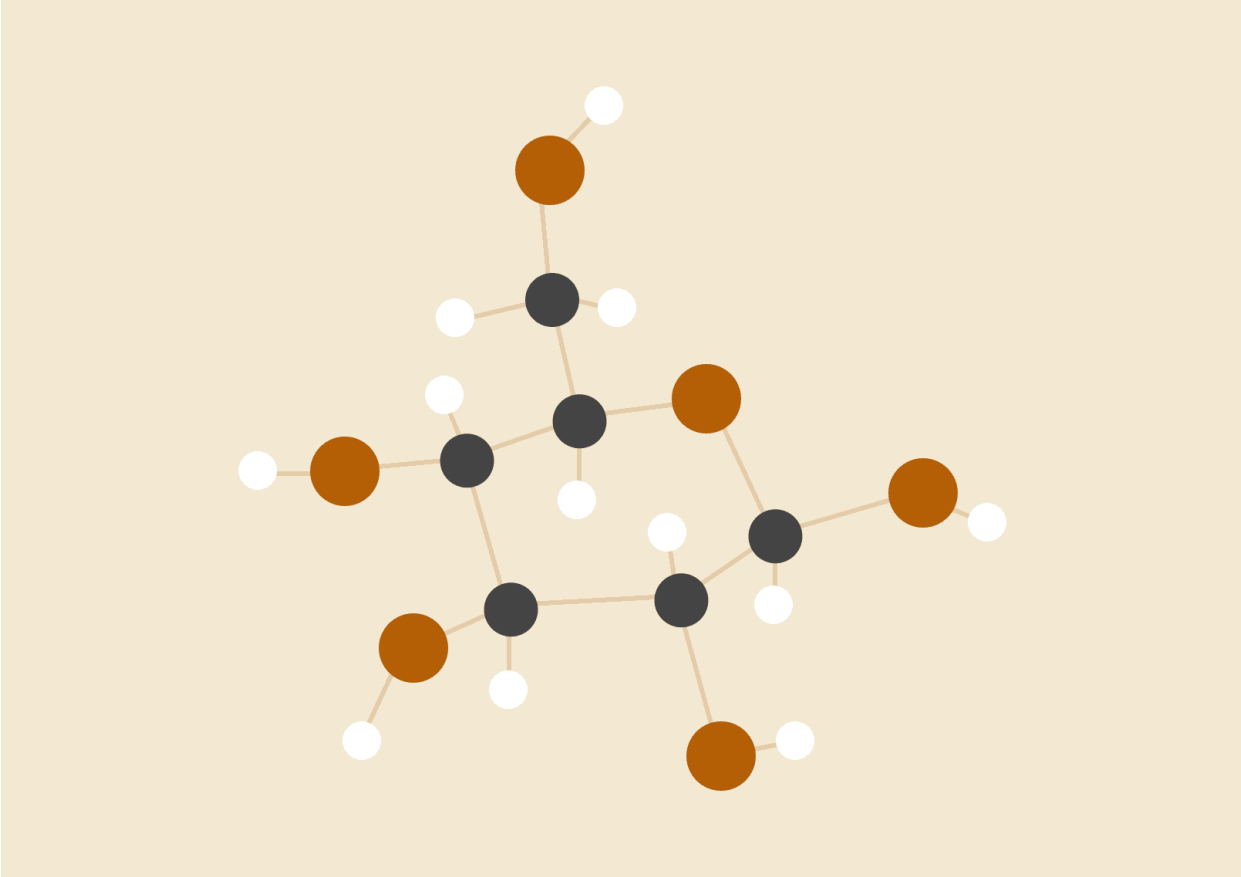
Máster Ingeniería Informática - E.T.S.I.I.T.

Clasificación de imágenes médicas

Sistemas Inteligentes para la Gestión en la Empresa - Trabajo de teoría



Óscar Jiménez Fernández, ***dni****

yoscar@correo.ugr.es

Lunes, 01 de junio de 2020

Contenidos

Descripción general del trabajo realizado

En el siguiente trabajo se pretende abordar el problema de **clasificación de imágenes médicas** empleando para ello ***Keras***, ***TensorFlow*** y ***Deep Learning***. En la literatura podemos encontrar diferentes casos de estudio o aplicaciones de las técnicas de visión por computador al campo de la detección automática de cualquier tipo de patología sobre imágenes médicas, ya sean tomadas mediante rayos-X, fotografías de análisis de biopsias por microscopio..., consiguiendo detectar desde **roturas** en los huesos, **células cancerosas**, o como será en nuestro caso, tipos de **neumonías**.

Se desarrollaran y estudiará el efecto de diversos modelos de clasificación basados en **redes neuronales convolucionales** complementados con técnicas como la **aumentación de datos** o **transferencia de aprendizaje**, sobre un conjunto de imágenes radiográficas de pulmones para tratar de discernir sobre la presencia o no de **neumonías** ocasionadas por **Covid-19**.

Como **recurso básico** para el desarrollo de este trabajo hemos empleado la siguiente **referencia**, donde se trata el mismo problema mencionado, que nos ha servido para asentar las bases del presente estudio.

Todo el contenido relativo al código puede ser consultado en el fichero .ipynb adjuntado junto a este documento. En este documento se describe por tanto a modo descriptivo todo el trabajo de estudio y análisis llevado a cabo, por lo que si desea conocer más acerca de los **aspectos** de **implementación**, puede dirigirse al siguiente **notebook**, o al fichero adjunto como he mencionado.

Entrando en contexto

Como desgraciadamente todos sabemos, la primera mitad de este año 2020 ha venido marcada por la pandemia de enfermedad por coronavirus, ocasionada por el virus **SARS-CoV-2** (coronavirus 2 del síndrome respiratorio agudo grave), identificado en diciembre de 2019 en la ciudad de **Wuhan** (República Popular China), siendo reconocido como **pandemia** por la OMS el 11 de marzo de 2020 [1].

El **alto grado** de **mortalidad** del COVID-19 llevó a que prácticamente el mundo entero se confinara en un estado de cuarentena para evitar la difusión de la enfermedad, tratando de **proteger** principalmente a los **grupos** considerados de **riesgo**, como personas mayores o aquellas afectadas de patologías respiratorias [2].

Situación actual

Ilustración 1. Situación de casos confirmados a día 8 de junio de 2020 por ***mscbs***

La comunidad científica de todo el mundo se volcó en el problema, aportando su grano de arena y tratando de buscar soluciones, ya fuera desde la investigación de vacunas, tratamientos, o uno de los problemas que más apremiaba, como era la posibilidad de disponer de modelos o **técnicas** de **clasificación** rápida para validar o descartar la presencia del virus en un paciente.

Encontramos de este modo, la rama de investigación que nos concierne en este trabajo, como es la aplicación de técnicas de **visión por computador** para la **detección** de este tipo de **neumonía** causada por el coronavirus mediante **imágenes radiográficas** de los pulmones.

Investigaciones actuales

Desde la propia **Universidad de Granada**, podemos destacar la labor realizada por el Instituto Andaluz de Investigación de Ciencias de Datos en Inteligencia Artificial, junto al servicio de Radiodiagnóstico del Hospital Universitario Clínico San Cecilio de Granada, que trabajan conjuntamente en un sistema de detección automática para afecciones pulmonares producidas por Covid-19 a través de radiografías de tórax de los pacientes [3].

Relacionado con el anterior proyecto, encontramos dos estudiantes de ingeniería de la **Universidad Europea de Madrid** y electrónica y comunicaciones en la **Universidad de Hertfordshire**, que desarrollaron un modelo para la detección de infecciones por Covid-19 a través de radiografías de tórax [4].

Desde otro campo, podemos encontrar investigaciones como la de los científicos del **CSIC** que buscan detectores de SARS-CoV-2 más rápidos y baratos que las PCR con balizas moleculares. Este se basa en 'sensores' de ADN que reaccionan emitiendo fluorescencia en presencia del ARN del coronavirus, aplicables a gran escala [5].

Obtención del conjunto de datos

El principal **problema** que nos hemos encontrado a la hora de abordar este estudio, es la **falta** de un gran **conjunto** de **imágenes** radiográficas de tórax con **Covid-19 positivo**, lo que dificulta por tanto el entrenamiento de una red de modo que nos permita generar un modelo válido, capaz de generalizar y obtener una tasa de acierto alta para nuevas entradas.

Aún así, hemos podido localizar un repositorio, mencionado en el recurso base, donde se trata de componer un conjunto de imágenes de este tipo para estudios relacionados al servicio de la comunidad. Desde ésta hemos podido extraer un total de **198 radiografías** de tórax con **Covid-19 positivo** [6], lo que en comparación con las 25 imágenes utilizadas en el recurso base, nos ha permitido componer conjuntos más robustos que poder emplear para entrenamiento, validación y test. Para la **extracción** de dicho **conjunto** hemos empleado el siguiente **script**.

Para poder componer el modelo adecuadamente, debemos disponer de igual forma de **imágenes** radiográficas de tórax de **pacientes sanos**, permitiendo al modelo aprender las características de cada tipo de imagen. Estas las hemos obtenido del conjunto de **train** del siguiente dataset que podemos encontrar en Kaggle [7], el cual distingue entre imágenes 'normales' (pacientes sanos) y afectadas por neumonías, obteniendo así un total de **1341 instancias** de **pacientes sanos**.

Debemos de **destacar** que podíamos obtener un mayor de instancias positivas, pero hay que tener en cuenta que las radiografías pueden ser tomadas atendiendo a diferentes **vistas**, como por ejemplo ***posteroanterior (PA), anteroposterior (AP)*** o ***anteroposterior supina***. En nuestro caso, las instancias de **pacientes sanos** son del tipo **PA**, por lo que para evitar errores de clasificación o resultados adulterados derivados de que todas las instancias del tipo de vista AP son del tipo Covid-19, hemos **filtrado** y

seleccionado sólo aquellas del tipo **AP**. Si no tuviéramos esto en cuenta, estaríamos introduciendo un sesgo y una falsa ilusión de acierto del modelo, que en caso de su aplicación a radiografías del tipo AP, llevaría al modelo a una situación de falsos positivos, ya que estaría asociando las características propias de la vista con esta enfermedad, lo que no tiene nada que ver.

Tenemos por tanto un conjunto de imágenes de **1539 instancias**, con una proporción de clases **desbalanceadas**, del tipo de vista **AP**, sobre las que se ha aplicado un cambio de tamaño para homogeneizarlas, de **224 x 224 píxeles**, con **3 canales**.

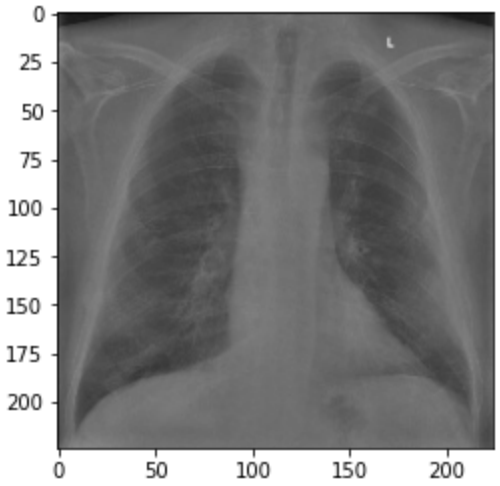


Ilustración 2. Covid-19 positivo

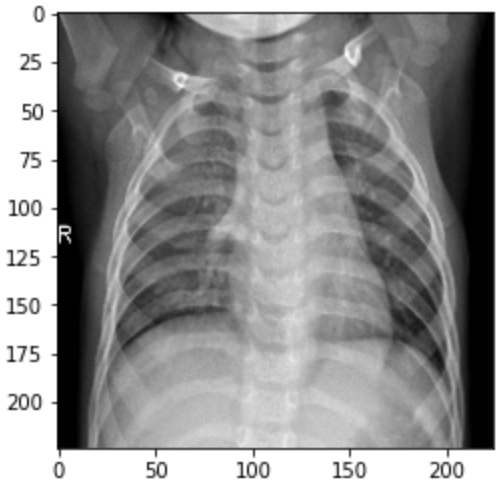


Ilustración 3. Paciente sano

Balanceo del conjunto de datos

Como podemos deducir del anterior apartado, si usaramos el total de instancias recogidas, tendríamos un **conjunto** fuertemente **desbalanceado**, con un fuerte **sesgo** para los **casos positivos**, los cuales nos interesa detectar, por lo que para solucionar este problema, realizaremos un **undersampling**, componiendo una muestra aleatoria del mismo tamaño que el conjunto de positivos, para el conjunto de pacientes sanos.

Con ello, obtenemos un **conjunto final** formado por **396 instancias**, totalmente **balanceado**.

Estructura del conjunto y particiones

Una vez tenemos las instancias del conjunto balanceado, lo hemos **organizado** en una **estructura** de **carpetas**, de modo que en apartados posteriores, al hacer uso de la

aumentación de datos, podamos tener una estructura clara y realizar el proceso desde el directorio sin tener que preocuparnos por más.

Distinguimos así un **directorio** formado por 3 **carpetas**, pertenecientes a las instancias de **entrenamiento**, **validación** y **test**, formadas a su vez cada una por dos carpetas, nombradas como ***covid*** y ***healthy***, que contendrán cada una las instancias de dicha partición etiquetadas como los respectivos directorios.

Las particiones se han realizado usando un **80%** para **entrenamiento y validación**, y el **20%** restante para **test**, todo ello de manera **aleatorizada**. Del 80% se ha obtenido una nueva proporción, destinando el **75%** a **entrenamiento**, y el **25%** a **validación**. Este proceso se ha realizado independientemente para cada una de las clases, por lo que dichos conjuntos se encuentran perfectamente balanceados, teniendo así para cada clase un conjunto de 118 instancias de entrenamiento, 40 instancias de validación, y 40 para test, lo que nos deja un **total** de **236 instancias** para **entrenamiento**, **80** para **validación** y **80** para **test**.

Data Augmentation

Para poder permitir una **mayor** capacidad de **generalización**, haciendo el modelo menos sensible a posibles alteraciones o ruido en las imágenes, como **desplazamientos** del paciente hacia un lado u otro, **mala calidad** de las radiografías, o diferentes zooms sobre estas, derivados de diferentes distancias a la hora de tomar las radiografías, se ha utilizado una de los métodos de aumentación de datos que nos ofrece keras, permitiéndonos definir **rangos** de **rotaciones**, **zooms**, **flips** o **reescalados** entre otras posibilidades.

Además, se han definido lotes de tamaño 32, de manera aleatorizada, con las clases categorizadas, para posteriormente poder usar la entropía cruzada categórica.

Aproximación CNN inicial

Como **primera aproximación**, hemos definido una **CNN** empleando tanto capas convolucionales, de pooling y densas, como capas adicionales para evitar problemas de **sobreajuste** en el proceso de aprendizaje. Podemos encontrar en la literatura numerosos métodos, como por ejemplo **dropout** para aumentar así la resistencia al aprendizaje y

evitar que el modelo dependa demasiado de determinadas neuronas, o **normalización** para modificar las salidas de las neuronas con el objetivo de que se mantengan dentro de un rango y hacer el modelo menos dependiente del preprocesamiento e inicialización de los pesos [8].

Adicionalmente, para poder aplicar los diferentes filtros convolucionales sin que la dimensionalidad sea reducida perdiendo así información relevante, como consecuencia del stride y el tamaño del kernel, hemos establecido la opción de **padding** a 'same', permitiendo así que tras cada paso por la capa convolucional, la dimensionalidad no será reducida, y podremos **encadenar** así los diferentes **filtros** [9].

Tras esta primera parte de filtros convolucionales, encontramos la capa **full-conected**, con una capa de salida del tipo softmax, de modo que podamos obtener la distribución de probabilidad de pertenencia de cada instancia a cada una de las clases.

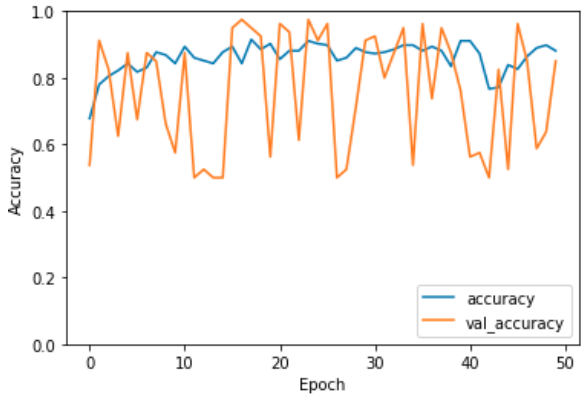
Layer (type)	Output Shape	Param #
conv2d_232 (Conv2D)	(None, 224, 224, 64)	9472
conv2d_233 (Conv2D)	(None, 224, 224, 64)	36928
max_pooling2d_5 (MaxPooling2	(None, 112, 112, 64)	0
dropout_5 (Dropout)	(None, 112, 112, 64)	0
conv2d_234 (Conv2D)	(None, 112, 112, 64)	200768
conv2d_235 (Conv2D)	(None, 112, 112, 64)	36928
average_pooling2d_6 (Average	(None, 56, 56, 64)	0
conv2d_236 (Conv2D)	(None, 56, 56, 32)	100384
conv2d_237 (Conv2D)	(None, 56, 56, 32)	9248
conv2d_238 (Conv2D)	(None, 56, 56, 32)	9248
average_pooling2d_7 (Average	(None, 28, 28, 32)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_8 (Dense)	(None, 64)	1605696
batch_normalization_204 (Bat	(None, 64)	256
dropout_6 (Dropout)	(None, 64)	0
dense_9 (Dense)	(None, 2)	130

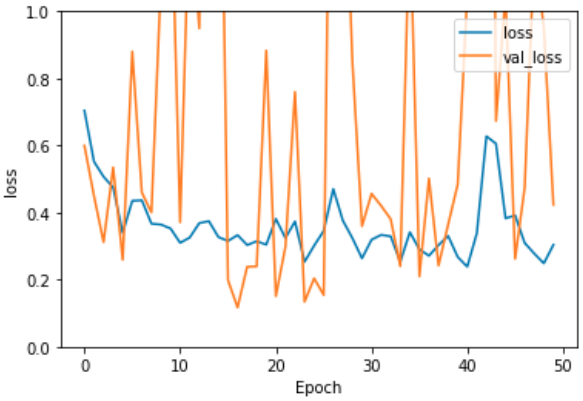
Total params: 2,009,058
Trainable params: 2,008,930
Non-trainable params: 128

Como parámetros del modelo compilado usaremos la función de pérdida ***categorical_crossentropy***, que tratará de minimizar el error cometido en dicha distribución de probabilidad de la capa softmax. Como optimizador hemos usado ***Nadam***, basado en Adam, con momento de Nesterov [10]. La métrica empleada será el valor de accuracy obtenido para la clasificación de las instancias. Adicionalmente, con las predicciones sobre el conjunto de test una vez entrenado el modelo, mostraremos la matriz de confusión para detectar los principales problemas del modelo.

En estos casos, nuestro principal **objetivo** será **reducir** los **falsos negativos**, entendiendo como caso positivo el hecho de detección de coronavirus en el paciente. Priorizamos este caso frente al caso de falsos positivos, puesto que sería peor calificar un caso positivo como negativo, que decir a un paciente sano que presenta la enfermedad, aunque no podemos dejar este caso contrário de lado, puesto que podría resultar muy perjudicial tratar a un paciente de una patología que no padece.

Los **resultados** obtenidos del **entrenamiento** presentan una gran fluctuación entre cada una de las épocas, ocasionado posiblemente por una tasa de aprendizaje demasiado elevada.





Los resultados obtenidos sobre **test** arrojan unos resultados iniciales con un 100% de precisión para la detección de casos positivos, aunque muestra ciertos problemas con los falsos positivos como podemos observar en las siguientes métricas recogidas:

```
[[28 12]
 [ 0 40]]
```

	precision	recall	f1-score	support
covid	1.00	0.70	0.82	40
healthy	0.77	1.00	0.87	40
accuracy			0.85	80
macro avg	0.88	0.85	0.85	80
weighted avg	0.88	0.85	0.85	80

Como futuros pasos en el desarrollo del trabajo, trataremos de aplicar nuevas técnicas para tratar de mejorar los resultados obtenidos, no solo de cara al conjunto de test, sino de cara a corregir las fuertes fluctuaciones del modelo, obteniendo un mayor poder de generalización.

Transfer-Learning

La técnica de **transferencia** de **aprendizaje** consiste en tomar características aprendidas en un problema y aprovecharlas para un nuevo problema similar. Generalmente, su uso viene motivado por problemas donde el conjunto de datos tiene un reducido número de instancias con el que poder entrenar el modelo a escala completa desde cero [11].

Para ello, se toman las capas de un **modelo pre-entrenado**, se **congelan** para evitar perder la información de los **pesos** de las capas **convolucionales** tras el entrenamiento, y **añadimos** nuevas **capas** que serán entrenadas para tratar de **aprender** a **convertir** las **características** detectadas en **predicciones** para el nuevo conjunto de datos.

Adicionalmente, podemos considerar una etapa adicional, denominada ***fine-tuning***, por la cual, las **capas** del modelo que habíamos congelado, son **descongeladas**, y las entrenamos con bajos valores de learning-rate para evitar causar grandes variaciones en los pesos de éstas. Este es un proceso delicado debido precisamente a la naturaleza del problema, ya que la red tratará de corregir los pesos para los nuevos datos, pudiendo generar grandes variaciones, sin implicar por ello mejores resultados. En este trabajo no lo hemos considerado, pero de cara a obtener otras aproximaciones y seguir investigando, sería una posible opción a tomar.

Keras nos **ofrece** numerosas **aplicaciones** o modelos de aprendizaje profundo que están disponibles junto **con pesos pre-entrenados** [12].

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-
EfficientNetB0	29 MB	-	-	5,330,571	-
EfficientNetB1	31 MB	-	-	7,856,239	-
EfficientNetB2	36 MB	-	-	9,177,569	-
EfficientNetB3	48 MB	-	-	12,320,535	-
EfficientNetB4	75 MB	-	-	19,466,823	-
EfficientNetB5	118 MB	-	-	30,562,527	-
EfficientNetB6	166 MB	-	-	43,265,143	-
EfficientNetB7	256 MB	-	-	66,658,687	-

Para el desarrollo de nuestro estudio, hemos seleccionado 3 de estos modelo que hemos adaptado a nuestro problema siguiendo esta técnica descrita y que hemos probado sobre el conjunto de test para poder comprarlos y obtener el modelo más idóneo para

resolver nuestro problema. Estos modelos seleccionados son ***VGG16*** [13], ***NasNetLarge*** [14] y ***InceptionResNetV2*** [15], empleando los pesos del conjunto ***Imagenet*** [16].

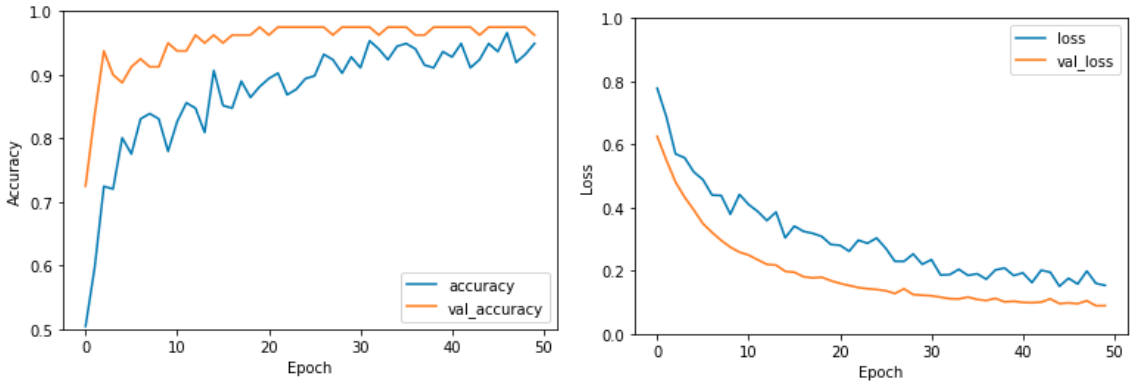
Como hemos descrito, usaremos la parte convolucional de cada uno de los tres modelos, a los que añadiremos una nueva parte de capa densa y de salida tipo **softmax**, unidas a la primera parte de la red por una capa **flatten**. Puesto que las topologías de estas redes tienen una mayor profundidad, no serán ilustradas en este documento, aunque pueden ser visualizadas en el notebook elaborado para este trabajo.

Los hiper-parámetros empleados para cada uno de los modelos serán los mismos que los indicados para el modelo inicial.

A continuación, mostraremos los resultados del entrenamiento obtenidos de cada modelo, junto con las métricas de evaluación sobre el conjunto de test.

VGG16

Los resultados de **entrenamiento** obtenidos para el primer modelo son los siguientes:



Como podemos apreciar, ahora sí tenemos un proceso de aprendizaje sin tantas fluctuaciones tanto para la función de pérdida como de accuracy. Destacamos el **mejor resultado** del conjunto de **validación sobre** el de **entrenamiento**, lo que es razonable teniendo en cuenta que dicho conjunto se encuentra aumentado con las variaciones indicadas anteriormente, por lo que para el conjunto de validación, que tiene las imágenes originales, es normal como indicamos, obtener un mejor resultado.

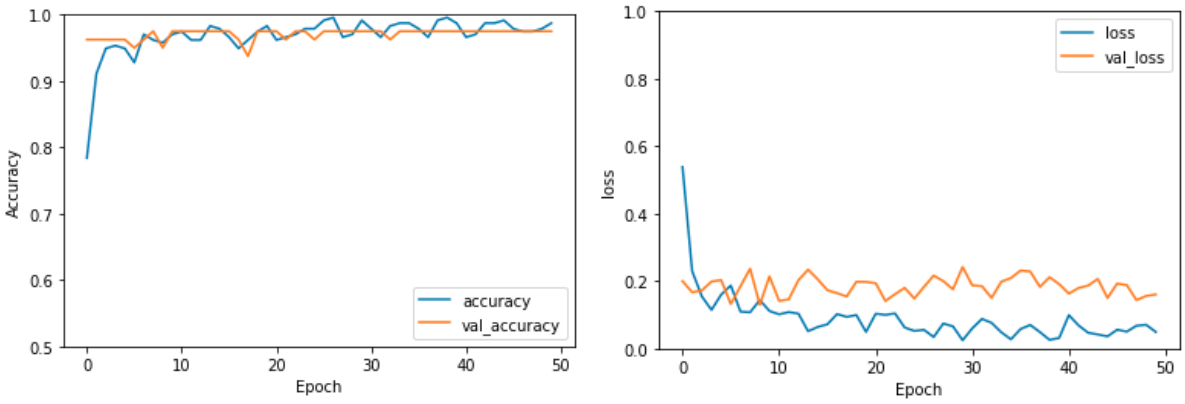
Si evaluamos el conjunto de **test**, podemos obtener los siguientes resultados:

```
[[39  1]
 [ 2 38]]

              precision    recall  f1-score   support

       covid       0.95      0.97      0.96        40
     healthy       0.97      0.95      0.96        40

    accuracy                           0.96        80
   macro avg       0.96      0.96      0.96        80
weighted avg       0.96      0.96      0.96        80
```

Analizando el resultado de la **matriz** de **confusión**, junto al valor de accuracy, podemos ver como hemos mejorado considerablemente el error de clasificación del modelo, aunque ello nos ha llevado a reducir la precisión de cada una de las clases, pudiendo observar en este caso **2 falsos negativos**, además de un falso positivo, destacando así la mejora del modelo en este segundo aspecto.

NasNetLarge

Los resultados de **entrenamiento** obtenidos para el segundo modelo son los siguientes:



Como podemos observar, el **número** de **épocas** necesarias para obtener unos **resultados** similares al anterior son bastante más **reducidas**, pudiendo cortar perfectamente el entrenamiento en 20 épocas.

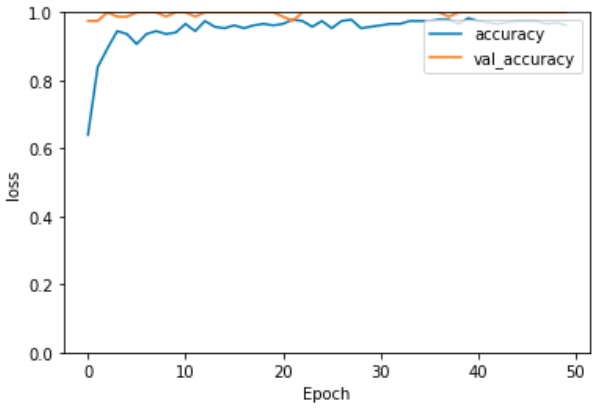
Respecto al conjunto de **test**, obtenemos los siguientes valores:

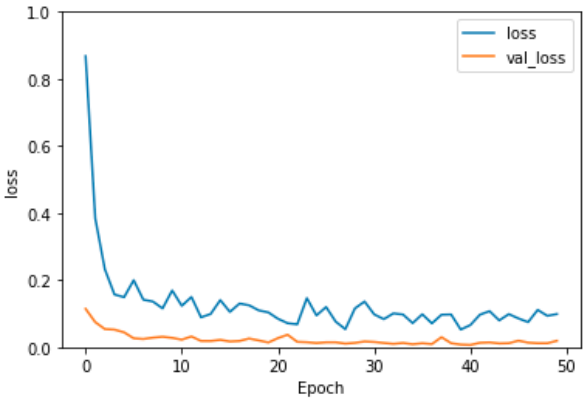
```
[[38  2]
 [ 0 40]]

              precision    recall  f1-score   support

       covid       1.00      0.95      0.97        40
     healthy       0.95      1.00      0.98        40

    accuracy                           0.97        80
   macro avg       0.98      0.97      0.97        80
weighted avg       0.98      0.97      0.97        80
```

En este caso, hemos conseguido **mejorar** el anterior modelo, aumentando el valor de clasificación, con un **100% de precisión** sobre la **clase positiva**, manteniendo un mejor resultado con respecto al modelo base para la clase negativa, aunque con cierto error, encontrando aún 2 falsos positivos.

InceptionResNetV2

Los resultados de **entrenamiento** obtenidos para el tercer y último modelo estudiado son los siguientes:





En este caso, podemos decir que el modelo presenta un mejor aspecto, consiguiendo obtener un valor en validación de prácticamente el 100%, y un bajo valor de pérdida, sin grandes fluctuaciones en el proceso de entrenamiento.

Si evaluamos el modelo sobre el conjunto de **test**, ahora si podemos comprobar como consigue un **100% de acierto en clasificación**:

```
[[40  0]
 [ 0 40]]

              precision    recall  f1-score   support

       covid       1.00      1.00      1.00        40
     healthy       1.00      1.00      1.00        40

    accuracy                           1.00        80
   macro avg       1.00      1.00      1.00        80
weighted avg       1.00      1.00      1.00        80
```

Con este tercer modelo hemos conseguido hacer pleno, consiguiendo así un 100% de precisión para cada una de las clases, **logrando** nuestro **objetivo** inicial, y resolviendo un problema de tanta relevancia en el panorama actual como es este, sin descuidar que estas mismas técnicas pueden ser aplicadas a otros numerosos problemas de las misma índole médica.

Análisis y conclusiones finales

A continuación mostramos una **tabla resumen** de las **métricas** obtenidas para cada uno de los **modelos** estudiados, respecto de los conjuntos de **entrenamiento** y **test**:

Red	Entrenamiento		Test precision		Test accuracy	Parámetros entrenables	Tiempo de entrenamiento por época (s)
	Accuracy	Loss	Healthy	Covid			
CNN	0.8814	0.3	0.77	1.0	0.85	2,008,930	4
VGG16	0.9492	0.15	0.97	0.95	0.96	32,962	4
NasNetLarge	0.9873	0.04	0.95	1.0	0.97	258,242	6
InceptionResNetV2	0.9619	0.09	1.0	1.0	1.0	98,498	4

Como podemos observar, en relación al **tiempo** de entrenamiento de cada uno de los modelos, todos presentan un tiempo por época similar, distanciandose algo más el caso de ***NasNetLarge***, que presenta un gran número de **parámetros** entrenables, que aunque no alcanza el valor del modelo inicial, la **profundidad** de ésta es mucho más elevada.

En el cómputo global, comprobamos como el modelo **InceptionResNetV2**, además de ofrecer un **accuracy del 100%**, nos ofrece un **bajo tiempo** de entrenamiento por época. Si atendemos al gráfico de entrenamiento, podemos observar cómo aunque el número de épocas que hemos empleado en todos los casos ha sido el mismo (50 épocas), podríamos reducirlo, como mínimo hasta 20 épocas, logrando posiblemente obtener los mismos resultados sobre el conjunto de test.

Se evidencia de este modo la **importancia** de técnicas como la **aumentación** de **datos** cuando tratamos con problemas en los que el número de instancias es reducido, además del poder de **generalización** que estas técnicas ofrecen, evitando el sobreajuste de la red al conjunto de entrenamiento.

Como segundo factor clave, mencionamos el gran potencial de las técnicas de **transferencia de aprendizaje**, gracias a la cual hemos podido conseguir esa precisión del 100% sobre el conjunto, y todo ello sin necesidad de complicar nuestro modelo o proceso de aprendizaje demasiado, y con unos tiempos muy asequibles, que de otra manera, jamás habríamos podido obtener, si necesitamos entrenar las capas de la parte convolucional de la red.

De cara a **planteamientos futuros**, podríamos complicar aún más el problema, obteniendo radiografías de otras **neumonías causadas por otro agente**, que podemos encontrar además en el mismo conjunto del que hemos extraído las empleadas en este estudio, para tratar así de identificar no solo los pacientes sanos de los afectados, sino ser capaces de discernir la causa de dicha neumonía (Covid-19, MERS, Bacteriana, Viral...).

Se propone la **aplicación** de técnicas como el **fine-tuning** que hemos comentado, tratando así de entrenar esa parte convolucional de las redes para estos casos, logrando resultados satisfactorios.

Referencias

[1] https://es.wikipedia.org/wiki/Pandemia_de_enfermedad_por_coronavirus_de_2019-2020

[2] https://www.mscbs.gob.es/en/profesionales/saludPublica/ccayes/alertasActual/nCov-China/situacionActual.htm

[3] https://www.granadahoy.com/granada/Detectar-coronavirus-rayos-X-avance-cientifico-Granada_0_1455754632.html

[4] https://as.com/deporteyvida/2020/05/10/portada/1589111967_660802.html

[5] https://www.csic.es/es/actualidad-del-csic/cientificos-del-csic-buscan-detectores-de-sars-cov2-mas-rapidos-y-baratos-que

[6] https://github.com/ieee8023/covid-chestxray-dataset

[7] https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

[8] https://www.tensorflow.org/tutorials/images/cnn

[9] https://www.machinecurve.com/index.php/2020/02/08/how-to-use-padding-with-keras/

[10] https://keras.io/api/optimizers/Nadam/

[11] https://keras.io/guides/transfer_learning/

[12] https://keras.io/api/applications/

[13] https://keras.io/api/applications/vgg/#vgg16-function

[14] https://keras.io/api/applications/nasnet/#nasnetlarge-function

[15] https://keras.io/api/applications/inceptionresnetv2/

[16] https://www.learnopencv.com/keras-tutorial-using-pre-trained-imagenet-models/