Apoyo al proceso de diagnóstico de neumonía asociada al COVID-19 utilizando modelos de Redes Neuronales Convolucionales, con Keras y Tensorflow

Jhan Carlos Caya Pérez, Ing[1], Pedro Huamaní-Navarrete, Dr[2],
[1,2]Ricardo Palma University, Peru, jhan.cayap@gmail.com, phuamani@urp.edu.pe

Abstract*-El propósito de este artículo fue evaluar el desempeño de tres modelos de Redes Neuronales Convolucionales para apoyar al proceso de diagnóstico de neumonía asociada al COVID-19, con el fin de reducir el tiempo que toma la labor de detección de la enfermedad en radiografías de tórax. Los modelos utilizados fueron: ResNet50, InceptionV3 y un modelo particular; para la implementación de dos de los modelos, ResNet50 e InceptionV3, se requirió aplicar transfer learning, mientras que para los tres se necesitó de la técnica data augmentation. Asimismo, la implementación se realizó utilizando las bibliotecas Keras y Tensorflow; y por otro lado, para el entrenamiento y validación se empleó un dataset compuesto por casos positivos (COVID-19) y casos negativos (NO COVID-19). Finalmente, se determinó que el modelo más efectivo fue el InceptionV3 con un accuracy de 0.9886, cuando se entrenó con Data Augmentation y 0.9848 sin la aplicación de Data Augmentation.

Keywords*—Red Neuronal Convolucional, Keras, Tensorflow, Transfer Learning, Data Augmentation.

I. INTRODUCCIÓN

En diciembre del 2019, la población mundial se enteró de la existencia del COVID-19, la cual fue declarada como pandemia mundial por parte de la OMS en marzo del siguiente año. Esto motivó a muchos países a establecer protocolos para el cuidado preventivo y detección de esta enfermedad, así como controlar la cantidad de infectados, dado que esta enfermedad afecta principalmente al sistema respiratorio y en algunos casos desarrolla neumonía severa. De esta manera, la presente investigación tiene por finalidad brindar apoyo al diagnóstico de neumonía asociada al COVID-19, por medio de la evaluación de tres modelos de redes neuronales convolucionales para la clasificación de radiografías de tórax. Los modelos evaluados fueron ResNet50, InceptionV3 y uno en particular; asimismo, debido a la problemática existente en cuanto al tiempo empleado para la detección de la enfermedad por medio de la revisión y análisis de radiografías de tórax, se optó por desarrollar este trabajo para ayudar a acelerar el tratamiento temprano al paciente, evitando mayores complicaciones en su salud. Para este desarrollo, se utilizó un DataSet conformado por imágenes de Radiografías de Tórax, que sirvió para entrenar cada uno de los modelos de Redes Neuronales Convolucionales propuestos, así como también se utilizaron las bibliotecas Keras con TensorFlow en Python en una laptop convencional.

Existen algunos estudios previos que han sido planteados como soluciones ante la problemática existente, tal es el caso de [1] quien presenta un estudio sobre el funcionamiento de las redes neuronales convolucionales y su aplicación en clasificación de imágenes, particularmente los casos de los modelos LeNet, AlexNet, VGG (VGG-16 y VGG-19), GoogleNet, ResNet y DenseNet, de los cuales para la parte experimental utiliza ResNet y CancerNet, siendo este último un modelo no oficial pero semejante al funcionamiento del VGG. Además, para la implementación emplea la técnica data augmentation; posteriormente, desarrolla un análisis sobre los lenguajes de programación más destacados y por qué se declina por Python utilizando librerías como Tensorflow y Keras.

Por otro lado, en [2] se proponen dos arquitecturas de redes neuronales convolucionales, una con capa de Dropout y otra sin capa Dropout, además ambas se encuentran estructuradas por una capa de convolución, un max pooling y una capa de clasificación diseñadas desde cero para detectar la neumonía a partir de imágenes de rayos X de tórax.

De igual manera, en [3], se presenta un nuevo modelo de red neuronal convolucional para la detección automática del COVID-19 mediante imágenes de rayos X de tórax; donde la base de datos que se utilizó está compuesta por otras bases de datos, obtenidas de plataformas como GitHub y Kaggle, además utilizó una cantidad de datos balanceados con la intención de optimizar la clasificación de las imágenes. El modelo posee una precisión de clasificación del 98.08% para las clases binarias y 87.02% para los casos de clases múltiples.

Y, como también, en [4], se presenta un estudio referente a cinco modelos basados en redes neuronales convolucionales pre-entrenados (ResNet50, ResNet101, ResNet152, InceptionV3 e Inception-ResNetV2), los cuales fueron propuestos para la detección de pacientes infectados con neumonía por coronavirus usando imágenes de radiografías de tórax.

II. MARCO TEÓRICO

A continuación, se indican algunas definiciones importantes y utilizadas en el desarrollo de este artículo.

A. Red Neuronal Convolucional (CNN)

Según [5], es un tipo de red neuronal artificial profunda de aprendizaje supervisado, con una considerable cantidad de capas ocultas.

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2021.1.1.566
ISBN: 978-958-52071-8-9 ISSN: 2414-6390

B. *Modelo de CNN ResNet50*

Según [6] es un modelo de red neuronal convolucional. Cuenta con 50 capas y su bloque base está compuesto por tres convoluciones secuenciales, una convolución de 1x1, una convolución de 3x3 y una convolución de 1x1.

C. *Modelo de CNN InceptionV3*

Según [6] es un modelo de red neuronal convolucional. Cuenta con 48 capas, clasifica hasta 1000 clases diferentes, y acepta hasta un tamaño de entrada máximo de 299x299.

D. *Métricas de Evaluación*

Utilizadas para evaluar el desempeño del modelo de CNN. Entre ellas, se tiene a la Matriz de Confusión, Accuracy, Precision, Recall, Specificity y F1-Score.

E. *Transfer Learning (TL)*

Según [7], es una técnica de aprendizaje empleada en Deep Learning, y utilizada en modelos de redes neuronales pre-entrenadas o redes neuronales demasiado profundas. Consiste en modificar la arquitectura de la red neuronal agregando nuevas capas, las cuales serán las únicas en ser entrenadas, mientras que el resto de capas serán "congeladas"; de esta forma, proporciona un menor esfuerzo computacional en el proceso de entrenamiento.

F. *Data Augmentation (DA)*

Según [6], es una técnica de aprendizaje que consiste en aumentar la cantidad de datos de forma artificial; por ejemplo, girar la imagen, realizar un zoom, cambiar el brillo, reflejar la imagen, entre otros, para obtener un conjunto mayor de imágenes para las etapas de entrenamiento y validación de la red neuronal.

G. *Diagnóstico de Neumonía*

Requiere de una radiografía de tórax, previo a una exploración física donde el médico evalúa los síntomas del paciente: tos, fiebre y dificultad para respirar. Sumado a ello, se realizan otras pruebas como el hemograma para determinar cuál es el tipo de agente que está provocando la enfermedad, y el tipo de severidad de la misma.

III. Diseño de ingeniería

En esta sección se muestra el desarrollo de este trabajo; en particular, los primeros 5 sub bloques representados en el diagrama de bloques general de la Figura 1. Es así que, el primer sub bloque corresponde a la adquisición de datos y generación del DataSet, el segundo al preprocesamiento de imágenes utilizando la biblioteca Keras con TensorFlow en Python, asimismo se considera la inclusión de la técnica de aprendizaje Data Augmentation. Luego, del tercer al quinto bloque, se desarrolla la implementación y el entrenamiento de cada uno de los modelos de CNN de forma independiente. Asimismo, se requirió de la aplicación de transfer learning para la implementación de los modelos ResNet50 e InceptionV3 al ser modelos muy profundos. Por otro lado, se realizaron dos pruebas por proceso de entrenamiento de cada modelo implementado, con la finalidad de analizar la utilidad de Data Augmentation en los mismos. En cuanto a los dos últimos bloques, serán abordados en la siguiente sección con las pruebas de predicción y comparación de resultados.

A. *Adquisición de datos y generación del DataSet*

En este trabajo se utilizaron dos bases de datos diferentes de imágenes de radiografías de tórax de casos con COVID-19 y casos NO COVID-19. Estas bases de datos pertenecen a la Sociedad Italiana de Radiología Médica e Intervencionista (SIRM) [8], y al usuario Prashant Patel [9] quien tiene su base de datos publicada en la plataforma Kaggle; además, ambas bases de datos son de acceso gratuito. Se eligió la base de datos de Patel porque, a su vez, se encuentra conformada por las bases de datos de Cohen perteneciente a la plataforma GitHub [10], y Mooney en la plataforma Kaggle [11], las cuales fueron utilizadas por autores como [3], [4] y [12] para la realización de sus proyectos de investigación, dado que las consideran fuentes confiables de información.

Asimismo, la implementación del algoritmo para generar el DataSet personal e implementación de algoritmos posteriores, requirió la instalación de la interfaz gráfica de usuario Anaconda Navigator, dado que permite la creación de entornos personalizados en los cuales se puede ejecutar distintos editores de desarrollo como Jupyter Notebook, Spyder y VSC, así como también instalar diferentes versiones del lenguaje de programación Python con sus librerías respectivas. La elección de Jupyter Notebook como editor de desarrollo se debió a [12].

Por otro lado, se implementó el algoritmo para la generación del DataSet, el cual tuvo un desarrollo basado en la importación de librerías Pandas, Os y Shutil, e inicialización de la selección y extracción de imágenes médicas positivas a COVID-19 de la base de datos del SIRM (ver la Figura 2); para esto, se importó el archivo "COVID-19.metadata.xlsx", y con el comando "pd.read_excel" se hizo la lectura; posteriormente, con el comando print(df.shape) se mostró la base de datos descargada con un total de 219 imágenes divididas en 4 columnas, de las cuales solo 86 imágenes corresponden a radiografías de Tórax. Ver la Figura 3.

Posteriormente, en forma manual, se copiaron 576 imágenes de radiografías de tórax asociadas a la base de datos de Patel. Por lo tanto, el grupo total ascendió a 662 imágenes para la carpeta de COVID-19. Luego, con los comandos "random.shuffle" y "shutil.copy2", se copiaron aleatoriamente 662 nuevas imágenes de casos normales hacia la carpeta NO_COVID_19. Con ello, se logra una data balanceada y mejores resultados durante la etapa del entrenamiento de los modelos de redes neuronales convolucionales.

Por último, se subdividió la cantidad de imágenes del DataSet en tres conjuntos: Train, Val y Test; no obstante, la cantidad de imágenes en cada conjunto se basó en el criterio de equilibrio de aprendizaje que explica [1], el cual consiste en dividir el total de datos en un 80% para el conjunto de entrenamiento y 20% para el de test.

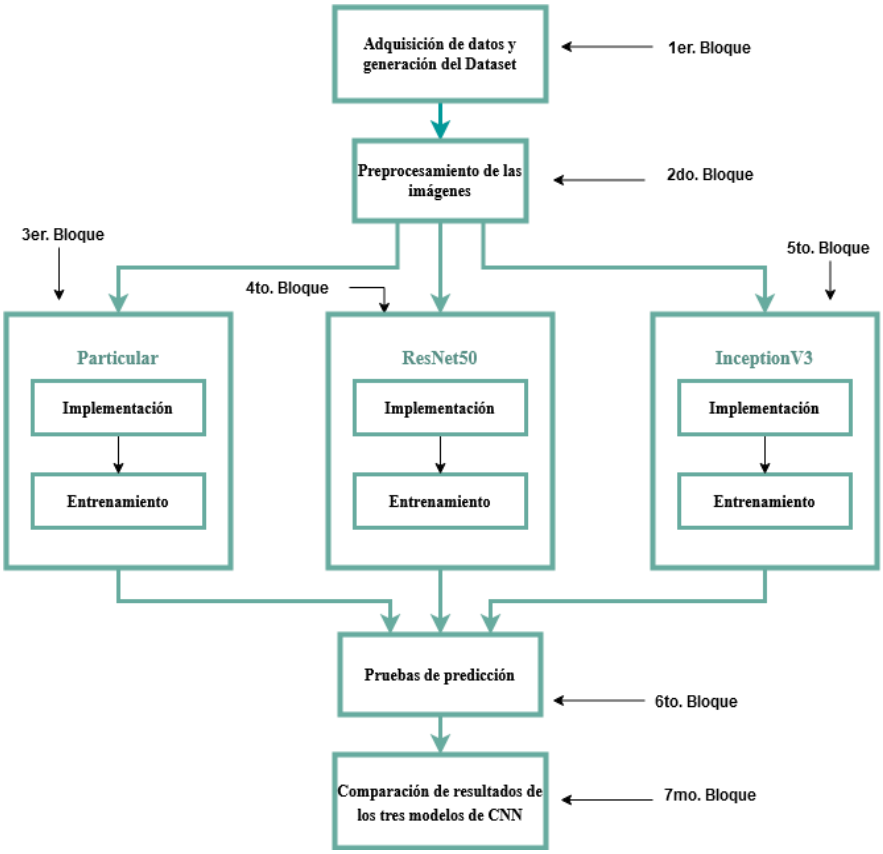

Fig. 1 Diagrama de bloques general del proyecto desarrollado.

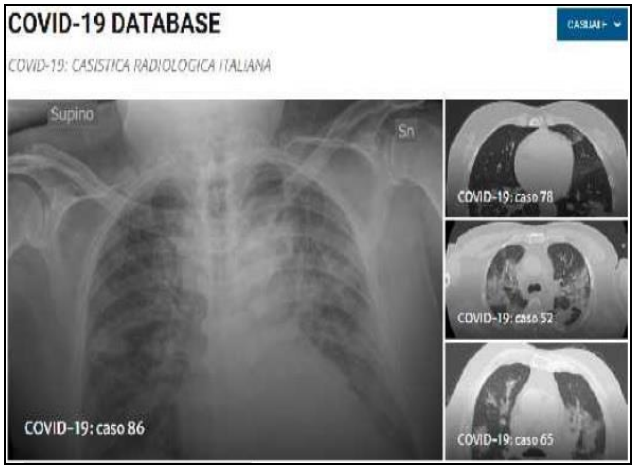

Fig. 2 Imagen de la Base de Datos del SIRM.

Además de ello, se dividió la cantidad de imágenes del conjunto de entrenamiento, en la relación de 80/20, de tal forma que el 20% fue asignado para el conjunto de validación (Val). Esto con el fin de evitar problemas de underfitting, por poseer una cantidad no muy grande de datos, y con la posibilidad que la CNN sea incapaz de generalizar lo aprendido. De esta forma, por cada clase, el conjunto de entrenamiento estuvo constituido por 848 imágenes, el conjunto de test por 264 imágenes y el de validación por 212 imágenes.

```
import pandas as pd
import os
import shutil
```

```
FILE_PATH = "D:/TESIS/COVID-19 Radiography Database/COVID-19.metadata.xlsx"
IMAGES_PATH = "D:/TESIS/COVID-19 Radiography Database/COVID-19"
```

```
df = pd.read_excel(FILE_PATH)
print(df.shape)
```

```
(219, 4)
```

```
df.tail()
```

	FILE NAME	FORMAT	SIZE	URL
214	COVID-19(215)	PNG	1024*1024	https://www.sirm.org/2020/03/28/covid-19-caso-64/
215	COVID-19(216)	PNG	1024*1024	https://www.sirm.org/2020/03/28/covid-19-caso-65/
216	COVID-19(217)	PNG	1024*1024	https://www.sirm.org/2020/03/28/covid-19-caso-65/
217	COVID-19(218)	PNG	1024*1024	https://www.sirm.org/2020/03/28/covid-19-caso-66/
218	COVID-19(219)	PNG	1024*1024	https://www.sirm.org/2020/03/28/covid-19-caso-66/

```
TARGET_DIR = "D:/TESIS/Dataset/COVID-19"
```

```
shutil.copytree(IMAGES_PATH,TARGET_DIR)
```

```
'D:/TESIS/Dataset/COVID-19'
```

Fig. 3 Parte del código de programa en Python para la importación de la base de datos.

B. Preprocesamiento de las imágenes

Esta etapa permitió mejorar el proceso de entrenamiento de los modelos de CNN, dado que un adecuado preprocesamiento de imágenes permite al algoritmo discernir mejor entre las clases definidas. Para ello, se importó la librería TensorFlow, la cual requirió para su uso, la instalación de CUDA 10.1 de NVIDIA y cuDNN 7.5. La versión utilizada de Tensorflow fue la 2.3.1, la cual contiene la API de Keras en la versión 2.4.3. Keras a su vez contiene librerías especializadas para la implementación de los modelos de redes neuronales convolucionales, preprocesamiento de imágenes, entrenamiento, entre otras funciones que fueron útiles en este trabajo. Entre las librerías utilizadas, se tiene "ImageDataGenerator", la cual realizó el preprocesamiento de las imágenes por medio de argumentos que son declarados en su algoritmo. Además, esta librería permitió realizar técnicas de aprendizaje como el de Data Augmentation, la cual fue utilizada en la etapa de entrenamiento para evaluar su utilidad en la mejora de su proceso, a través de dos pruebas (con Data Augmentation y sin Data Augmentation). Esta técnica fue utilizada por [1] y [2] debido a que consideraba que sus bases de datos no eran muy amplias. Asimismo, en la Tabla I, se muestran los principales valores de argumentos declarados en la librería ImageDataGenerator() que fueron utilizados en el proceso de entrenamiento, tanto para los casos con y sin Data Augmentation. A continuación, en la Figura 4, se muestra parte de la programación empleada para el preprocesamiento de las imágenes.

Adicionalmente, se declaró el argumento: "target_size" para redimensionar el tamaño de las imágenes (largo y ancho), siendo 224x244 para los modelos Particular y ResNet50 y 299x299 para el modelo InceptionV3.

TABLA I
VALORES DE LOS ARGUMENTOS DECLARADOS PARA EL PRE PROCESAMIENTO DE LAS IMÁGENES

ARGUMENTOS DECLARADOS	VALORES	
	Con Data Augmentation	Sin Data Augmentation
rescale	1.0 / 255	1.0 / 255
rotation_range	40	--------
width_shift_range	0.2	--------
height_shift_range	0.2	--------
shear_range	0.2	--------
zoom_range	0.2	--------
horizontal_flip	True	--------
fill_mode	nearest	--------

C. Modelo de CNN Particular

La implementación y entrenamiento del modelo particular de CNN se realizó utilizando el API de Keras con la librería Tensorflow en el lenguaje de programación Python. A continuación, se indican las características principales.

```
train_datagen = ImageDataGenerator(rescale=1. / 255)

val_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical')

val_generator = val_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical')
```

```
Found 848 images belonging to 2 classes.
Found 212 images belonging to 2 classes.
```

Fig. 4 Parte del código de programa en Python para el preprocesamiento de las imágenes.

- La implementación de la arquitectura del modelo Particular, se realizó aplicando el método heurístico de prueba y error. Se tomó como base el ejemplo proporcionado por [13] y el realizado por [3], que consistió en aplicar diferentes cantidades de filtros con la finalidad de mejorar la extracción de características de la imagen, además se consideró la evaluación realizada por parte de [2] sobre el uso de la capa Dropout, con la finalidad de reducir la posibilidad de sufrir un sobreajuste. Ver la Figura 5.

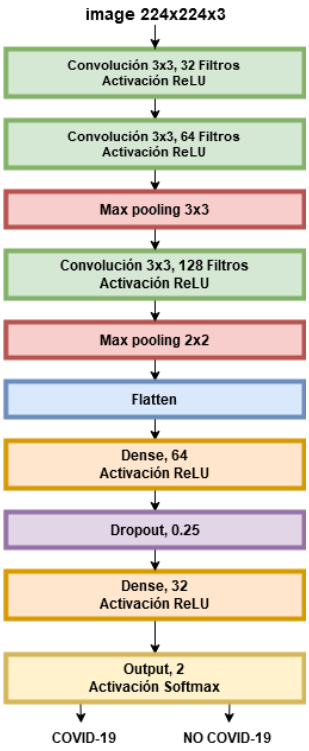

Fig. 5 Arquitectura de la CNN Particular.

- Para el entrenamiento, se definió el valor del hiperparámetro “steps_per_epoch”, el cual representa el número de veces que se procesa la información en cada una de las “epocas”. Su valor se obtuvo dividiendo la cantidad de muestras del conjunto de entrenamiento (848 imágenes) con el parámetro batch_size (64), dando un valor aproximado de 13. Luego, se definió el hiperparámetro “validation_steps”, este se obtuvo dividiendo la cantidad de muestras del conjunto de validación (212 imágenes) con el hiperparámetro batch_size, dando un valor aproximado de 3. Y, finalmente, se ejecutó el entrenamiento con y sin Data Augmentation. El tiempo que tomó el proceso de entrenamiento con Data Augmentation fue de 56 minutos con 38 segundos, mientras que sin Data Augmentation fue de 50 minutos con 10 segundos.

D. Modelo de CNN ResNet50

La implementación y entrenamiento del modelo ResNet50 de CNN, también se realizó utilizando el API de Keras con la librería Tensorflow en el lenguaje de programación Python.

- Este modelo de CNN, tal como se menciona en [6], es un modelo que cuenta con 50 capas, las cuales se encuentran pre-entrenadas, admite una dimensión de entrada máxima de 224x224 pixeles con una profundidad igual a 3 y puede realizar una predicción de hasta 1,000 clases. Igualmente, Keras tiene disponible el algortimo para la implementación del modelo ResNet50 con sus respectivos argumentos, los cuales fueron declarados de forma que sea una red pre-entrenada con la base de datos ImageNet, e importando la librería ResNet50 con keras.applications. Sin embargo, la arquitectura del modelo ResNet50 es demasiada profunda como para entrenar todas sus capas y pesos. Por ello, se aplicó la técnica de aprendizaje llamada Transfer Learning, para realizar el reconocimiento de solo dos clases aprovechando el preentrenamiento de sus pesos con la base de datos ImageNet, y así reducir el esfuerzo computacional por parte de una laptop convencional. Por lo tanto, se agregaron 6 capas al modelo ResNet50 para que pueda ser capaz de discernir entre las clases existentes, y proporcionar una respuesta adecuada al momento de realizar una prueba de predicción. Esta técnica de aprendizaje llamada transfer learning fue utilizada por [4] y [12]. De esta manera, las 6 capas antes mencionadas fueron: una capa Flatten para “aplanar” las dimensiones de los datos recibidos (esta capa posee la misma cantidad de neuronas que la capa anterior a ella), una Capa Dense de 1024 neuronas con activación ReLu, una Capa Dense de 256 neuronas con activación ReLu, una Capa Dense de 32 neuronas con activación ReLu, una Capa Dropout de 0.5 para evitar el overfitting, esto implica desactivar el 50% de la cantidad total de neuronas recibidas, y una Capa con función de activación Softmax, que permite asignar a cada clase un valor probabilístico como salida situada entre 0 y 1. Siendo la salida con mayor valor probabilístico la predicción realizada por el modelo. Esta predicción permitirá conocer si se trata de un caso positivo (COVID-19) o de un caso negativo (NO COVID-19). Ver la Figura 6.

Fig. 6 Arquitectura del modelo ResNet50 luego de aplicar Transfer Learning.

- Para el entrenamiento, se utilizaron los mismos hiperparámetros empleados en el modelo de CNN particular. Asimismo, se ejecutó el entrenamiento con y sin Data Augmentation. Respecto al tiempo de entrenamiento con Data Augmentation fue de 73 minutos con 23 segundos, mientras que sin Data Augmentation el tiempo fue de 70 minutos con 45 segundos.

E. Modelo de CNN InceptionV3

La implementación y entrenamiento del modelo InceptionV3 de CNN, también utilizó el API de Keras con la librería Tensorflow en el lenguaje de programación Python.

- Este modelo de CNN, tal como se menciona en [6], es un modelo que cuenta con 48 capas, las cuales se encuentran pre-entrenadas, admite una dimensión de entrada máxima de 299x299 pixeles con una profundidad igual a 3 y puede realizar una predicción de hasta 1,000 clases. Igualmente, Keras tiene disponible el algortimo para la implementación del modelo InceptionV3 con sus respectivos argumentos, los cuales fueron declarados de forma que sea una red pre-entrenada con la base de datos ImageNet, e importando la librería InceptionV3 con keras.applications. De igual modo, al ser una red demasiada profunda como para

entrenar todas sus capas y pesos; requirió de la aplicación de Transfer Learning. Por lo cual, también se agregaron 6 capas a dicho modelo con el fin de discernir entre las clases existentes sin realizar un excesivo esfuerzo computacional. Las capas agregadas fueron: una capa Flatten para "aplanar" las dimensiones de los datos recibidos con la misma cantidad de neuronas que la capa anterior, una Capa Dense de 128 neuronas con activación ReLu, una Capa Dense de 64 neuronas con activación ReLu, una Capa Dense de 32 neuronas con activación ReLu, una Capa Dropout de 0.5 para evitar el overfitting, esto implica desactivar el 50% de la cantidad total de neuronas recibidas, y una Capa con función activación Softmax que permite que la salida probabilística se encuentre entre 0 y 1. Es así que, la salida con el mayor valor probabilístico será la predicción realizada por el modelo. Además, esta predicción corresponderá a un caso positivo (COVID-19) o a un caso negativo (NO COVID-19). Ver la Figura 7.

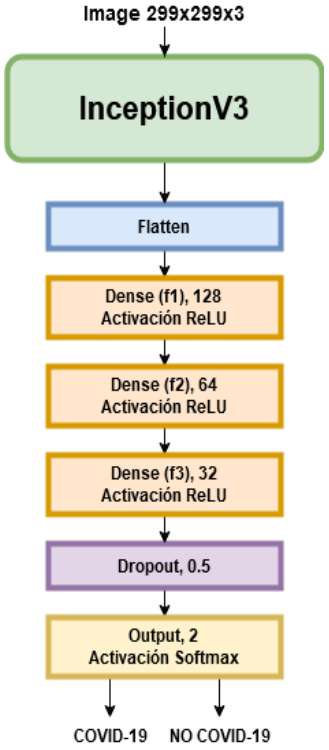

Fig. 7 Arquitectura del modelo InceptionV3 luego de aplicar Transfer Learning.

- Para el entrenamiento, se utilizaron los mismos hiperparámetros empleados en el modelo de CNN particular (epocas=40 y batch_size=64). Y como también, se ejecutó el entrenamiento con y sin Data Augmentation. Respecto al tiempo de entrenamiento con Data Augmentation fue de 67 minutos con 17 segundos, mientras que sin Data Augmentation el tiempo fue de 69 minutos con 50 segundos.

IV. Resultados

En esta sección, se muestan los contenidos del sexto y séptimo bloque representados en la Figura 1. En el sexto bloque se realizaron las pruebas de predicción de los tres modelos de CNN, con Data Augmentation y sin Data Augmentation, Y, en el último bloque, se realizó la comparación de los resultados obtenidos por medio de las matrices de confusión y las métricas de evaluación Accuracy, Precision, Recall y F1-Score, por cada modelo de CNN utilizado. Igualmente, los procesos de entrenamiento se muestran a través de gráficos que utilizan como parámetros de medición la función de optimización "Loss" y la de "Accuracy".

A. Resultados del Modelo de CNN Particular

Los resultados del entrenamiento y validación de este modelo de CNN son mostrados en la Tabla II, donde también se visualiza la comparación entre los valores de las métricas obtenidas cuando fue entrenada con Data Augmentation y sin Data Augmentation.

TABLA II
MÉTRICAS DEL MODELO DE CNN PARTICULAR

CLASES	Precision		Recall		F1-Score	
	Con Data Augmentation	Sin Data Augmentation	Con Data Augmentation	Sin Data Augmentation	Con Data Augmentation	Sin Data Augmentation
COVID-19	0.9921	0.9424	0.9545	0.9924	0.9730	0.9668
NO COVID-19	0.9562	0.9920	0.9924	0.9394	0.9740	0.9650

Asimismo, el proceso de entrenamiento con y sin Data Augmentation tuvo un total de 40 "epocas". De esta manera, en la Figura 8 se presenta la gráfica basada en la métrica de "accuracy" para el caso con Data Augmentation.

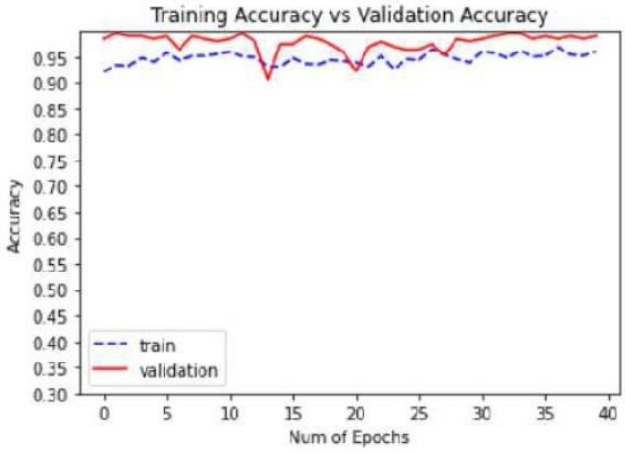

Fig. 8 Gráfica de Accuracy con Data Augmentation, para el modelo de CNN Particular.

B. Resultados del Modelo de CNN ResNet50

Los resultados del entrenamiento y validación de este modelo de CNN son mostrados en la Tabla III, donde también se visualiza la comparación entre los valores de las métricas obtenidas cuando fue entrenada con Data Augmentation y sin Data Augmentation.

TABLA III
MÉTRICAS DEL MODELO DE CNN RESNET50

CLASES	Precision		Recall		F1-Score	
	Con Data Augmentation	Sin Data Augmentation	Con Data Augmentation	Sin Data Augmentation	Con Data Augmentation	Sin Data Augmentation
COVID-19	0.9762	0.9771	0.9318	0.9697	0.9535	0.9734
NO COVID-19	0.9348	0.9699	0.9773	0.9773	0.9556	0.9736

Asimismo, el proceso de entrenamiento con y sin Data Augmentation tuvo un total de 40 "epocas". De esta manera, en la Figura 9 se presenta la gráfica basada en la métrica de "accuracy" para el caso sin Data Augmentation.

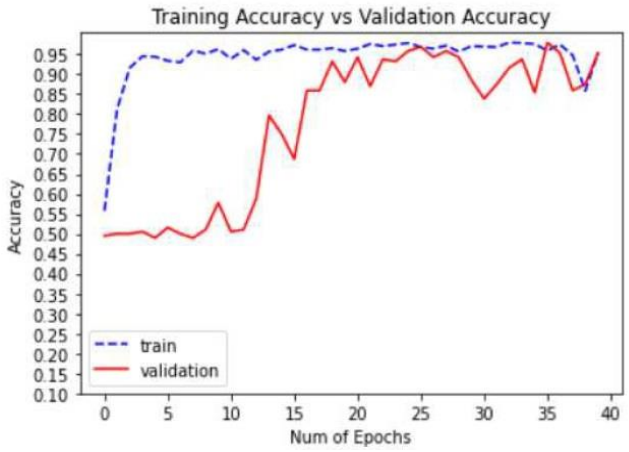

Fig. 9 Gráfica de Accuracy sin Data Augmentation, para el modelo de CNN ResNet50.

C. Resultados del Modelo de CNN InceptionV3

Los resultados del entrenamiento y validación de este modelo de CNN son mostrados en la Tabla IV, donde también se visualiza la comparación entre los valores de las métricas obtenidas cuando fue entrenada con Data Augmentation y sin Data Augmentation.

Asimismo, el proceso de entrenamiento con y sin Data Augmentation tuvo un total de 40 "epocas". De esta manera, en la Figura 10 se presenta la gráfica basada en la métrica de "accuracy" para el caso con Data Augmentation.

D. Comparación de los 3 modelos de CNN

La comparación de los resultados de los 3 modelos de CNN, con y sin Data Augmentation, están basados en las métricas de evaluación y las matrices de confusion (TP: Verdadero Positivo, TN: Verdadero Negativo, FP: Falso Positivo y FN: Falso Negativo). Ver las Tablas V y VI.

TABLA IV
MÉTRICAS DEL MODELO DE CNN INCEPTIONV3

CLASES	Precision		Recall		F1-Score	
	Con Data Augmentation	Sin Data Augmentation	Con Data Augmentation	Sin Data Augmentation	Con Data Augmentation	Sin Data Augmentation
COVID-19	0.9924	0.9776	0.9848	0.9924	0.9886	0.9850
NO COVID-19	0.9850	0.9923	0.9924	0.9773	0.9887	0.9847

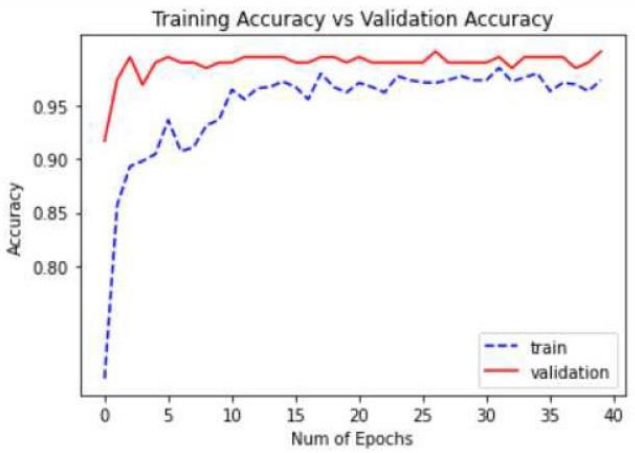

Fig. 10 Gráfica de Accuracy con Data Augmentation, para el modelo de CNN InceptionV3.

TABLA V
COMPARACIÓN DE LOS 3 MODELOS CON DATA AUGMENTATION

Modelos de CNN			Métricas de Evaluación				
			Accu-racy	Speci-ficity	Preci-sion	Recall	F1-score
Particular	TP	126	0.9735	0.9924	0.9742	0.9735	0.9735
	TN	131					
	FP	1					
	FN	6					
ResNet50	TP	123	0.9545	0.9773	0.9555	0.9545	0.9545
	TN	129					
	FP	3					
	FN	9					
Incepction V3	TP	130	0.9886	0.9924	0.9887	0.9886	0.9886
	TN	131					
	FP	1					
	FN	2					

TABLA VI
COMPARACIÓN DE LOS 3 MODELOS SIN DATA AUGMENTATION

Modelos de CNN			Métricas de Evaluación				
			Accu-racy	Speci-ficity	Preci-sion	Recall	F1-score
Particular	TP	131	0.9659	0.9394	0.9672	0.9659	0.9659
	TN	124					
	FP	8					
	FN	1					
ResNet50	TP	128	0.9735	0.9773	0.9735	0.9735	0.9735
	TN	129					
	FP	3					
	FN	4					
Incepction V3	TP	131	0.9848	0.9773	0.9850	0.9848	0.9848
	TN	129					
	FP	3					
	FN	1					

V. CONCLUSIONES

El modelo Particular, según las Tablas V y VI obtuvo un accuracy de 0.9735 con Data Augmentation (DA), y 0.9659 sin DA. Por lo cual, se deduce que la aplicación de DA mejoró el proceso de entrenamiento. Por el contrario, el entrenamiento con DA sufrió de overfitting lo que repercute en el aprendizaje de este modelo de CNN.

Por otra parte, el modelo ResNet50 según las Tablas V y VI, obtuvo una accuracy de 0.9545 aplicando DA y 0.9735 sin DA; por lo cual, este modelo no requiere de DA para mejorar su proceso de entrenamiento. Sin embargo, el entrenamiento sin DA sufrió de overfitting tal como se observó. De igual modo, en las pruebas de predicción tuvo menor acierto que el entrenamiento con DA.

De la misma forma, el modelo InceptionV3 según las Tablas V y VI, obtuvo un accuracy de 0.9886 aplicando DA y 0.9848 sin DA; por consiguiente, el modelo InceptionV3 requiere de DA para poder tener mejor desempeño. Además, en el entrenamiento de este modelo sin DA se observó presencia de overfitting.

Entonces, teniendo en cuenta los resultados que se muestran en las Tablas V y VI, se concluye que el mejor modelo del grupo de tres fue la CNN InceptionV3, tanto para el entrenamiento con DA como sin DA. Por lo contrario, el modelo con peor desempeño luego de aplicar DA fue la CNN ResNet50, mientras que sin aplicar DA fue la CNN Particular. Por ello, se determina que el modelo más apto para realizar la clasificación adecuada de radiografías de tórax es el modelo de CNN InceptionV3.

Finalmente, los resultados obtenidos muestran que el uso de redes neuronales convolucionales tiene una utilidad muy relevante como apoyo en el área médica, y particularmente en el diagnóstico de enfermedades. Por ello, para trabajos futuros se recomienda mejorar los parámetros de precisión de los modelos evaluados, así como la ampliación de la base de datos a utilizar. Del mismo modo, se plantea el desarrollo de nuevas arquitecturas de CNN enfocadas en el ámbito del diagnóstico médico.

REFERENCIAS

[1] A. Medrano Roldán, “Red neuronal convolucional en un ambiente pseudo-distribuido para la clasificación de radiografías de pacientes con neumonía.” Reporte Técnico, Universidad Autónoma de Ciudad Juarez, Juarez, México, 2019.

[2] A. Gómez-Ríos, S. Tabik, J. Luengo, & F. Herrera, “Redes Neuronales Convolucionales para una Clasificacion Precisa de Imágenes de Corales,” in XVIII Conferencia de la Asociación Española para la Inteligencia Artificial, 2019, pp. 1171-1176.

[3] P. Costa. (2019). Transfer Learning ¿qué es y para que sirve? Recuperado el 23 de septiembre de 2020 [Online]. Available: https://pochocosta.com/podcast/transfer-learning-que-es-y-para-que-sirve/

[4] SIRM. (2020). COVID-19 DATABASE. Recuperado el 27 de octubre del 2020 [Online]. Available: https://www.sirm.org/category/senza-categoria/covid-19/

[5] P. Patel, (2020). Chest X-ray (Covid-19 & Pneumonia). Recuperado el 27 de octubre de 2020 [Online]. Available: https://www.kaggle.com/prashant268/chest-xray-covid19-pneumonia

[6] J. Cohen, (2020). COVID-19 Image Data Collection: Prospective Predictions Are the Future. Recuperado el 21 de Setiembre de 2020, [Online]. Available: https://github.com/ieee8023/covid-chestxray-dataset

[7] P. Mooney, (2018). Chest X-Ray Images (Pneumonia). Recuperado el 15 de Setiembre de 2020, [Online]. Available: https://www.kaggle.com/paultimothymooney/chest-xraypneumonia

[8] R. Sethi, M. Mehrotra, & D. Sethi, “Deep Learning based Diagnosis Recommendation for COVID-19 using Chest X-Rays Images”, in Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, pp. 1-4. July 2020.

[9] T. Ozturk, M. Talo, E. Yildirim, U. Baloglu, O. Yildirim, & U. Acharya, “Automated detection of COVID-19 cases using deep neural networks with X-ray images,” Computers in Biology and Medicine, vol. 121, no. 103792, pp. 1-11, June 2020.

[10] A. Narin, C. Kaya & Z. Pamuk. (March, 2020). Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks [Online]. Available: https://arxiv.org/ftp/arxiv/papers/2003/2003.10849.pdf

[11] A. Artola, “Clasificación de imágenes usando redes neuronales convolucionales en Python,” Trabajo Fin de Grado, Universidad de Sevilla, Sevilla, 2019.

[12] H. Sharma, J. Jain, P. Bansal, & S. Gupta, “Feature Extraction and Classification of Chest X-Ray Images Using CNN to Detect Pneumonia”, in 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, pp. 227-231, January 2020.

[13] S. Colula, & L. Ángel, “Reconocimiento de patrones en imágenes médicas por medio de redes neuronales convolucionales,” Master's Thesis, Universidad Autónoma de Puebla, Puebla, México, 2019.