimcv-projects/bimcv-covid19/bimcv-covid19-dataset-research-use-agreement-2/>.

53. **COVID-19, BIMCV COVID-19 +: un gran conjunto de datos anotados de imágenes RX y CT de pacientes con.** [En línea] Junio de 2020. <https://arxiv.org/abs/2006.01174>.

54. **final, Foro de discusión KAGGEL - Clasificación.** [En línea] <https://www.kaggle.com/c/siim-covid19-detection/discussion/266057>.

55. **tamaños, xhlulu - Imágenes redimensionadas en JPG / PNG y diferentes.** [En línea] <https://www.kaggle.com/c/siim-covid19-detection/discussion/239918>.
56. **JPG, xhlulu - SIIM COVID-19: Redimensionado a 512px.** [En línea] <https://www.kaggle.com/xhlulu/siim-covid19-resized-to-512px-jpg>.
57. **Database, Tawsifur Rahman - COVID-19 Radiography.** [En línea] <https://www.kaggle.com/tawsifurrahman/covid19-radiography-database>.
58. **PNG, xhlulu - SIIM COVID-19: Redimensionado a 256px.** [En línea] <https://www.kaggle.com/xhlulu/siim-covid19-resized-to-256px-png>.
59. **python, scikit-image - image preprocessing in.** [En línea] https://scikit-image.org/docs/dev/api/skimage.exposure.html#skimage.exposure.match_histograms.
60. **Histogram Equalization - scikit-image.** [En línea] https://scikit-image.org/docs/dev/auto_examples/color_exposure/plot_equalize.html#sphx-glr-auto-examples-color-exposure-plot-equalize-py.
61. **preprocessing, Keras - image data.** [En línea] <https://keras.io/api/preprocessing/image/>.
62. **tf.keras.preprocessing.image.ImageDataGenerator.** [En línea] https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator.
63. **ImageNet.** [En línea] <http://www.image-net.org>.
64. **periférica, Wilson Alfredo - Memoria TFM Deteccion y clasificación de células normales de la sangre.** [En línea] 2020. http://openaccess.uoc.edu/webapps/o2/bitstream/10609/107806/6/wc_astrozTFM0120memoria.pdf.
65. **Nacelle, Andres.** [En línea] 2009. <http://www.nib.fmed.edu.uy/Seminario%202009/Monografias%20seminario%202009/Nacell-Redes%20NeuronalesImplementacion.pdf>.
66. **OPENGENUS.ORG, IQ-.** [En línea] <https://iq.opengenus.org/vgg16/>.