DOI: https://doi.org/10.56712/latam.v4i1.296

Reconocimiento de la presencia de sars-cov-2 en pulmones a través de imágenes de radiodiagnóstico haciendo uso de Machine Learning con Python

Recognition of the Presence of Sars-Cov-2 in Lungs from Radiodiagnostic Images Using Machine Learning with Python

Bryan Darwin Luna Bravo
bryanlunabravo@gmail.com
https://orcid.org/0000-0002-8740-3384
Carrera de Física, Facultad de Ciencias, Escuela Superior Politécnica del Chimborazo ESPOCH
Centro de Capacitación, Desarrollo y Transferencia de Ciencia, Educación y Tecnología SCIEDTEC
Santo Domingo – Ecuador

Luis Emilio Carranza Quispe
luisemilio36@gmail.com
https://orcid.org/0000-0002-1891-2986
Facultad de Ciencias de la Salud, Universidad Técnica Estatal de Quevedo.
Centro de Capacitación, Desarrollo y Transferencia de Ciencia, Educación y Tecnología SCIEDTEC
Quito - Quevedo – Ecuador

Artículo recibido: día 16 de diciembre de 2022. Aceptado para publicación: 19 de enero de 2023.
Conflictos de Interés: Ninguno que declarar.

Resumen

El objetivo de este trabajo utilizar Machine Learning (ML), para reconocimiento de SARS-CoV-2, mediante imágenes médicas adquiridas por tomografía computarizada de la región del tórax en formato DICOM, a partir de un tomógrafo Siemens somatom de 2 cortes y un data set en la nube, que posteriormente fueron transformadas a imágenes “png”. El sistema de reconocimiento fue construido mediante el lenguaje de programación “Python”, haciendo uso de librerías de código abierto, tanto como para Machine Learning siendo esta “TensorFlow”, para el manejo de archivos DICOM se hizo uso de “Pydicom” y para imágenes “Open CV”. Las imágenes se importaron a una red neuronal convolucional pre entrenada adaptándola al tipo de clasificación multiclase del proyecto, aplicando técnicas de aumento de datos (Data Augmentation), decaimientos exponenciales de parámetros de la red neuronal como el Learning Rate, entrenando la red neuronal convolucional, optimizando los parámetros adecuados para su correcto funcionamiento de reconocimiento, posteriormente se desarrolló una interfaz web mediante la librería “Streamlit” para el manejo y la aplicabilidad del modelo siendo de uso dinámico para el usuario siendo multiplataforma. Se obtuvieron resultados cuantitativos que permitieron reflejar la eficacia del modelo con una eficacia del 88% para detectar COVID-19. Se recomienda la instalación previa de librerías de Python para el correcto funcionamiento del sistema de reconocimiento.

Palabras clave: imágenes médicas, tomografía computarizada, covid 19, machine learning, red neuronal artificial (rna)

Abstract

The goal of this work was to use Machine Learning (ML), for SARSCoV-2 recognition, using medical images acquired by computed tomography of the chest region in DICOM format, from a 2-slice Siemens somatom tomograph and a data set in the cloud, which were subsequently transformed to "png" images. The recognition system was built using the "Python" programming language, making use of open-source libraries, both for Machine Learning and "TensorFlow"; "Pydicom" was used to manage DICOM files and "Open CV" for images. The images were imported into a pre-trained convolutional neural network, adapting it to the type of multi-class classification of the project, applying data augmentation techniques (Data Augmentation), exponential decays of neural network parameters such as the Learning Rate, training the convolutional neural network, optimising the appropriate parameters for its correct recognition operation, subsequently a web interface was developed using the "Streamlit" library for the management and applicability of the model being of dynamic use for the user and being multiplatform. Quantitative results were obtained that reflected the effectiveness of the model with an efficiency of 88% for detecting COVID-19. The prior installation of Python libraries is recommended for the correct functioning of the recognition system.

Keywords: medical images, computed tomography, covid 19, machine learning, artificial neural network (ann)

Como citar: Luna Bravo, B. D., & Carranza Quispe, L. E. (2023). Reconocimiento de la presencia de sars-cov-2 en pulmones a través de imágenes de radiodiagnóstico haciendo uso de Machine Learning con Python. *LATAM Revista Latinoamericana de Ciencias Sociales y Humanidades 4(1), 788-806. https://doi.org/10.56712/latam.v4i1.296*

INTRODUCCIÓN

El Covid 19, es una enfermedad que ha azotado a la población mundial en estos últimos años, y la demanda de sistemas reconozcan la enfermedad se ha aumentado donde la existencia de métodos para la identificación de Covid 19 como el PCR, Rx y tomografías han sido de gran ayuda, sin embargo, siempre detrás de estas debe estar un profesional de la salud que este especializado en el diagnóstico de las mismas. Partiendo de las tomografías computarizadas como medio de reconocimiento se ha planteado desarrollar un sistema que ayude a identificar patologías de una manera autónoma en las imágenes, tomando en consideración nuevas tecnologías computacionales que aportan a la medicina medios de diagnóstico automatizados, esa aquí donde la inteligencia artificial se hace presente mediante técnicas de aprendizaje automático o también llamado Machine Learning, esto gracias a grandes cantidades de datos de imágenes previamente diagnosticadas por personal capacitado como médicos radiólogos, para modelar sistemas de redes neuronales convolucionales que permite el reconocimiento de patrones en las imágenes que poseen patologías, ayudando así a un diagnóstico más rápido, además siendo de gran ayuda en lugares donde no se cuenta con la presencia de un personal médico capacitado.

Este proyecto desarrollará un sistema de reconocimiento de imágenes médicas, para evaluar la presencia de SARS-CoV-2 en pulmones mediante imágenes tomográficas, aplicando Machine Learnig usando Python como lenguaje de programación y librerías que permitan llevar acabo algoritmos de ML, entre las cuales esta Tensor Flow, obteniendo como resultado una aplicación web que contenga un sistema de reconocimiento imágenes médicas.

MÉTODOLOGÍA

En el presente trabajo se utiliza Python como lenguaje de programación principal haciendo uso de sus librerías o frameworks, tanto para el análisis de datos, manejo de matrices y arrays, de imágenes médicas en formato ".dcm" pasando por la implementación de redes neuronales convolucionales hasta la creación de una interfaz amigable para el usuario.(Serna & Trujillo, 2010)

Adquisición de Imágenes o Archivos de Tomografía computarizada

Los datos de tomografías computarizadas son la materia prima para la implementación de la Red neuronal convolucional (CNN por sus siglas en el inglés),archivos de imágenes médicas que se obtendrán de una institución médica, estos archivos son datos con su respectivo tratamiento de imagen realizados en el tomógrafo de marca Siemens y modelo Somaton Spirit de 2 cortes que exporta los datos con las diferentes etiquetas como: nombre del paciente, ID, tipo de estudio, entre las diferentes etiquetas propias del formato Dicom (.dcm), al servidor de datos médicos de la clínica conocido como PACS por sus siglas en inglés "Picture Archiving and Communication System", una vez que los datos alojados en el servidor, son manipulados mediante el visualizador de archivos dicom "Horos Viewer", se procede a la clasificación de estudios tomográficos por paciente y por el tipo de estudio. Para este trabajo, el tipo de estudio o la descripción del estudio requerido será "Tórax Simple" por consiguiente tenemos que exportar los estudios requeridos de dos formas, una copiando los archivos exportados a una memoria extraíble o permitiendo la conexión remota al servidor PACS por medio de "Horos Viewer" y "RadiiAnt" (Serna & Trujillo, 2010)

Existen dos maneras para adquirir los datos del sistema PACS

Memoria o disco extraíble

Para este proceso es necesaria una memoria o un disco lo suficientemente grande para alojar la cantidad de archivos exportados, conectando directamente el extraíble al servidor copiando los archivos.

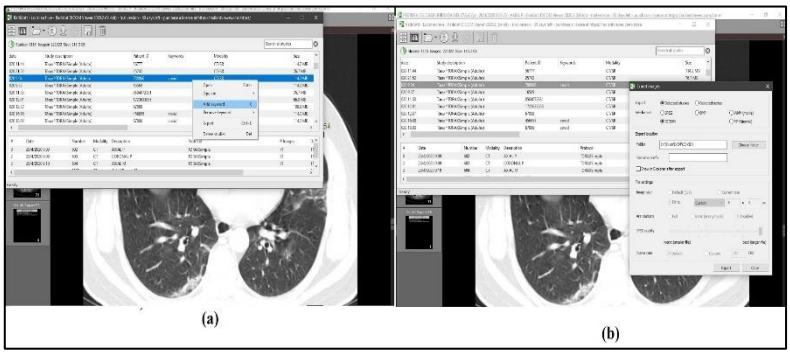
Conexión Remota y descarga

Siendo el PACS un servidor que tiene la virtud de conceder una conexión remota al mismo, podemos valernos de un visualizador de imágenes. dcm llamado "RadiAnt" configurando la ip pública y el puerto del PACS, con un protocolo CGET de los archivos en cuestión y procediendo a respectiva descarga.

Descartar Estudios

Si bien se extrae todos los estudios de tórax simple, es importante determinar cuál de estos archivos están asociado a pacientes que poseen Covid 19, para esto procedemos a cargar los archivos y visualizarlos con el software "RadiiAnt". Etiquetamos con una keyword para identificar los estudios de pacientes que presentan Covid 19. Una vez finalizada la clasificación de los estudios tomográficos exportamos los estudios atribuidos a los pacientes que padecen de Covid 19.

Figura 1. *Colocación de etiqueta del estudio. b) Exportación de estudios*



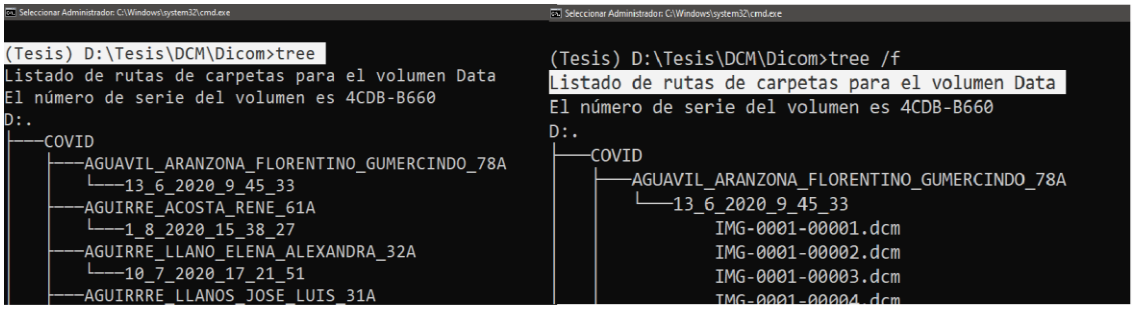
Fuente: Radiant Dicom 2021.

Cargar Datos

Se usa el entorno de desarrollo integrado (IDE) Spyder debido a sus características, ya que cuenta con ventanas que permiten saber con qué tipo de datos estamos trabajando, gracias a su panel de exploración de variables, además de que posee una consola para el intérprete de Python.

Cargar la data es el primer paso, para esto importamos el directorio o path en la ruta "…/COVID" del Pc que contiene todos los archivos DCM de los pacientes diagnosticados con Covid19, sin embargo este directorio está organizado por nombres y apellidos, luego con subcarpetas que le corresponden a fecha y dentro de estas los archivos ".dcm" (Jung, 2021).

Figura 2. *Listado de carpetas. Listado de archivos de imágenes médicas*

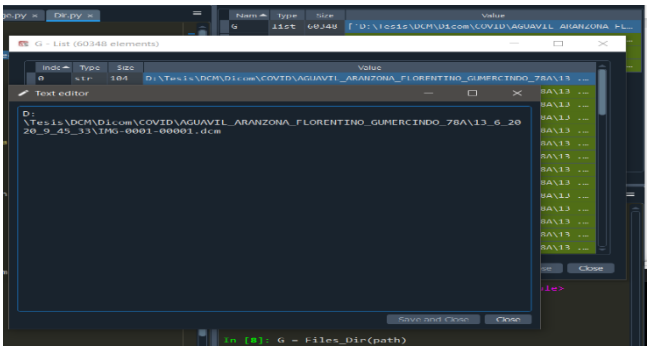


Fuente: Luna, Bryan 2021.

Para leer los archivos en las carpetas del sistema haremos uso del módulo os, y propone una función que guarda como un módulo en un archivo "Dir.py" que nos ayude a encontrar las rutas

de cada uno de los archivos DICOM. Se importa las funciones Files_Dir y Fol_Dir, sin embargo, en concreto, solo se necesita las listas de rutas de los archivos DICOM y luego ser manipuladas. Para esto se lo guarda en una lista (Mason, 2011).

Figura 3. *Resultado o salida de rutas de la función Files_Dir*

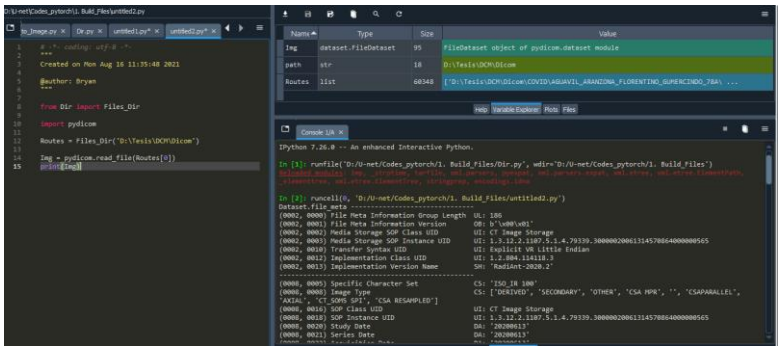


Fuente: Spyder 2021.

Análisis de Datos

Una vez ya exportados los datos se dará un repaso por cada uno de los metadatos que conlleva un archivo ". dcm", tomando en cuenta los más importantes para luego extraer la matriz de píxeles que ayudan a normalizar y crear una imagen ".png" (Mason, 2011).

Figura 4. *Lectura del archivo DICOM*



Fuente: Spyder 2021.

Normalización de Arrays de los Pixeles de imágenes médicas

La array de píxeles de un archivo DICOM está en valores de 16 bits, esta array resultante tiene valores muy elevados para que puedan ser visualizados de una forma correcta como imágenes jpg o png, es decir se debe hacer un proceso de normalización, primero transformando la array en valores de la escala Hounsfield mediante una transformación lineal de unidades (LUT, por sus siglas en inglés Lineal Transform Units), donde se obtendrá una escala de entre -1024 a 3071.

Es importante tener en cuenta los valores de grises inferiores y superiores, para esto se debe tomar a consideración el window width y el window center donde el valor inferior seran negros y los superiores serán blancos, para este cálculo se considera la siguiente formula (Sande & Ramdurg, 2020):**Ecuación 1**

Window max

$$Window_{max} = Window_{center} + \frac{Window_{width}}{2}$$

Fuente: Cai et al., 2017

Ecuación 2

Window min

$$Window_{min} = Window_{center} \frac{Window_{width}}{2}$$

Fuente: Cai et al., 2017

Ecuación 3

Normalización

$$y = X * m + b$$

Fuente: Cai et al., 2017

Donde m es la pendiente o rescale Slope, b es la intersección o recale interception y x hace referencia a los valores de los pixeles en la array (Cai et al., 2017).

Ecuación 4

Normalización escala Hounsfield

$$Rescale_{Pix\,val} = Pix_{Val} * Rescale_{Slope} + Rescale_{Intercept}$$

Fuente: Cai et al., 2017

Sin embargo, para que la computadora pueda leer los archivos, luego de transformarlos al formato ".png", se debe convertir la matriz de pixeles de 16 bits a una matriz de 8 bits donde el valor mínimo será de 0 y el valor máximo será 255 y para los valores intermedios entre 0 y 255 haremos el siguiente cálculo cada uno de los pixeles (Cai et al., 2017).

Ecuación 5

Normalización Png

$$Val_{newpix} = \frac{Rescale_{pix\,val} - Window_{min}}{Window_{max} - Window_{min}} * 255$$

Fuente: Cai et al., 2017

Para todo el proceso de normalización se crea un archivo "Dicom_to_Image.py" donde se define la función Dicom2array.

Conversión de Array a imagen ".png"

La transformación de las arrays de 16 bits a 8 bits permite crear archivos de imagen, pues se sabe que una imagen no es más que una matriz, y en contraste con las imágenes RGB las arrays generadas solo posee un valor de profundidad lo que hace que se caractericen, es decir, si son de 8 o 16 bits, dado que las arrays generadas poseen valores entre los 255 y 0, así, de esta manera poseen 255 tonos en escala de grises en un solo canal. (Oliphant, 2006).

Para la transformación de un array de pixeles del archivo DICOM hasta la creación de imágenes de 8 bits en el pc, se crea un algoritmo, que hace uso de las funciones ya antes definidas e importándolas, además del módulo cv2 de open-cv que es una librería para transformar las arrays de 8 bits procesadas a imágenes .png y salvarlas en el ordenador (Viera Maza, 2017).

Tabla 1. *Número de Archivos de Imagen*

Clases:	Data Set propio
Covid_19	18026
Normal	9739
Pneumonia	0
Total	27765

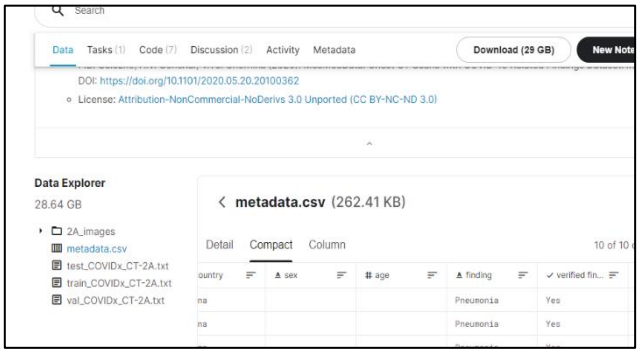
Elaborado por: Luna, Bryan 2021.

Adición de imágenes médicas de Kaggle

A pesar de la gran cantidad de imágenes convertidas, no en todas se puede apreciar debidamente con signos de Covid 19, así que se procede a eliminar las que no aportan al proyecto. Sin embargo, existen más fuentes de imágenes, se adiciona más imágenes de un repositorio de Kaggle. llamado COVIDx CT -2 que contiene aproximadamente 194922 imágenes médicas o CT de pacientes.

Este data set posee una carpeta que contiene la totalidad de las imágenes con su respectiva etiqueta en el nombre, sin embargo dentro del mismo se tiene un archivo .csv que almacena la metadata de cada uno de los pacientes y las imágenes correspondientes, así como 3 archivos de texto plano .txt los cuales corresponden a imágenes de test, entrenamiento y validación, pero para este proyecto se hace caso omiso de estos 3 últimos archivos utilizando únicamente el archivo metadata.csv que engloba los metadatos de cada uno de los pacientes etiquetados y ubicándolos en cada una de sus carpetas, para esto se crean 3 carpetas denominadas: 'Covid_19' , 'Normal' y 'Pneumonia', que corresponden a las clases, que deben ser reconocidas por el modelo (Usmani, 2016).

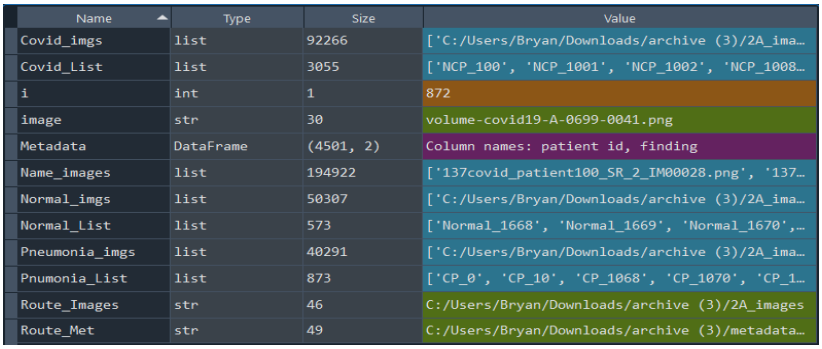
Figura 5. *Dataset COVIDx CT -2 en Kaggle*



Fuente: Kaggle, 2021, párr.3.

Ya con las rutas en las cuales se deja el data set distribuido de acuerdo con el archivo csv que contiene la metadata, se procede a realizar un código en Python que ayude a mover los archivos a sus carpetas correspondientes según la metadata.csv.

Figura 6. *Visualización de Imágenes a partir del archivo "csv"*



Name	Type	Size	Value
Covid_imgs	list	92266	['C:/Users/Bryan/Downloads/archive (3)/2A_ima…
Covid_List	list	3055	['NCP_100', 'NCP_1001', 'NCP_1002', 'NCP_1008…
i	int	1	872
image	str	30	volume-covid19-A-0699-0041.png
Metadata	DataFrame	(4501, 2)	Column names: patient id, finding
Name_images	list	194922	['137covid_patient100_SR_2_IM00028.png', '137…
Normal_imgs	list	50307	['C:/Users/Bryan/Downloads/archive (3)/2A_ima…
Normal_List	list	573	['Normal_1668', 'Normal_1669', 'Normal_1670',…
Pneumonia_imgs	list	40291	['C:/Users/Bryan/Downloads/archive (3)/2A_ima…
Pnumonia_List	list	873	['CP_0', 'CP_10', 'CP_1068', 'CP_1070', 'CP_1…
Route_Images	str	46	C:/Users/Bryan/Downloads/archive (3)/2A_images
Route_Met	str	49	C:/Users/Bryan/Downloads/archive (3)/metadata…

Fuente: Luna, Bryan 2021.

El archivo metadata.csv contiene una lista de todas los estudios de los pacientes con su respectiva ID, así que relacionara el ID según la condición "finding" que contempla el estado del paciente según sea Covid 19, Normal o Pneumonia.(Varma, 2012)

En total se obtuvieron 182864 imágenes de data set COVIDx CT-2 el resto de imágenes no se pudieron obtener debido a que el archivo metadata.csv no estaba debidamente etiquetado.

Tabla 2. *Número de imágenes de Kaggle*

Datos	Nº de archivos
Covid_19	92266
Normal	50307
Pneumonia	40291
Total	182864

Elaborado por: Luna, Bryan 2021.

Adición de Archivos

Este paso se fundamenta únicamente en agregar el data set creado anteriormente por medio de archivos DICOM al nuevo data set de Kaggle incrementando así el número de archivos png necesarios para el posterior entrenamiento de la red neuronal de clasificación teniendo como objetivo predecir 3 tipos de clases: Covid_19, Neumonías e imágenes normales, de esta adición de datos se obtendrán como resultado:

Tabla 3. *Número de imágenes de Kaggle más data set propio*

Clases:	**Data Set propio**	**COVIDx CT -2**	**Nº de archivos Totales**
Covid_19	18026	92266	110292
Normal	9739	50307	60046
Pneumonia	0	40291	40291
Total	27765	182864	210629

Elaborado por: Luna, Bryan 2021.

Datos para pruebas

El data set de pruebas tiene la finalidad de alojar imágenes que no serán utilizadas ni en el entrenamiento ni en la validación, estos archivos tienen como finalidad hacer uso de los modelos obtenidos posteriormente al entrenamiento, así como también la evaluación de los modelos mediante las matrices de confusión. Cabe recalcar que este conjunto de datos de prueba está posterior a la Data Augmentation , puesto que se necesitan imágenes reales obtenidas de manera aleatoria sin modificaciones para el testeo del modelo.

Data augmentation

Para homogenizar la cantidad de archivos en cada una de las carpetas se utilizó la técnica de aumento de datos así se otorga mayor cantidad de datos de entrenamiento a la data set, sobre todo para afrontar la disparidad entre las carpetas: Covid_19 , Normal y Pneunomia esto consiste en tomar cada una de las imágenes y aplicar transformaciones a las mismas para obtener una nueva matriz de imagen, a pesar de que para el ojo humano se percibe como la misma imagen a pesar de las transformaciones como rotaciones, zoom e inversiones horizontales y verticales, esto implicará que la matriz tiene nuevos valores, por ende tenemos una nueva matriz de pixeles que nos será muy útil para el entrenamiento.

Tabla 4. *Número de imágenes destinadas para Entrenamiento y validación*

Datos Entrenamiento y Validacion	Nº de Archivos
Covid _19	100000
Normal	100000
Pneumonia	100000
Total	300000

Elaborado por: Luna, Bryan 2021

Sin embargo, con el fin de homogenizar los datos y números de archivos. “png” para la futura creación de lotes únicamente aplicaremos esta técnica a las carpetas: Normales y Neumonía con el fin de tener un data set de 100.000 imágenes para Covid_19, 100000 imágenes en Normal y 100000 en Neumonía.

Figura 7. *Carpetas para entrenamiento y validación: Covid_19, Normal y Pneumonia*

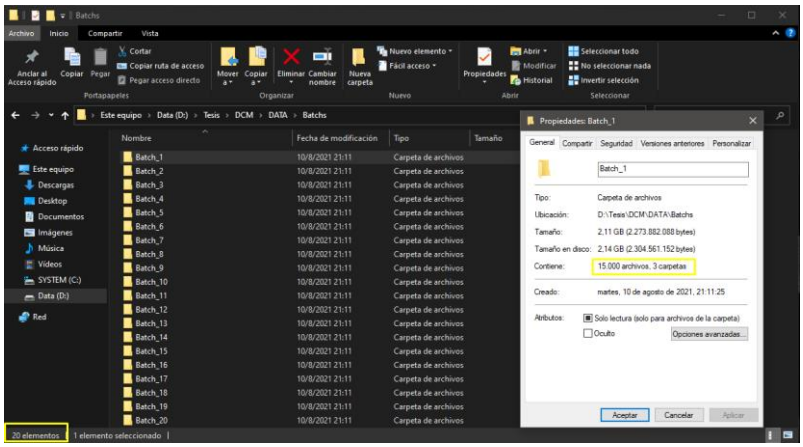
Fuente: Luna, Bryan 2021.

Creación de super batchs o lotes de imágenes para entrenamiento

Se ha obtenido un data set de aproximadamente 300000 imágenes, esto implicará de la demanda de recursos de hardware computacional muy grande, sin embargo, podemos realizar un

entrenamiento por partes, que se denominan en este proyecto como lotes o super batch, cabe recalcar que el terminó batch es utilizado también para establecer el valor de batch size en el algoritmo de entrenamiento de la red neuronal. Los lotes de imágenes estarán distribuidos por 15,000 imágenes por lote, distribuidas en 3 carpetas haciendo un total de 20 lotes para el entrenamiento de la red.

Figura 8. *Carpetas con 15000 imágenes por batchs distribuidos en 3 carpetas*



Fuente: Luna Bryan, 2021

Modelo red neuronal TF funcional

En primera instancia se procederá a la creación de una red funcional con tensorflow que tenga convoluciones 2D capas, capas de Max Polling y drop outs pero con el fin de optimizar el entrenamiento se procese a hacer tranfer Learning agregando únicamente una capa densa con una salida de 3 neuronas que corresponden a las clases a reconocer (Hope et al., 2017).

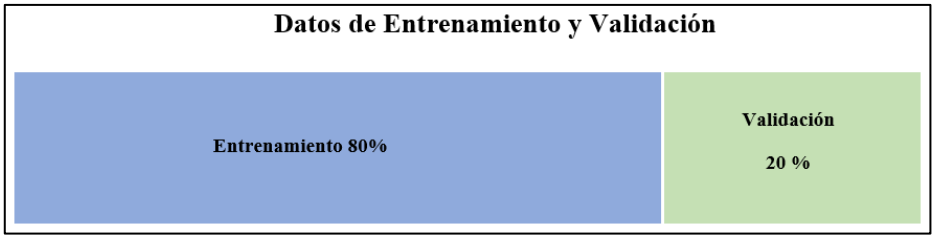
Data set de entrenamiento

Para tomar los datos de entrenamiento se toma cada una de las carpetas de los batchs y llamando a la función de tensorflow "tf.keras.preprocessing.image_dataset_from_directory()" que se encargará de tomar pequeños lotes de datos determinados por un valor respectivo de batch size, de las 15,000 imágenes correspondiente a cada uno de los lotes de datos. Es importante tener en cuenta que el argumento subset en "training" se usa para especificar que train_ds se refiere al conjunto que datos exclusivamente para entrenamiento (V. Subramanian, 2018).

Validation Split

Hace referencia al porcentaje de datos que se van a ocupar para el entrenamiento para este caso lo pondremos en 0,2 lo que implica que se ocuparán el 80 % de datos para entrenamiento y el 20 % será para validación (S. Subramanian et al., 2020).

Figura 9. *Split de datos para entrenamiento y Validación*

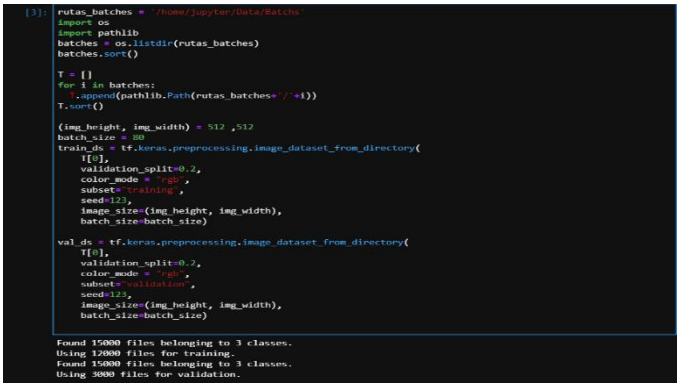


Fuente: Luna Bryan, 2021

Data set de validación

De la misma manera que se toman los datos de entrenamiento mediante esta función Tf, "tf.keras.preprocessing.image_dataset_from_directory()", se toman los datos de validación especificando en la clase "validation" en el argumento subset, así como también el mismo valor de batch_size y el "validation Split" en 0.2 que hace referencia que el 20 % de archivos será para la validación.

Figura 10. *Datos de entrenamiento y Validación por batches*



```
[3]: rutas_batches = '/home/jupyter/Data/Batchs'
     import os
     import pathlib
     batches = os.listdir(rutas_batches)
     batches.sort()

     T = []
     for i in batches:
       T.append(pathlib.Path(rutas_batches+'/'+i))
     T.sort()

     (img_height, img_width) = 512 ,512
     batch_size = 80
     train_ds = tf.keras.preprocessing.image_dataset_from_directory(
         T[0],
         validation_split=0.2,
         color_mode = 'rgb' ,
         subset='training',
         seed=123,
         image_size=(img_height, img_width),
         batch_size=batch_size)

     val_ds = tf.keras.preprocessing.image_dataset_from_directory(
         T[0],
         validation_split=0.2,
         color_mode = 'rgb' ,
         subset="validation",
         seed=123,
         image_size=(img_height, img_width),
         batch_size=batch_size)

     Found 15000 files belonging to 3 classes.
     Using 12000 files for training.
     Found 15000 files belonging to 3 classes.
     Using 3000 files for validation.
```

Fuente: Luna Bryan, 2021.

Entrenamiento en la nube

Debido a la ineficiencia del hardware local se procede a tomar otras medidas, como el en entrenamiento en la nube haciendo uso de las instancias virtuales en los servicios de Google cloud. (Thakurratan, 2018)

Creación cuenta Google cloud

Para esto el primer paso es crear una cuenta, habilitar los servicios de Compute Engine y crear una máquina o instancia virtual con las especificaciones de hardware necesario para la tarea y proceder a escribir el algoritmo de entrenamiento (Thakurratan, 2018)

Crear bucket para almacenamiento de datos en Google Storage

Otro paso importante es habilitar los servicios de Google storage para subir los datos en este caso las imágenes médicas que se utilizarán para el entrenamiento (Thakurratan, 2018).

Creación de instancia para ejecutar el entrenamiento

Para la creación de la instancias o máquina virtual necesaria, se necesita haber habilitado el servicio de Vertex Ai e Ai Platform que nos proporcionaran los recursos necesarios para la instancia y crear un notebook de jupyter para escribir el código a ejecutar.

Tabla 5. *Características de Hardware para entrenamiento*

Hardware	Especificaciones
Sistema Operativo	Linux Debian
Memoria ram	60GB
CPU	16 núcleos
GPU	Nvidia Tesla T4
Disco	400 Gb

Elaborado por: Luna, Bryan 2021.

Transfer Learning

Para aumentar la eficiencia de la red neuronal se usa una red neuronal pre entrenada con sus respectivos pesos, para esto se ocupó EfficientnetB7, esta red neuronal convolucional fue previamente entrenada con un data set llamado ImageNet, este cuenta con aproximadamente 14 millones de imágenes que están contempladas en 1000 categorías, una vez importado el modelo se congelaron las aquellas capas que tienen la facultad de entrenarse para no volver a ser entrenadas, sino más bien mantener los pesos que están optimizados, además se agregó una capa densa al fina añadiendo las 3 clases que se requieren para este proyecto.(Stevens et al., 2020; V.

Compilador

Cuando se compila el modelo se definen las funciones de perdida, optimizador y las métricas para evaluar la red neuronal a medida que se realiza el entrenamiento. (V. Subramanian, 2018)

Decaimiento exponencial

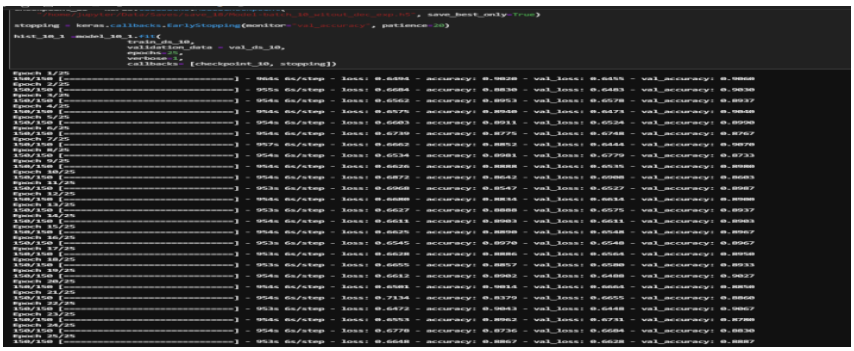
Reducir la tasa de aprendizaje cuando se entrena un modelo resulta útil, esto ayuda aumentar la velocidad del entrenamiento haciendo que la búsqueda del mínimo local sea eficiente y rápida, para optimizar la función de perdida que será definida dentro del compilador, así como también el optimizador.

Algoritmo de entrenamiento

El algoritmo de entrenamiento se compone únicamente de los datos que será usados para el entrenamiento, validación, el batchzise por lotes de entrenamiento que para este caso será de 80 y los callbacks que contiene las configuraciones de guardado del modelo entrenado en caso de que exista una falla además de un early stopping en caso de que el entrenamiento no mejore, esto último en función del accuracy de los datos de validación. Para esto se utiliza la función “fit” del modelo introduciendo como argumento el dataset ya sea de entrenamiento, validación, en número de épocas (epochs) y callbacks (Geewax, 2018).

Figura 11

Progreso de entrenamiento de la red neuronal de un lote

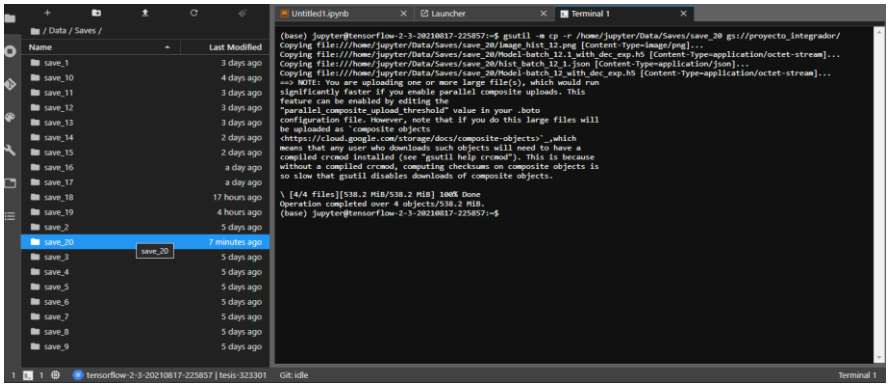


Fuente: Luna Bryan, 2021.

Salvado de modelos entrenamos mediante gsutil

Si bien con la función model.save() se procede a guardar el modelo dentro de la misma instancia para evitar errores es preferible llevarlo al bucket creado en Google Storage y posteriormente descargarlo al pc local, para esto al igual que al momento de subir los datos hacemos uso del comando “gsutil” desde la terminal de Linux de la instancia indicando al dirección del bucket donde queremos alojar la carpeta y los modelos entrenados (Thakurratan, 2018).

Figura 12. *Copia de modelos entrenados al bucket de Google storage*



Fuente: Luna Bryan, 2021.

Stramlit y creación de interfaz gráfica web

Para la presentación del sistema de reconocimiento haremos uso de una librería que permite desarrollar aplicaciones web para ciencia de datos y ML, como lo es Streamlit que brinda la posibilidad de abrir por medio del cualquier navegador la aplicación desarrollada, cargando el modelo entrenado, es importante establecer el guardado del modelo en la memoria cache del navegador mediate Streamlit para reducir el tiempo de espera en la carga del programa (Richards, 2021).

RESULTADOS

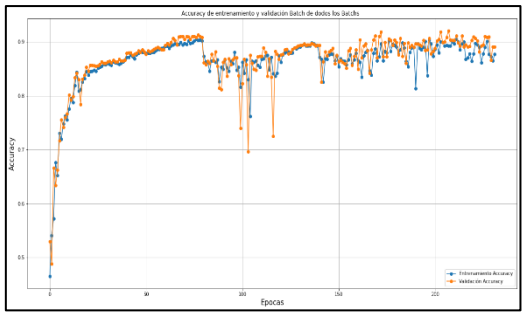
Análisis de entrenamiento de Batchs

Para los análisis de los batchs o lotes haremos uso de gráficas de la época frente al exactitud (accuracy) y perdida (loss).

Loss y Acuraccy de todos los Batchs entrenados

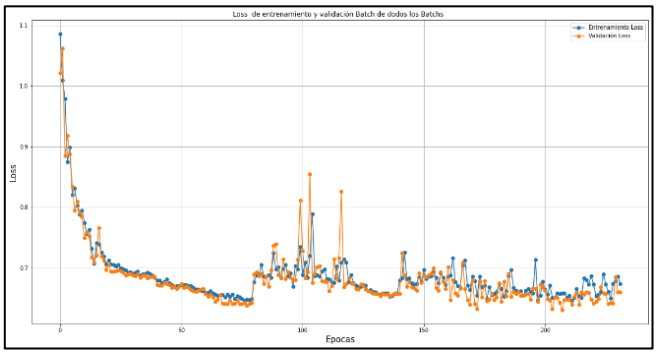
En el gráfico 1 y 2 se observa las métricas de la función de exactitud (Accuracy, por su nombre en inglés) del entrenamiento de 12 lotes de datos con la cantidad de 180 mil imágenes de las cuales el 80% se destinaron para el entrenamiento y el 20% para la validación. De estos, se pudo determinar que en algunos se dieron casos de underfitting y overfitting es decir, la falta y sobre entrenamiento de la red neuronal, utilizando una función de decaimiento exponencial en el optimizador Adam, se pudieron solucionar los problemas de capacidad del modelo.

Gráfico 1. *Exactitud (Accuracy) de entrenamiento y validación los largo de 12 Batchs*



Fuente: Luna Bryan, 2021

Gráfico 2. *Perdida (loss) entrenamiento y validación los largo de 12 Batchs*



Fuente: Luna Bryan, 2021

Decaimiento exponencial de la función de pérdida

Se propone un decaimiento exponencial a los valores de Learning rate , para que le aprendizaje sea más rápido ajustándolo a un valor de 0.0001 con una base de decaimiento de 0.96 por cada 100000 pasos, como resultado obtenemos que tanto los valores de perdida como exactitud se estabilizan en el batch 2 que comprenden las épocas (epochs) 20 a la 40.

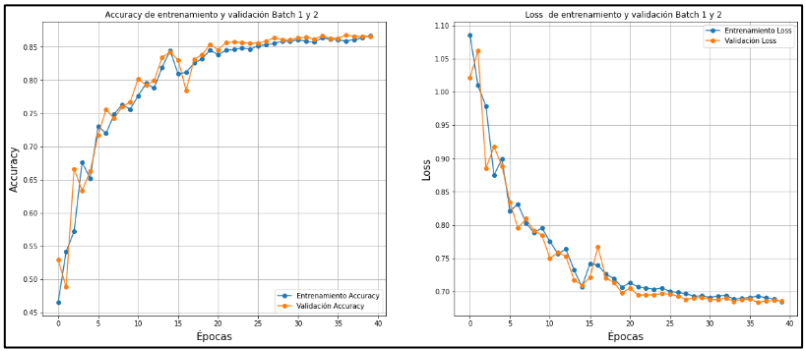


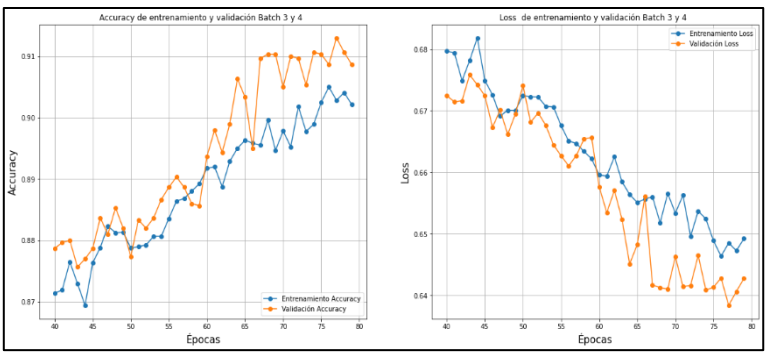
Gráfico 3. *Lote 1 sin decaimiento exponencial con lote 2 con decaimiento exponencial*

Fuente: Luna Bryan, 2021

Interpretación de underfitting

Debido al ajuste con el decaimiento exponencial grafico 4, la presencia del underfitting es muy notable en el lote 3 y 4 de la época 40 a la 80 y esto se debe precisamente al tamaño de paso o Learning rate muy pequeño (V. Subramanian, 2018).

Gráfico 4. *Presencia de underfitting en el lote 3 y 4 debido a la aplicación del decaimiento exponencial*

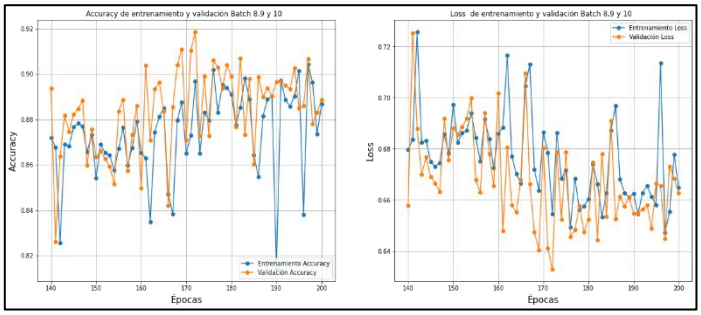


Realizado por: Luna, Bryan 2021.

Para este punto debido a la presencia de Underfitting se procede a cancelar el decaimiento exponencial del Learning rate en los batchs 5 y 6 , sin embargo se genera ruido en cuanto a las

funciones de perdida y exactitud en estos batchs que comprenden de la época 80 a la 120 como se aprecia en el grafico 5.

Gráfico 5. *Batch 5 y 6 sin decaimiento exponencial, Batch 7 con Decaimiento exponencial en el compilador*

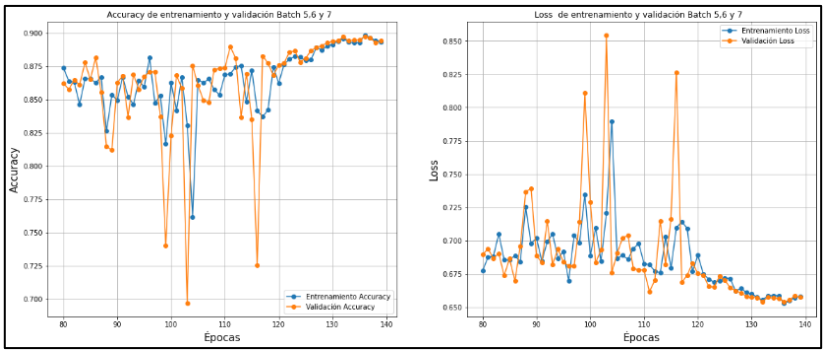


Realizado por: Luna, Bryan 2021.Esto se debe a que sin el decaimiento exponencial del Lerning rate el optimizador toma valores estocásticos en busca del mínimo local adecuado, y con el fin de amortiguar el ruido nuevamente se aplica el decaimiento exponencial del Learning rate y aplicándolo al batch 7 que comprende la época 120 a la 140.

Reducción del ruido de las graficas

Pese que el Batch 7 se aplicó un decaimiento exponencial en el Learning rate o tamaño de paso, si la gráfica bien se estabilizó se pudo notar la presencia underfitting nuevamente, por lo que se estableció entrenar la red neuronal sin decaimiento exponencial como se muestra en la gráfica 6, pese a la presencia de ruido a medida que se entrenando en cada una de las diferentes épocas este se iba reduciendo, esto también se puede solucionar añadiendo más épocas de entrenamiento mediante el análisis de las matrices de confusión de los modelos entrenados y guardados se fue contando la eficacia de los mismos hasta determinar un modelo óptimo.

Gráfico 6. *Batch 8,9 y 10 sin decaimiento exponencial*

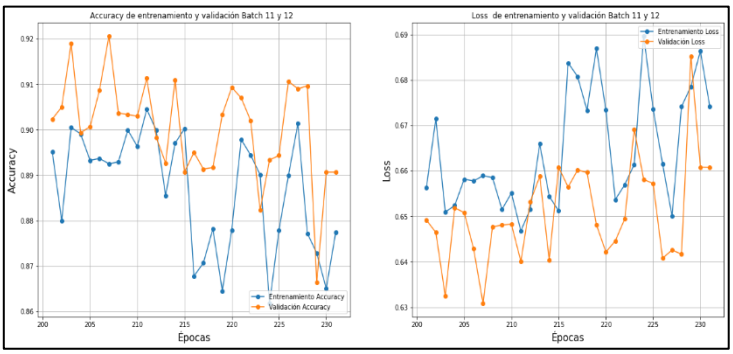


Realizado por: Luna, Bryan 2021.

Decrecimiento de Accuracy y Aumento de Loss

La grafica 7 muestra el decremento de la función de accuracy y un incremento de la función de perdida, sin embargo, el entrenamiento se detiene debido a que aplica un early sptopping que evalúa la métrica de la exactitud de validación o val accuracy, esto significa que el entrenamiento de la red convolucional para imágenes médicas estaría perdiendo facultades para el reconocimiento de imágenes.

Gráfico 7. *Batch 11 y 12 decremento de accuracy y aumento de loss*



Realizado por: Luna, Bryan 2021

Determinación del mejor modelo

Para determinar el mejor modelo para el reconocimiento del Sars-cov-2 que produce la enfermedad del Covid 19 se procedió a evaluar las correspondientes matrices de confusión decada uno de los modelos evaluando los falsos positivos y falsos negativos así como también las métricas: precisión, recall, f1 score y Accuracy.(Goutte & Gaussier, 2005).

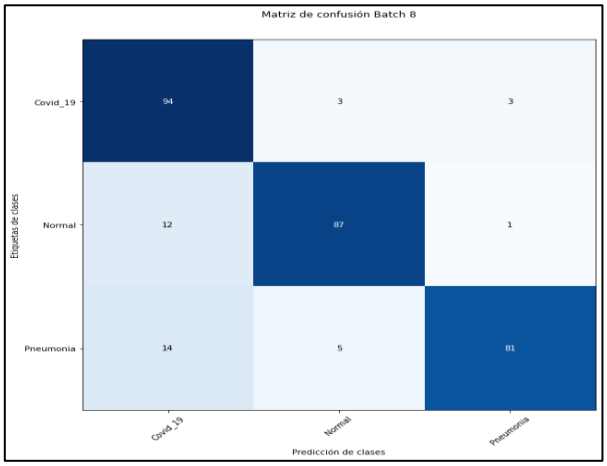
Tabla 6. *El Mejor modelo se obtuvo en el Batch 8*

	Precision	Recall	F1-Score	support
Covid_19	0.8533	0.9400	0.8545	100
Normal	0.9158	0.8700	0.8923	100
Pneumonia	0.9529	0.8100	0.8757	100
Accuracy			0.8833	300

Elaborado por: Luna, Bryan, 2021.

En la tabla 6 se muestra la eficiencia del modelo entrenado que corresponde al batch 8 de con un resultado del 88 % en la exactitud (Accuracy) del modelo, con tan solo un 12% de error, eso después de la evaluación con una matriz de confusión tomado 300 imágenes de prueba, 100 con etiquetas correspondientes a Covid 19, 100 para Penumonia y 100 para imágenes normales sin patología alguna (Ting, 2010).

Gráfico 8. *Matriz de confusión del mejor modelo*



Realizado por: Luna Bryan 2021.

Con la evaluación de los modelos gracias a sus respetivas matrices de confusión y métricas, se hace uso del modelo con el mejor resultado y mediante la función " model.predict" de tensor Flow se procede a reconocer las imágenes según las clases con las que se ha entrenado el modelo, las cuales están descritas en la tabla 2-3. Para finalizar se procede a elaborar un Script quepermita dar una interfaz gráfica al usuario para su mejor comodidad haciendo uso de Streamlit en Python.

Tabla 7. *Clases resultantes del reconocimiento*

Clases	Diagnóstico
Covid 19	Indica que la imagen evaluada presenta la presencia de SARS-CoV-2
Normal	Indica que la imagen no presenta ninguna patología
Pneumonia	Indica que la imagen presenta neumonía adquirida por SARS-CoV-2

Elaborado por: Luna Bryan,2021.

Interfaz web

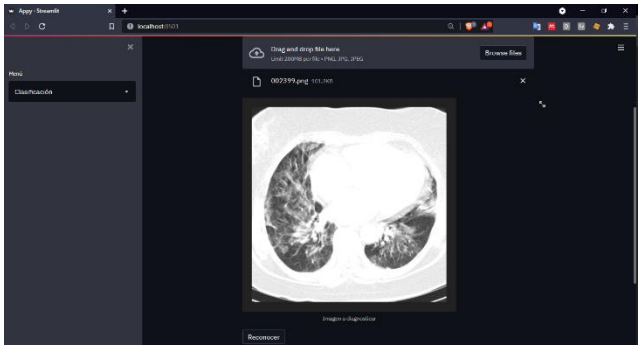
Se procede al desarrollo de una interfaz web utilizando la librería de Streamlit.

Figura 13. *Interfaz aplicación web*

Realizado por: Luna Bryan 2021.

En la figura 14 se muestra la sección de clasificación del menú para proceder a cargar la imágen, dando la función de arrastrar la imagen desde una carpeta o cargarla desde una ventada de búsqueda.

Figura 14. *Carga de imagen en formato PNG*



Realizado por: Luna Bryan 2021.

En la figura 15 se muestra la imagen cargada sin el análisis correspondiente, se prede a hacer un clic en el botón reconocer que se encuentra en la parte inferior.

Figura 15. *Reconocimiento de la imagen*

Realizado por: Luna Bryan 2021.

En la figura 16 muestra el resultado final de la clasificación de imágenes médicas, este resultado muestra una eficacia del 88 % con una tasa de error del 12 %

DISCUSIÓN

Se desarrolló un sistema de reconocimiento de imágenes médicas, con la capacidad de reconocer Covid 19 y neumonía, mediante una red neuronal convolucional, con una efectividad del 87 % que posteriormente se presentó mediante una aplicación web. utilizando la librería Streamlit.

Se creó un modelo de red neuronal convolucional pre entrenado, a partir de otro llamado EfficentNet y se lo adapto para reconocer imágenes médicas con la presencia de SARS-CoV-2, que produce la enfermedad del Covid 19, así como también la capacidad de reconocer patrones de neumonía en las imágenes médicas.

Se incorporó al software imágenes producto del manejo de archivos de imágenes médicas DICOM , así como el de un data set de terceros en Kaggle

Se indagó sobre el método idóneo para crear un sistema de reconocimiento de imágenes médicas con la presencia de SARS-CoV-2, optando por el uso de Python como lenguaje de programación, Pydicom para archivos DICOM, Pillow para archivos de Imágenes, Numpy para arrays de pixeles, librerías para ML como TensorFlow, el uso del servicio de terceros como

Google Cloud para el entrenamiento de la red neuronal convolucional y Streamlit para la creación de una interfaz web amigable para el uso del usuario.

Se recopiló archivos de imágenes médicas DICOM de un tomógrafo Siemens Somatom de 2 cortes, mismas que fueron utilizadas para el entrenamiento de la red neuronal.

Realizar un correcto etiquetado de imágenes, descartando aquellas que no coincidan con el propósito de utilizarse para apreciar la presencia de Covid 19 y neumonía.

Aplicar técnicas de regularización, como el aumento de datos y el decaimiento exponencial de Learning Rate con el fin de prevenir overfitting y underfitting, debido a que se puede afectar la capacidad del modelo.

Instalar un entorno virtual de Python 3.5 y librerías como, Numpy, TensorFlow, Pillow , Pydicom y Streamlit, el análisis de datos de imágenes médicas así como también para ejecutar la aplicación web.

Para el entrenamiento de grandes cantidades de datos tengan la configuración de clústers de máquinas o instancias, dividiendo de esa forma la carga de trabajo, permitiendo entrenamientos más rápidos, dando la posibilidad de aumentar la cantidad de épocas para cada lote de imágenes.

Para evitar la pérdida de modelos entrenados es útil aplicar callbacks de guardado automático, evitando así el progreso del entrenamiento previo en caso de alguna falla con el sistema.

REFERENCIAS

Cai, K., Yang, R., Chen, H., Li, L., Zhou, J., Ou, S., & Liu, F. (2017). A framework combining window width-level adjustment and Gaussian filter-based multi-resolution for automatic whole heart segmentation. Neurocomputing, 220, 138–150. https://doi.org/10.1016/j.neucom.2016.03.106

Geewax, J. (2018). Google Cloud Platform in Action. Manning Publications Co.

Goutte, C., & Gaussier, E. (2005). A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. In Lecture Notes in Computer Science (pp. 345–359). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-31865-1_25

Hope, T., Resheff, Y. S., & Lieder, I. (2017). Learning TensorFlow. In Learning Tensorflow. O´Reilly.

Jung, H. (2021). Basic Physical Principles and Clinical Applications of Computed Tomography. Progress in Medical Physics, 32(1), 1–17. https://doi.org/10.14316/pmp.2021.32.1.1

Mason, D. (2011). SU-E-T-33: Pydicom: An Open Source DICOM Library. In Medical Physics (Vol. 38, Issue 6Part10, p. 3493). https://doi.org/https://doi.org/10.1118/1.3611983

Oliphant, T. E. (2006). Guide to NumPy. Massachusetts Institute of Technology.

Richards, T. (2021). Getting started with Streamlit for data science create streamlit applications from scratch. Packt Publishing.

Sande, A., & Ramdurg, P. (2020). Comparison Of Hounsfield Unit Of CT With Grey Scale Value Of CBCT For Hypo And Hyperdense Structure. European Journal of Molecular & Clinical Medicine, 07, 4654–4658.

Serna, W., & Trujillo, J. (2010). Descripción del estándar DICOM para un acceso confiable a la información de las imágenes médicas. 2(45), 289–294. https://doi.org/10.22517/23447214.347

Stevens, E., Antiga, L., & Viehmann, T. (2020). Deep Learning with PyTorch. Manning Publications Co.

Subramanian, S., Wang, L. L., Mehta, S., Bogin, B., van Zuylen, M., Parasa, S., Singh, S., Gardner, M., & Hajishirzi, H. (2020). MedICaT: A Dataset of Medical Images, Captions, and Textual References. Findings of the Association for Computational Linguistics, 2112–2120.

Subramanian, V. (2018). Deep Learning with PyTorch. In Publications of the Astronomical Society of the Pacific (Vol. 88). Packt Publishing Ltd. https://doi.org/10.1086/129982

Thakurratan, R. S. (2018). Google Cloud Platform Administration Design (Vol. 148). Packt Publishing.

Ting, K. M. (2010). Confusion Matrix. In C. Sammut & G. I. Webb (Eds.), Encyclopedia of Machine Learning (p. 209). Springer US. https://doi.org/10.1007/978-0-387-30164-8_157

Usmani, Z. (2016, December). What is Kaggle, Why I Participate, What is the Impact? | Data Science and Machine Learning.

Varma, D. R. (2012). Managing DICOM images: Tips and tricks for the radiologist. Indian Journal of Radiology and Imaging, 22(01), 4–13. https://doi.org/10.4103/0971-3026.95396

Viera Maza, G. I. (2017). Procesamiento de imágenes usando OpenCV aplicado en

Raspberry Pi para la clasificación del cacao. In Thesis.