ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE CIENCIAS

CARRERA FÍSICA

"RECONOCIMIENTO DE LA PRESENCIA DE SARS-COV-2 EN PULMONES A TRAVÉS DE IMÁGENES DE RADIODIAGNÓSTICO HACIENDO USO DE MACHINE LEARNIG CON LENGUAJE DE PROGRAMACIÓN PYTHON"

Trabajo de Titulación

Tipo: Proyecto de Investigación

Presentado para optar el grado académico de:

FÍSICO

AUTOR: BRYAN DARWIN LUNA BRAVO

DIRECTOR: Dr. RICHARD WILLIANS PACHACAMA CHOCA Msc.

Riobamba -ecuador

2021

Yo, Bryan Darwin Luna Bravo, declaro que el presente trabajo de titulación es de mi autoría y los resultados del mismo son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor asumo la responsabilidad legal y académica de los contenidos de este trabajo de titulación; el patrimonio intelectual pertenece a la Escuela Superior Politécnica de Chimborazo.

Riobamba, 8 de diciembre de 2021.

Bryan Darwin Luna Bravo

CI: 2350215600

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE CIENCIAS

CARRERA FÍSICA

El Tribunal del Trabajo de Titulación certifica que: El trabajo de titulación; Tipo Trabajo de Investigación **"RECONOCIMIENTO DE LA PRESENCIA DE SARS-COV-2 EN PULMONES A TRAVÉS DE IMÁGENES DE RADIODIAGNÓSTICO HACIENDO USO DE MACHINE LEARNIG CON LENGUAJE DE PROGRAMACIÓN PYTHON"** ,realizado por el señor: **BRYAN DARWIN LUNA BRAVO**, ha sido minuciosamente revisado por los Miembros del Trabajo de Titulación, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal Autoriza su presentación.

	FIRMA	**FECHA**
Biof. María Fernanda Heredia Moyano. Msc. **PRESIDENTE DEL TRIBUNAL**	Firmado electrónicamente por: **MARIA FERNANDA HEREDIA MOYANO**	2021-12-03
Dr. Richard Willians Pachama Choca Msc. **DIRECTOR DEL TRABAJO DE TITULACIÓN**	RICHARD WILLIANS PACHACAMA CHOCA c=EC, serialNumber=0601921703, sn=PACHACAMA CHOCA, cn=RICHARD WILLIANS PACHACAMA CHOCA, 1.3.6.1.4.1.37442.10.4=0601921703, givenName=RICHARD WILLIANS, email=rpachacama@gmail.com, st=RIOBAMBA, l=CHIMBORAZO, ou=Certificado de Clase 2 de Persona Física EC (FIRMA)	2021-12-03
Mat. Luis Marcelo Cortez.Bonilla Msc. **MIEMBRO DEL TRIBUNAL**	LUIS MARCELO CORTEZ BONILLA Firmado digitalmente por LUIS MARCELO CORTEZ BONILLA Nombre de reconocimiento (DN): c=EC, o=BANCO CENTRAL DEL ECUADOR, ou=ENTIDAD DE CERTIFICACION DE INFORMACION-ECIBCE, l=QUITO, serialNumber=0000497075, cn=LUIS MARCELO CORTEZ BONILLA Fecha: 2022.01.20 18:40:31 -05'00'	2021-12-03

DEDICATORIA

Este trabajo de investigación se lo dedico a Dios que me ha colmado de bendiciones, entre ellas mi hermosa familia.

Bryan

AGRADECIMIENTO

Agradezco a Dios por haberme colmado de bendiciones al largo de mi vida por ser ese apoyo incondicional en tiempos difíciles y esa luz que alumbra mi vida día a día.

Agradezco a mis padres Darwin Luna y Sobeida Bravo que han sido mi fuerza y mi motor, rectificando mi camino y apoyándome incondicionalmente, sin duda, una de las bendiciones más grandes que Dios me ha dado.

Bryan

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE GRÁFICOS

ÍNDICE DE ECUACIONES

ÍNNDICE DE ANEXOS

RESUMEN

El objetivo de este proyecto fue 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 claves: <IMÁGENES MÉDICAS>, <TOMOGRAFÍA COMPUTARIZADA>, <COVID 19>, <SARS-COV-2> <MACHINE LEARNING>, <RED NEURONAL ARTIFICIAL (RNA)>.

ABSTRACT

The goal of this project 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>, <SARS-COV-2> <MACHINE LEARNING>, <ARTIFICIAL NEURAL NETWORK (ANN)>.

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 diagnostico 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
El presente trabajo de Titulación consta de 3 capítulos:

En el Capítulo I, se abordan los antecedentes, planteamientos del problema, justificación y la bibliografía necesaria, para el entendimiento del trabajo

En el Capítulo II es el marco metodológico, donde se tratará acerca de la realización del proyecto, los pasos a seguir y diferentes técnicas utilizar para la elaboración de un sistema de reconocimiento de imágenes médicas.

En Capítulo III se tomarán los resultados obtenidos para la evaluación del sistema de reconocimiento tomando en consideración los diferentes parámetros que componen una matriz de confusión.

CAPÍTULO I

1. MARCO TEÓRICO REFERENCIAL

1.1. Antecedentes de la investigación

Gracias a las múltiples herramientas computacionales y lenguajes de comunicación, se han implementado redes neuronales para identificar o clasificar imágenes, partiendo de un data set de entrada y salida uno de los primeros trabajos corresponde a Martínes,(2015, p 1), quien realizó una tesis: *"Reconocimiento de Imágenes mediante Redes Neuronales Convolucionales"* . En este trabajo se manejaron conceptos de Machine Learning(ML) y Deep Learning(DL) cuyo objetivo fue elaborar un programa que es capaz de reconocer una especie biología, en este caso las ballenas jorobadas a partir de imágenes tomadas de sus colas , implementando una red neuronal, donde el análisis y preprocesado de la imagen se le realiza utilizando TensorFlow(Tf) y Keras. Así mismo el estudio de diferentes técnicas actuales relativas al campo de la IA, principalmente al campo del aprendizaje profundo.El proyecto se realizó obteniendo un data set de imagines de las colas de ballenas jorobadas como patrones de reconocimiento, así como un lenguaje de programación, resolviendo la utilización de Python ya que permite la implementación de librerías tales como TensorFLow y Keras que son capaces de satisfacer las necesidades en cuando a ML y DL se refiere. Concluyendo que se pudo comprobar que el rendimiento de una red neuronal no radica únicamente en la arquitectura sino también en el conjunto de datos o data set , siendo este balanceado y con gran cantidad de muestras.(Martínez, 2015. p 3).

Tenemos también otro proyecto realizado por Senén (2019, p.5) denominado *"Diseño, implementación y evaluación una red neuronal convolucional de regresión en clasificación de naranjas"* este trabajo se ha centrado en la creación y edición de forma supervisada de un conjunto de imágenes en bien etiquetadas, para ello se utiliza una herramienta de etiquetado disponible en la institución donde se realizó este proyecto, posteriormente se diseñaron diferentes arquitecturas de redes neuronales implementando regresiones logrando optimizar dichas arquitectura haciendo uso de Python y Pytorch. Los resultados obtenidos de estas redes neuronales convolucionales muestran que son un método potencial para la preselección y selección de cítricos, es decir, se logra un proceso de automatización de acuerdo con las características del objeto a analizar concluyendo en que una de las más relevantes variables es que el conjunto de datos utilizados influye en las capacidades de predicción de las redes neuronales.

1.2. Identificación del Problema

Los exámenes realizados por medio de radio diagnóstico requieren una previa revisión de un especialista en este caso de un médico radiólogo, el cual está capacitado para diagnosticar patologías por medio de imágenes. En este último año hemos sido testigos de cómo ciertas enfermedades pueden ser fácilmente detectadas por métodos tomográficos, generando imágenes médicas. Así tenemos al Covid 19, puesto que la presencia de esta infección causa manchas en los pulmones denominadas vidrios deslustrados, las cuales son identificadas gracias a las imágenes ya antes mencionadas.

Existen situaciones donde no se cuenta con la presencia del médico radiólogo para que efectuase un diagnóstico, entonces, los estudios radiológicos toman un determinado tiempo en realizarse retrasando así el proceso de tratamiento para el paciente, sin embargo, esto se podría solucionar haciendo uso de las nuevas tecnologías a disposición y empleando la inteligencia artificial como herramienta, siendo esta una idea viable gracias a los avances de las últimas décadas.

Con todo esto nos podemos plantear las siguientes preguntas ¿Pero, y si no se cuenta con la presencia de un médico especialista? Además ¿se podría de cierta manera "automatizar" el diagnóstico de patologías que poseen características similares?, tenemos los suficientes recursos para identificar anomalías gracias a las nuevas tecnologías de procesamiento de datos como la inteligencia artificial y sus determinadas ramas como el Machine Learning y Deep Learning.

1.3. Justificación del problema

Día por día la humanidad avanza y así mismo las enfermedades causadas por virus, esto supone que el ser humano debe emplear su ingenio para prevenirlas y tratarlas lo más eficientemente posible. Hasta el momento se tienen los métodos de diagnóstico asistidos por personal médico, sin embargo, sabemos que el reconocimiento de la existencia de una patología se da gracias a métodos cuantitativos y cualitativos de ciertas características o anomalías en el cuerpo que son identificados con la ayuda de diferentes exámenes médicos.

En este presente proyecto hablaremos sobre la determinación de Covid 19, de cómo su presencia y caracterización dentro de los pulmones puede ser diagnosticada sin la presencia de un médico. Gracias a las imágenes médicas, ya sea por la adquisición de tomografías, una vez obtenidas la imágenes se puede obtener cierta información que nos permite caracterizar la patología, sin embargo en cuanto al COVID 19 , la presencia de características en los pulmones, ahora gracias a la particular forma en la que se manifiesta esta patología dentro de los pulmones es posible

“automatizar” este tipo de diagnóstico pues basándonos en los innumerables diagnósticos hechos por las manos de un médico Radiólogo alojados en una base de datos se puede crear un sistema computacional que haga uso de estos y diagnostique nuevos exámenes haciendo uso de la inteligencia artificial mediante redes neuronales convolucionales, simplificando así la tarea del médico radiólogo obteniendo un diagnóstico

Para llevar a cabo el Machine Learning (ML) es necesario una banco de información , dentro de un servidor PACS una vez ya previamente revisada y diagnosticada tratándose en este caso específicamente al COVID 19**,** y luego aplicarlo mediante el lenguaje de programación Python siendo este el lenguaje elegido gracias a que es un leguaje de fuente abierta ,posee un propósito general y que posee una colección de librerías o bibliotecas para poder trabajar con imágenes como Pillow y para la modelaciones de redes neuronales TensorFlow.

1.4. Objetivos

1.4.1. Objetivos General

- Desarrollar un sistema que permita reconocer SARS-COV2 mediante imágenes médicas importadas a una red neuronal.

1.4.2. Objetivos Específicos

- Desarrollar un método de reconocimiento para el SARS COV2 mediante imágenes médicas para caracterizar la presencia del virus
- Incorporar las imágenes producto del método al software
- Investigar el método para desarrollar un algoritmo de red neuronal para reconocimiento de imágenes en mediante el lenguaje de programación Python
- Recopilar información de una base de datos que permita entrenar una red neuronal

1.5. Fundamentación teórica

1.5.1. Imágenes médicas

Actualmente conocemos la existencia de muchos procesos de diagnóstico que generan imágenes o una secuencia de imágenes médicas como Rayos X(Rx), tomografías computarizadas (CT), resonancias magnéticas (MR), ecografías (ECO), tomografía por emisión de positrones (PET CT) entre otras más. Gracias a la gran cantidad de información que se obtiene de estos procesos diagnósticos, en los años 70's surgen las necesidades del almacenamiento y manipulación de este tipo de imágenes denominado "Digital Imaging and Communication On Medicine" o conocido también con su abreviatura DICOM cuya extensión de archivo es "*.dcm".(Varma 2012, p. 5).
Es muy importante que la estandarización del formato de imágenes médicas consiste en garantizar la igualdad de condiciones desde el momento de la adquisición de un estudio imagenológico hasta el momento de ser procesada o impresa.(Serna y Trujillo 2010, p. 290).

1.5.2. Rayos x

El 1895 uno de los físicos más importantes del siglo 19 descubrió los rayos x, al poco tiempo de su descubrimiento se convirtió en una herramienta útil usada para fines médicos. Dado la naturaleza de estos, tienen la capacidad de penetrar los cuerpos opacos por su baja longitud de onda y su alta energía. Son producidos en un tubo de rayos X que es un conversor específico de energía que funciona al recibir una energía eléctrica y transformarla en dos formas de energía de radiación que corresponde al 1% y calor que corresponde al 99%. En este proceso, los electrones son acelerados a través de un gradiente de campo eléctrico con un alto voltaje e impactar con un metal en su mayoría tungsteno, produciendo una desaceleración de estos, este alto voltaje suele estar entre los 10 kV y 100 kV junto con una intensidad electrones que están en un rango de 100 a 50 mA. (Ramírez Giraldo, Arboleda Clavijo y McCollough 2008, p. 16).

El tubo tiene dos elementos principales: el cátodo que es un filamento de tungsteno calentado con electricidad que proporciona la fuente electrónica y el otro es el ánodo, formado por una aleación de tungsteno donde se producen una radiación electromagnética, con un aspecto continuo de energía entre los 20 y 200 KeV. La atenuación exponencial es el fundamento en que la aplicación de rayos x está basada dentro de la medicina ya que estos tienen la capacidad para atravesar ciertos materiales que se crucen en su camino, pueden ser absorbidos o a su vez dispersados, disminuyendo la intensidad original. Este tipo de evento depende de la estructura y el grosor del material. Tanto en absorción como en la dispersión se debe a interacciones entre los fotones y

electrones del interior del átomo las interacciones más importantes energía y de interés en el área del radio diagnóstico son el efecto fotoeléctrico y la dispersión Compton. (Fos Guarinos 2016, p. 8)

1.5.3. Tomografía computarizada

Godfrey Newbold Hounsfield presentó el primer tomógrafo computado en 1972. Este desarrollo tecnológico ha sido considerado fundamental que le permitió obtener el Premio Nobel de Medicina en 1979. La Tomografía Axial Computada (TAC) consta de tubo de RX que emite radiación a medida que gira alrededor del paciente, en el lado contrario existe una fila de cristales que detectan la radiación remanente luego de haber sido absorbida por los diferentes tejidos. El tubo y los detectores giran conjuntamente alrededor del paciente dando vueltas completas mientras los detectores reciben una y otra vez la información a lo largo de este proceso, los datos son trasladados a la computadora generando una imagen por sumatoria de aspectos en 2 planos del espacio. (Jung 2021, p. 15).

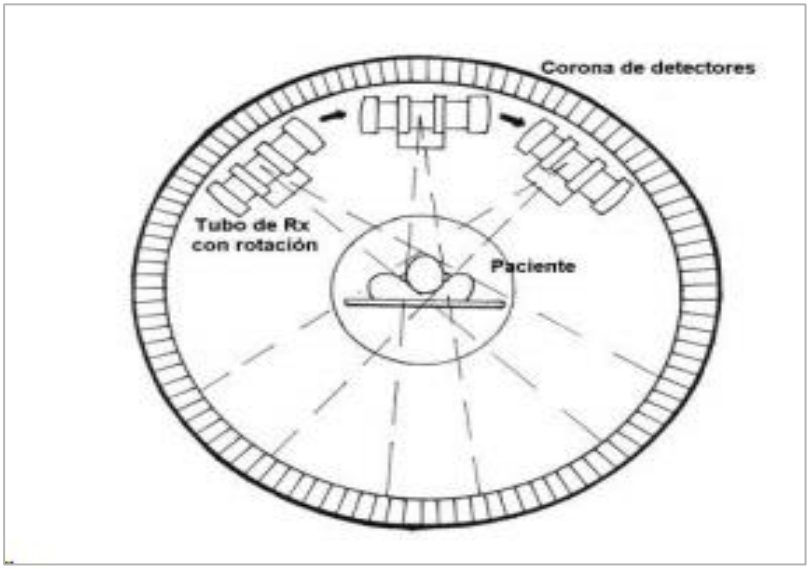


Figura 1-1: Diagrama de la estructura de los rayos X en una tomografía.
Fuente: Angerami, 2016., p.**4.**

La memoria del ordenador hace una recomposición de los datos recibidos y por medio de fórmulas matemáticas (Transformada de Fourier); asignando un valor o también llamado costo de absorción a cada volumen estudiado (Voxel.), que más adelante se representa en un sólo plano (Pixel.)(Angerami 2016, p. 4).

1.5.4. Escala de absorción o de unidades Hunsfield

En la tomografía computarizada se hace uso de una escala para determinar la densidad de los tejidos, esta es la escala de unidades Hunsfield(UH), que mediante la atenuación de rayos X se puede terminar rangos entre -1000 a +1000, cuyos valores dependerá del tejido en cuestión siendo -1000 un valor dado para la radiosensibilidad del aire , 0 para el agua destilada y +1000 para tejidos óseos (Sande y Ramdurg, 2020, párr. 5).

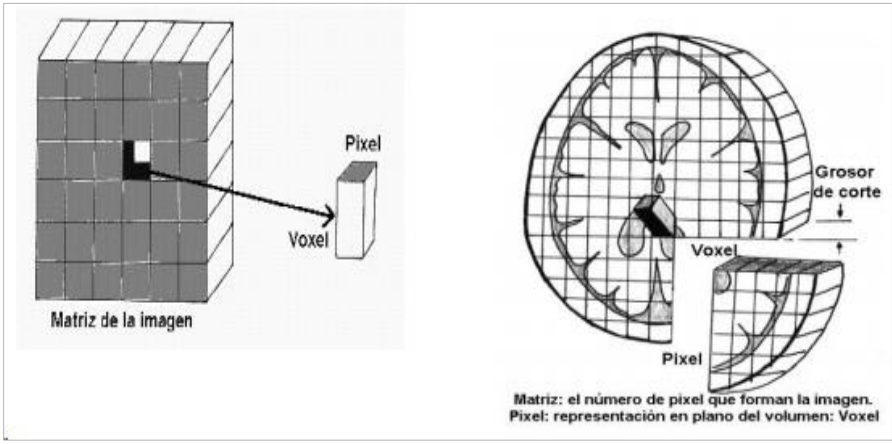


Figura 2-1: Reconstrucción de imagen a partir de voxeles y matriz de datos de una tomografía

Fuente: Angerami, 2016, p4.

1.5.5. SARS-CoV-2 2 en tomografías computarizadas

Desde primeras instancias una de las pruebas y métodos de diagnóstico para el Covid 19 es el método de diagnóstico por imágenes con una sensibilidad del 77% y especificidad del 96% , siendo un método igual de óptimo que reacción en cadena de la polimerasa con reverso transcripción (PCR-RT, del inglés reverse transcription polymerase chain reaction).

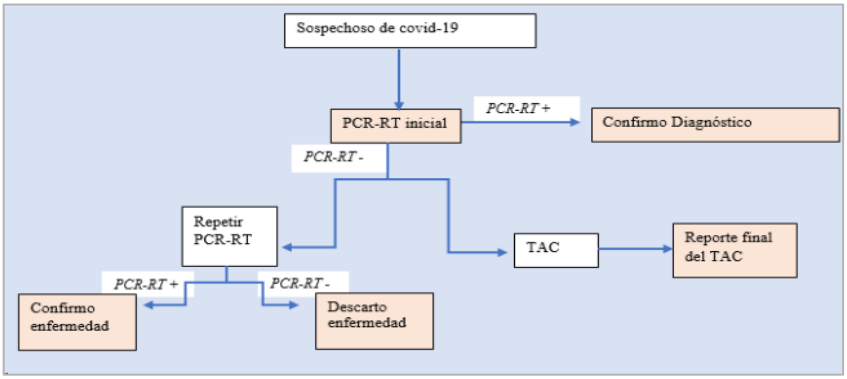


Figura 3-1: Algoritmo de diagnóstico de COVID-19

Fuente: Ortiz et al., 2020, p 354.

En las tomografías computarizadas podemos encontrar diferentes patrones tomográficos, sin embargo, ninguno de ellos están en capacidad de descartar o confirmar la presencia de SARS-CoV-2 virus que es causante de la enfermedad “Covid 19” , puesto que los hallazgos pueden solapar con otras infecciones incluyendo la influencia ,H1N1, SARS, MERS (Ortiz et al., 2020, pp. 354-358).

1.5.6. Fisiopatología de la Imagen tomográfica por Covid19

Dada su capacidad viral se sabe que el SARS-CoV2 se introduce en las células del epitelio alveolar, macrófagos alveolares y células endoteliales vasculares, para luego liberar su ARN al citoplasma del huésped, comenzando su replicación el virus infecta más células, provocando inflamaciones a nivel alveolar y sistémico, mediado por una respuesta inmunológica no regulada.(Gheblawi et al. 2020, p. 1457)

Los pacientes con Covid 19 que superan los 60 años suelen presentar Síndrome de Liberación de Citoquinas (SCL), el cual compromete la capacidad pulmonar provocando dificultad respiratoria aguda (Hinostroza 2020).Los hallazgos por tomográficos del Covid 19 se presentan relacionados a su característica fisiopatológica como se observan en las figuras 1-1, 2-1 y 3-1.

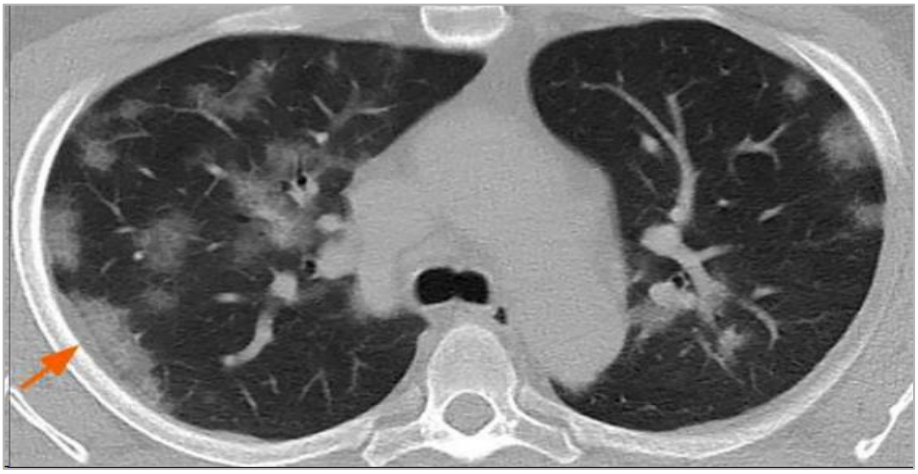


Figura 4-1: : Engrosamiento pleural.

Fuente: Ortiz, 2020, p.23.

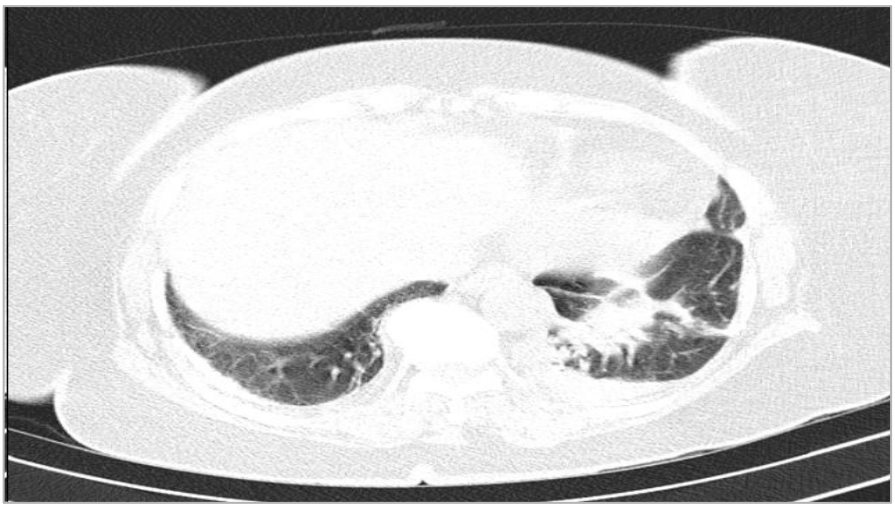


Figura 5-1: Consolidación en la base pulmonar derecha.

Fuente: PACS, 2021, p.23.

Figura 6-1: Opacidad de vidrio esmerilado en pulmón derecho.

Fuente: PACS, 2020, p.23.

1.5.7. Dataset

Para el proceso de aprendizaje de una red neuronal es necesario conocer la naturaleza de esta información, puede ser cualquiera siempre y cuando exista la manera de representarla y dentro de esta representación cada una de las variables correspondientes a un proyecto en específico, esta gran cantidad de datos servirá para el entrenamiento de la red neuronal y es conocido como Data-Sets. (Subramanian et al. 2020, p. 2113)

1.5.7.1. Dataset a partir de un sistema PACS

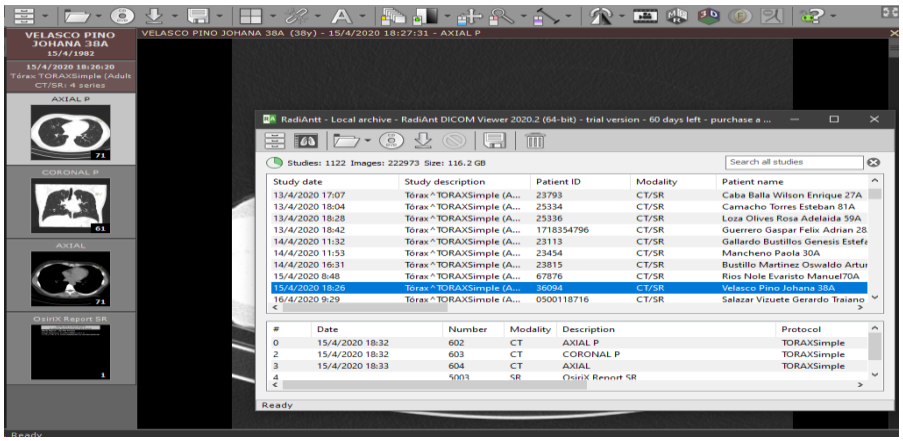


Figura 7-1: Estudios Tórax Simple en el sistema PACS(RadiAnt DICOM Viewer.).

Fuente: RadiAnt DICOM Viewer 2021.

Se conoce como sistema PACS al sistema de almacenamiento y distribución de imágenes su abreviatura corresponde a “Picture Archiving and Comunications System”. Este tipo de sistemas se asemeja a un servidor y comúnmente es asociado a la radiología ya que su labor principal es almacenar imágenes radiológicas en distintas modalidades ya sean tomográficas, rayos X, mamografías entre otras, en su mayoría el protocolo que utilizan los sistemas PACS es el DICOM.(Bordils y Chavarría 2008, p. 56)

1.5.7.2. Dataset desde Kaggle.

Kaggle es una plataforma que posee recursos de Machine learning (ML), además de contar con una gran comunidad de Data Scientist, donde se pueden encontrar muchos data-set creados por la comunidad, así como varios modelos de aprendizaje automático para la realización del proyecto en diferentes áreas. (Usmani, 2016, p.14).

1.5.8. Python

Python es un lenguaje de programación potente creado a finales de los 90s por Guido Van Rossum, es una muy buena opción para una gran cantidad de problemas y dada su baja complejidad, es un lenguaje ideal para comenzar a programar debido a su sintaxis, también es importante destacar que al ser un lenguaje de código abierto posee una gran cantidad de librerías implementada por los usuarios además de la gran librería preinstalada que incluye construcciones familiares como bucles ,declaraciones, matrices entren otras.(Ceder 2010, p. 19).

1.5.9. Variables

Las variables no son más que espacios en memoria que se reservan para almacenar valores, estos valores y caracteres se atribuyen de acuerde al tipo de datos que deseemos guardar en memoria, entre las más comunes, tenemos las clases de variables como enteros (int) , flotantes (float), booleanos (bool) y cadenas (str).(Ceder, 2010, pp. 19-21).

Se declara así:

```
# Asignar variables
Var_1 = int(9)
Var_2 = (34.6)
Var_3 = True
Var_4 = 'Cadena'
print(type(Var_1))
print(type(Var_2))
print(type(Var_3))
print(type(Var_4))

# salida
In [Out]:
    <class 'int'>
    <class 'float'>
    <class 'bool'>
    <class 'str'>
```

1.5.10. Listas como Arrays

Una *Array* en Python es similar a una matriz en java o cualquier otro lenguaje de programación, que se denota como una colección ordenada de objetos o valores básicos como enteros, flotantes, caracteres y se comportan como listas así como “listas de listas” según Ceder y Mcdonald (2010, p.23), además, esta funcionalidad con los módulos *arrays* pueden crear matrices de números con el uso de librerías como *Numpy*.(Oliphant 2006, p. 59).

Se puede definir de la siguiente forma:

```
# Asignar elementos a una lista X y Matrix
X = ['4' ,2, True, 7.4 ]
Matrix = [['4', 345],[2,4],[True, False]]
print(f'X: {X} \nMatrix es : {Matrix}')
```

```
print(f'Tipo de variable X:{type(X)}\nTipo de variable de la matriz: {type(Matrix)}')
# Importar modulo numpy
import numpy as np
# Crear Array de la variable Matrix
Array = np.array(Matrix)
print(f'Array: {Array} \nTipo de variable: {type(Array)}')
# Salida
In [Out]:
  X: ['4', 2, True, 7.4]
  Matrix es : [['4', 345], [2, 4], [True, False]]
  Tipo de variable X:<class 'list'>
  Tipo de variable de la matriz: <class 'list'>
  Array: [['4' '345']
   ['2' '4']
   ['True' 'False']]
  Tipo de variable: <class 'numpy.ndarray'>
```

1.5.11. Tensores

En algebra lineal los tensores es una generalización de escalares, vectores y matrices con el cual se puede representar fenómenos físicos que son representados estrictamente en sistema de referencia pero son independientes del mismo, sin embargo, las componentes de dicho tensor será dependientes del sistema de referencia donde se encuentre, entre los más representativos están los de tensores de orden 0 o escalares que representan magnitudes pero no dirección como la temperatura, luego tenemos los de orden 1 o vectores. Ayudan a representar una dirección y una magnitud como la aceleración y también están los de orden 2 que poseen magnitud y dos direcciones como la tensión.(Izquierdo, 2012, p.45)

En ML los tensores son una generalización de matrices n-dimensionales, también como una matriz de números dispuestos de una cuadrícula con un número variable de ejes (Goodfellow, Bengio y Courville 2016, p. 36).

Para el manejo de tensores en Python una de las librerías a usar es Tensor Flow o Pytorch que son librerías de ML que hacen uso de tensores para sus tareas de cómputo. Una idea muy representaba de un tensor nivel computacional es una imagen, puesto que esta posee 3 canales (RGB) que corresponde una matriz para los valores de color rojo, verde y azul.(Moreno 2019, p. 70).

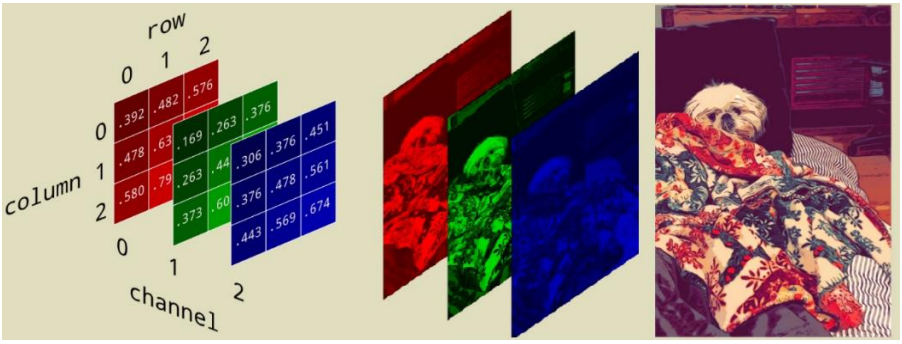


Figura 8-1: Imagen a manera de tensor con sus respectivos canales.

Fuente: Rohrer Diana, 2016, párr 4.

Para la representación de una imagen como en la figura 5-1 se utilizan módulos que obtiene las matrices de la imagen y la manejan como tensores con modulo *Numpy*, es importante recalcar que para esta tarea de deben importar módulos de lectura de imágenes como *Open-cv* o *Pillow*.(Viera Maza 2017, p. 91).

1.5.12. Funciones

Las funciones son un grupo de sentencias también conocidos como bloques de código que se encuentran estructurados dentro de un programa con el propósito de realizar una tarea en concreto, es de buena práctica establecer funciones para evitar la repetición de código, además de que existe la posibilidad de que una función se escriba como varias funciones más pequeñas que realicen tareas específicas para un fin común.(Davis 2019, p. 93).

```
# librerías a usar
import numpy as np
# defino una función con def
def funcion(Val_1, Val_2):
  matrix_1 = np.random.rand(Val_1,Val_2)
  matrix_2 = np.random.rand(Val_2,Val_1)
  Producto = np.dot(matrix_1,matrix_2) #Mutiplicacion de matrices
  print(Producto)
  return Producto
#invoco a la función "funcion" y la guardo en la variable G
G = funcion(3, 2)
#Salida
In [Out]:
  [[0.29412312 0.55184122 0.44558496]
```

[0.15572109 0.25811937 0.47496577]
[0.36834899 0.66085326 0.77043672]]

1.5.13. Objetos y Clases

Para definir las clases es importante conocer sobre la programación orientada a objetos (POO), este tipo de programación hace que el código de un programa pueda ser reutilizable, organizado y fácil de entender, un código reutilizable evita que se duplique líneas de código optimizando el programa , aquí es donde interviene el diseño orientado a objetos (DOO) que en sí, es el proceso de convertir e implementar el código mediante objetos y clases que poseen sus propios comportamientos convirtiendo un diseño de programa completamente definido. Los objetos son una colección de datos con comportamientos asociados que pueden ser funciones, estos objetos, tiene una misma naturaleza, se pueden agrupar en una clase de objetos que poseen de igual forma datos con comportamientos asociados. Las clases y objetos ayudan a definir el conjunto de datos para el entrenamiento en un proceso de ML, estos datos crean una clase con la información del conjunto de entrenamiento y validación, que será útil al momento de llevar a entrenar un modelo neuronal convolucional. Para crear una clase en Python se utiliza la palabra reservada *class* seguido del nombre que desee adjudicar a su clase en minúscula excepto la primera letra de la palabra. (Phillips 2010, p. 9).

```
# Creación de clases
class Paciente:
  pass
```

La palabra reservada *pass* indica que esta clases está vacía y no posee ninguna función por el momento, para contener una función dentro de una clase a las que llamaremos métodos.

```
#Asignar una funcion a la clases
class Paciente:
  def Sistoma():
    """Imprime en pantalla."""
    print('Tos')
```

El argumento *self*, dentro de la función o en este caso hace referencia a la instancia de la clase. Una clase puede contener variables , llamadas *atributos* , estos atributos están albergados en una función o como ya habíamos mencionado anteriormente el método __*init*__ conocido también

como método constructor, su característica principal es que invoca variables automáticamente siempre que se crea una instancia de la clase (Phillips 2010, p. 10).

Creación de una clase con variables vacías.

```
# Creación de la clase
class Paciente:
  def __init__(self):
    self.nombre = None
    self.edad = None
    self.sexo = None
  def _edad(self):
    print(f'La edad del paciente es: {self.edad} años')
  def _nombre(self):
    print(f'El nombre del paciente es: {self.nombre}')
  def _sexo(self):
    print(f'El sexo es : {self.sexo}')
```

Una vez definida la clase *Paciente* la instanciamos pero dado que las variables no tienen valor alguno o están vacías *None* , agregamos valores a cada una de sus variables para luego hacer uso de los métodos en la clase.

```
# instanciamos y damos valores a las variables
Paciente_1 = Paciente()
Paciente_1.nombre = 'Pedro Lopez'
Paciente_1.edad = '25'
Paciente_1.sexo = 'Masculino'
#hacemos uso de los métodos de la clase
Paciente_1._edad()
Paciente_1._nombre()
Paciente_1._sexo()
#salida
In[Out]:
  La edad del paciente es: 25 años
  El nombre del paciente es: Pedro Lopez
  El sexo es : Masculino
```

1.6. Módulos y Librerías

1.6.1. Numpy

Numpy es uno de los paquetes fundamentales en cuanto a computación científica se refiere en Python, esta biblioteca proporciona un objeto denominado *array* o matriz que puede ser multidimensional y también varios objetos derivados como matrices de matrices, además, de una gama de operaciones entre *arrays*, vectores y escalares, posee también funciones matemáticas, probabilísticas, estadísticas. Este módulo es ideal para el manejo de matrices matemáticas además de matrices de imágenes (Oliphant 2006, p. 13).

```
# importamos el módulos Numpy
import numpy as np

A = np.array([2,5,6,7])
B = np.array([1,3,5,7])
C = np.matrix([A,B])

print('Array A:',A)
print('Array B:',B)
print(f'Matriz de A y B: \n{C}')

#salida
In[Out]:
  Array A: [2 5 6 7]
  Array B: [1 3 5 7]
  Matriz de A y B:
  [[2 5 6 7]
   [1 3 5 7]]
```

1.6.2. Pydicom

Pydicom es una librería de Python para trabajar con archivos DICOM (*.dcm, extensión de archivo DICOM) que son los archivos por defecto de las imágenes médicas, esta al ser una librería pura en Python puede ejecutarse sin ningún otro requisito sin embargo cuando se quiere trabajar con los datos de los pixeles de las imágenes es recomendable hacer unos de *Numpy*.(Mason, 2011, p.34).

```
# Importamos modulo pydicom
import pydicom as pyd

# leemos el archivo .dcm
Archivo = pyd.read_file('archivo.dcm')
print(Archivo)

#salida
In[Out]:
 Dataset.file_meta -------------------------------
 (0002, 0000) File Meta Information Group Length UL: 186
 (0002, 0001) File Meta Information Version    OB: b'\x00\x01'
 (0002, 0002) Media Storage SOP Class UID     UI: CT Image Storage
 (0002, 0003) Media Storage SOP Instance UID   UI: 1.3.12.2.1107.5.1
 (0002, 0010) Transfer Syntax UID         UI: Explicit VR Little
 (0002, 0012) Implementation Class UID      UI: 1.2.804.114118.3
 (0002, 0013) Implementation Version Name     SH: 'RadiAnt-2020.2'
 -------------------------------------------------
 (0008, 0005) Specific Character Set       CS: 'ISO_IR 100'
 (0008, 0008) Image Type             CS: ['AXIAL', 'CT_SOM5 SPI']
 (0008, 0016) SOP Class UID            UI: CT Image Storage
  .. .
```

Como salida obtenemos toda la metadata correspondiente al archivo .dcm entre las cuales también se encuentra los pixeles de la imágenes.

1.6.3. Tensor Flow

TensorFlow (Tf) es un framework de código abierto desarrollado por Google que sirve para crear y entrenar modelos de Inteligencia Artificial (AI, por sus siglas en el inglés), que tiene como fin llevar cálculos numéricos basados en gráficos de fujo de datos. *Tf* fue diseñado con la posibilidad de que los grafos se ejecuten en una amplia variedad de entornos con un código esencialmente idéntico una red neuronal podría ser entrenada en la nube e inclusive distribuida en un clúster de máquinas, el núcleo de *Tf* está escrito en C+ +. Dado que *Tf* puede entrenar y ejecutar redes neuronales profundas para la clasificación de imágenes, reconocimiento de imágenes , inclusión

de palabras , redes neuronales recurrentes así como también modelos de secuencia a secuencia para la traducción automáticas entre muchos más (Hope, Resheff y Lieder, 2017, p.43).

Además es sencillo a la hora de expresar conceptos de aprendizaje automático en el lugar de ecuaciones matemáticas complejas proporcionando diferentes tipos de variables, y la utilización se matrices sin dejar a un lado el uso de tensores y la implementación de algoritmos de optimización de redes neuronales (Geewax 2018, p. 488).
Para empezar a utilizar esta librería *Tf* a manera de previa visualización debemos importarlo como tensorflow.

```
# importamos tensorflow
import tensorflow as tf
# creamos Arrays a manera de tensores
Tensor_1 = tf.constant([[1,3],[4,7]])
Tensor_2 = tf.constant([[3,7],[49,5]])
# procedemos a realizar una multiplicación de tensores
Producto_Tensor= tf.matmul(Tensor_1, Tensor_2)

print(f'El producto es: \n{Producto_Tensor } \n')
print(f'El tipo de variable: \n {type(Producto_Tensor)}')
```

```
# Salida
In[out]:
  El producto es:
 [[150 22]
  [355 63]]

 El tipo de variable:
  <class 'tensorflow.python.framework.ops.EagerTensor'
```

1.6.4. Google Cloud

Google Cloud es una colección de servicios y productos que brinda parte de la infraestructura interna de Google, servicios como almacenamiento, base de datos inclusive máquinas virtuales bajo demanda a través de Google Compute Engine. Aunque existen otros proveedores similares como Amazon Web Services (AWS) o Azure de Microsoft (Thakurratan 2018, p. 10)

Las ventaja de usar un servicio de nube como Google Cloud es que el alojamiento de datos ofrece una gran flexibilidad en cuanto a la demanda de máquinas virtuales o instancias, es una característica muy favorable dada las limitaciones de hardware con las que se cuentan, y por un precio razonable por hora se puede disponer de potentes instancias para la creación de proyectos haciendo uso de las tecnologías Google Cloud, pues el uso de GPU es una característica significativa a la hora de entrenar redes neuronales convolucionales (Geewax 2018, p. 509).

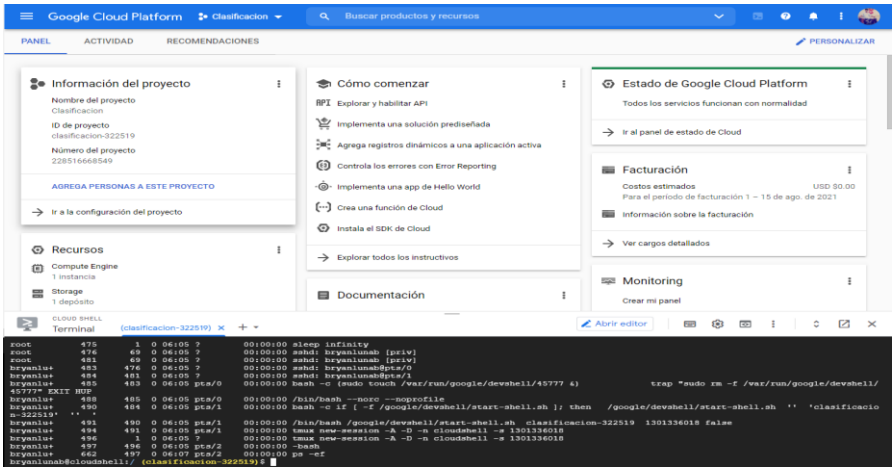


Figura 9-1: Consola y terminal Shell de Google Cloud.

Fuente: Google cloud ,2021, párr.1.

1.6.5. Streamlit

Es un marco de aplicación web que se enfoca en el desarrollo y construcción aplicaciones web mediante Python, que tiene la capacidad de compartir análisis y resultados , construyendo experiencias interactivas para modelos de ML, además posee una optimización la hora de cargar modelos de previamente entrenados en memoria cache (Richards, 2021, p.43).

1.6.6. Inteligencia artificial.

Se puede definir a. la *IA* como un campo de la ciencia e ingeniería que tiene como objetivo la comprensión computacional de un sistema inteligente y de la creación de algoritmos aplicados a dispositivos que muestren dicho comportamiento (Ramesh et al. 2004, p. 1).

La *IA* se lleva a cabo estudiando el pensamiento del cerebro humano, como se decide y como se trabaja mientras se intenta resolver un problema, para luego en base al estudio modelar o desarrollar un software y sistemas inteligentes, tomando como herramienta los diferentes sistemas y tecnologías informáticas (Tutorialspoint, 2016, párr.3).

Gracias a la recopilación de estudios o información realizados al día de hoy se tiene una basta fuente de datos que denominamos “Big Data” y el auge del Internet de las Cosas (IoT) ha desarrollado un entrono perfecto para que las nuevos servicios y aplicaciones basadas en IA sean viables en diferentes áreas como la conducción autónoma, seguridad pública, educación, medicina , entretenimiento entre muchas más áreas. Dentro de la IA tenemos dos ramas muy importantes como son el Machine Learning (ML) y Deep Learning (DL) (Tabassum 2020, p. 700).

1.6.7. Machine Learning

El Machine Learning o por la siglas ML es una rama de la IA nada nueva , puesto que muchos de los algoritmos propuestos en este campo ya vienen implementados desde hace décadas, en el ML los ordenadores son programados para aprender algo que por lo general no están programados gracias al aprendizaje de nuevos patrones y conocimientos a partir de datos (Tabassum 2020, p. 701). Esta área de la IA se ha venido desarrollando gracias tres puntos:

1.6.7.1. Disponibilidad de datos

Gracias a la inmensa cantidad de personas en el mundo que son aproximadamente 3000 millones conectadas a través de unos 17000 millones de dispositivos y sensores se ha podido recolectar datos para utilizarlos como datos de entrenamiento en los algoritmos de aprendizaje (Tabassum 2020, p. 701).

1.6.7.2. Poder de cómputo o potencia de cálculo

El desarrollo de las nuevas tecnologías en ordenadores es un punto a favor para el ML, ya que los ordenadores cada vez tienen más la capacidad de procesamiento de datos (Tabassum 2020, p. 701).

1.6.7.3. Innovación de Algoritmos

Las técnicas de aprendizaje automático también han venido en desarrollo a lo largo de los años desde la concepción del perceptrón creado por Frank Rosenblat hasta la evolución a las redes neuronales en capas (Tabassum 2020, p. 701).

1.6.8. Deep Learning

El Deep Learning (DL) es un campo de la IA que va mucho más allá que el ML, se puede denominar un método específico del machine Learning que incorpora redes neuronales de manera

sucesivas para aprender del input o entrada de datos de manera iterativa, ocupa un papel importante cuando se trata de aprender patrones de datos no estructurados. EL DL está diseñado para emular el funcionamiento de neuronas cerebrales para entrenar a ordenadores que puede abstraer patrones que sirven para resolver problemas definidos, sin embargo en contraste con el ML el uso de las redes neuronales dentro del Deep Learning es de manera jerárquicas combinando algoritmos supervisados y no supervisados por esto se puede denominar al DL como una subdisciplina del Machine Learning (Hurwitz y Kirsch 2018, p. 17).

1.6.9. Fundamentos matemáticos

1.6.9.1. Algebra lineal

El Algebra Lineal permite formalizar conceptos intuitivos para construir un conjunto de objetos y reglas para manipular dichos objetos denominados símbolos, el Algebra lineal se encarga del estudio de los vectores así como ciertos métodos y reglas para manipularlos , estos vectores geométricos se suelen denotar con letras con una pequeña flecha por encima de una letra para identificar dicho vector $\vec{x}$, $\vec{y}$, por lo general , los vectores son objetos matemáticos con los que se pueden realizar operaciones como sumas y multiplicaciones por escalares para producir otro del mismo tipo. Así desde un punto de vista abstracto matemático un vector es aquel objeto que cumple ya antes mencionadas propiedades (Peter et al. 2021, p. 17).

1.6.9.2. Probabilidad

La probabilidad se requiere al estudio de incertidumbre, en si es la fracción de veces que ocurre algún acontecimiento. Ya en el área matemática la probabilidad se encarga de describir los resultados aleatorios de los experimentos de razonamiento automatizado que se encarga de generalizar un razonamiento lógico, en ML es importante tener en cuenta la probabilidad dado que muchas veces cuantificamos la incertidumbre de los datos producidas por un modelo, es por eso que para cuantificar la incertidumbre requerimos la idea de una variable aleatoria (Peter et al. 2021, p. 172).

En si muchas de las ramas de la informática se encargan de resolver problemas totalmente determinísticos y seguros. Pero cuando tratamos con aprendizaje automático la teoría de probabilidad ocupa un papel muy importante ya que este debe tratar con cantidades inciertas y a veces se pueden tratar con cantidades estocásticas o llamadas también no determinísticas, esto gracias a que la incertidumbre y estocasidad surgen de muchas fuentes, por esta razón a lo largo

de los años diferentes investigaciones han presentado argumentos convincentes para cuantificar la incertidumbre utilizando la probabilidad (Goodfellow, Bengio y Courville 2016, p. 53).

1.6.9.3. Perceptrón

El perceptrón es un algoritmo clásico de aprendizaje a pesar de su simplicidad es un algoritmo que funciona sorprendentemente bien en algunos casos sobre todo cuando se trata de clasificación binaria (Peter et al. 2021, p. 392).

1.6.10. Redes Neuronales

Una neurona hablando a nivel computacional es la unidad básica de procesamiento de una red neuronal, en el cual se procesan la entrada de otros nodos generando así una salida de acuerdo con la función de transferencia hoy llamada también función de activación, qué representa un mapeo lineal o no lineal de la entrada de salida de una manera análoga, las sinapsis de variables se moldean mediante los pesos, que son perillas que dan relevancia a cada una de las conexiones hacia los nodos, cuyos valores deben ser óptimos para que el modelo se ajuste al valor real (Du 2014, p. 5).

$$\boldsymbol{neurona} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{C}} \boldsymbol{W_i X_i} + \boldsymbol{b}$$

Ecuación 1-1: Neurona

Fuente: Du, 2014, p. 5.

$$\boldsymbol{neurona} = (\boldsymbol{w_1 x_1} + \boldsymbol{w_2 x_2} + \boldsymbol{w_3 x_3} + \cdots + \boldsymbol{w_i x_i}) + \boldsymbol{b}$$

Ecuación 2-1: Suma ponderada de pesos y bias

Fuente: Du, 2014, p. 5.

Donde W_i hace referencia a cada uno de los pesos de acuerdo a sus conexiones X_i que son los valores de entrada y *b* hacer referencia al parámetro de *bias*, realizándose así una suma ponderada de cada uno de los pesos y entradas, cuya ponderación viene dada por los valores asignados a los pesos que afectarán a cada uno de los pesos (Du 2014, p. 5).

1.6.10.1. Función de Activación

La función de activación está presente en cada una de las neuronas de la red neuronal dado que se utiliza especialmente para transformar las señales de entrada en una señal de salida que a su

vez es utilizada como entrada para la siguiente capa de neuronas, las funciones de activación en su mayoría tienen como objetivo transformar la suma ponderada lineal y pesos a una función no lineal dado que en el mundo real los errores poseen características no lineales (Sharma y Athaiya 2020, p. 310).

1.6.10.2. Función ReLU

La función de Unidad Lineal Rectificada o por sus siglas ReLU es una función que provee una activación no lineal, utilizada ampliamente dentro de las redes neuronales, la ventaja presente en esta función es que provoca que las neuronas no se activen al mismo tiempo dado que la neurona sólo se activará cuando la salida de la transformación lineal es mayor que 0 (Sharma y Athaiya 2020, p. 312).

$$f(x) = \max(0, x) = \begin{cases} 0 \; para \; x < 0 \\ x \; para \; x \geq 0 \end{cases}$$

Ecuación 3-1: Función ReLU

Fuente: Sharma y Athaiya, 2020, p. 312.

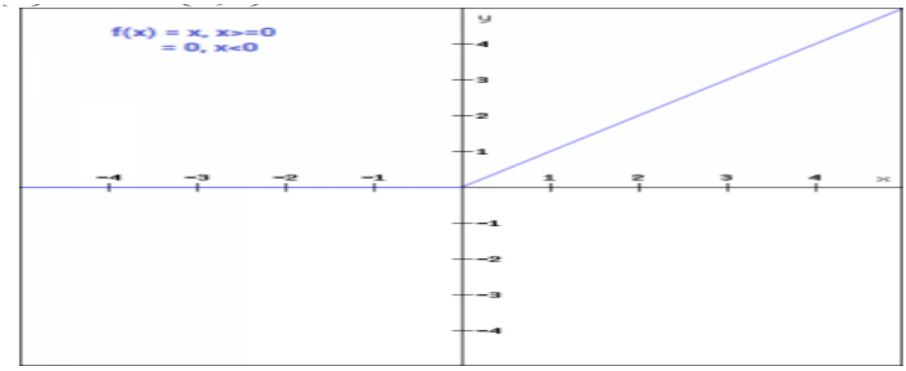


Figura 10-1: Función de activación ReLU.

Fuente: Sharma y Athaiya, 2020, p.312.

1.6.10.3. Función sofmax

La función *softmax* es utilizada para calcular la probabilidad a partir de números reales, esta función produce una salida de valores en un rango entre 0 y 1, con una suma de probabilidades igual a 1. Esta función es utilizada en su mayoría para salir de modelos de clase dado que revuelve las probabilidades de cada clase, así la clase que tiene mayor probabilidad es el resultado obtenido (Enyinna Nwankpa et al. 2018, p. 8).

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Ecuación 4-1: Función softmax

Fuente: Enyinna Nwankpa et al., 2018, p. 8.

1.6.10.4. Función de coste

La función de coste o también denominada *loss function*, $L(p, y)$ es una función que compara las predicciones de la red neuronal p con los valores reales o etiquetados y, esto para crear un valor o una medida que nos ayuden encontrar la distancia entre los valores reales y las predicciones realizadas es decir, para poder encontrar un error estimado de la red (Hennig y Kutlukaya 2007, p. 21).

1.6.10.5. Entropía Cruzada

También llamada *Cross-Entropy Loss Function* es una función de perdida da muy utilizada en la clasificación multiclase de imágenes ,dado que es una función de costo se utiliza para ajustar los pesos del modelo ya que se encarga de minimizar la función de perdida, también es conocida como el logaritmo de la función softmax (Andreieva y Shvai 2021, p. 7).

$$L_{CE} = -\sum_{n=1}^{k} g_n \log(S_n)$$

Ecuación 5-1: Entropía Cruzada

Fuente: Andreieva y Shvai, 2021, p. 7.

Donde g_i hace referencia a la etiqueta y S_i es la función de activación Softmax para la n enésima clase.

1.6.10.6. Regresión Lineal

La regresión lineal es un método científico que se usa muy frecuentemente, en las Ciencias biológicas físicas y sociales, así como en la empresa e ingeniería. Los modelos de revisión lineal son útiles en la planificación de investigaciones como en el análisis de datos resultantes y es uno de los pilares fundamentales en cuanto a redes neuronales se refiere (Rencher y Schaalje 2008, p. 1).
El objetivo de la regresión lineal simple es intentar modelar la relación entre 2 variables: una dependiente y una independiente para una relación lineal podemos utilizar la siguiente forma:

$$\mathbf{y = \beta_0 + \beta_1 x + \varepsilon}$$

Ecuación 6-1: Regresión Lineal

Fuente: Rencher y Schaalje, 2008, p. 1.

Donde y es la variable dependiente mientras que x es la variable independiente, seguramente se añaden otros supuestos en la distribución como el error que estado independientemente de los valores observados de y. Además utilizando los valores de x e y , se pueden estimar lo valores de β_0 y β_1 , para hacer inferencias como intervalos de confianza, dado que estos son parámetros del modelo que tienen influencia en los valores de x. Pero sin duda, uno de los usos más comunes para la regresión lineal es construir un modelo estimado para pronosticar o predecir el valor de y en función de x (Rencher y Schaalje 2008, p. 1).

1.6.10.7. Regresión Lineal y método de mínimos cuadrados

Dada la similitud en la regresión lineal a la solución del problema de mínimos cuadrados lineales, se puede hacer uso de esta o implementarla en los problemas de clasificación de los algoritmos de aprendizaje automático o Machine Learning, revisando la solución del problema de mínimos cuadrado en función de la regresión lineal simple en la cual si tomamos como referencia la ecuación donde se menciona a la relación lineal simple para encontrar los valores de β_0 y β_1 de manera que la distancia cuadrada acumulada de la respuesta real $\hat{y}$ se acerque al mínimo de todos los posibles valores de β_0 y β_1 donde:

$$\mathbf{(b_0, b_1) = \min_{(\beta_0,\beta_1)} \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 X_i)]^2}$$

Ecuación 7-1: Regresión Lineal y método de mínimos cuadrados

Fuente: Maulud y Abdulazeez, 2020, p. 141.

Dado que mediante el método de los mínimos cuadrados debemos encontrar las estimaciones de los parámetros utilizando el mínimo cuadrado, que es la línea más cercana de los puntos(x_i, y_i) (Maulud y Abdulazeez 2020, p. 141).

1.6.11. Descenso del gradiente

El método de descenso del gradiente o algoritmo del descenso de gradiente es un método de optimización para encontrar el mínimo de una función, este algoritmo es aplicado a la función de coste de las redes neuronales como se representa en la ec.8-1 (Peter et al. 2021, p. 227).

$$min_x = f(x)$$

Ecuación 8-1: Descenso del gradiente

Fuente: Peter et al., 2021, p. 227.

Donde $f: \mathbb{R}^d \rightarrow \mathbb{R}$ es una función objetivo que captura el problema de aprendizaje automático, esto solo si la función es diferenciable y nos podemos encontrar analíticamente una solución para la misma. Este algoritmo de primer orden sirve para usar un mínimo local en función del descenso de gradiente en los cuales se toman pasos profesionales al negativo de la gradiente de la función en un punto dado.

Como si se tratara de una pelota bajando una pendiente, el descenso de gradiente explota el hecho de que una función disminuye más rápido si se mueve desde un punto A en dirección a gradiente negativo hacia el punto B siempre y cuando se suma que esta función es diferenciable.

$$\boldsymbol{x_1 = x_0 - \gamma((\nabla f)(x_0))}$$

Ecuación 9-1: Descenso del Gradiente extendida

Fuente: Peter et al., 2021, p. 227.

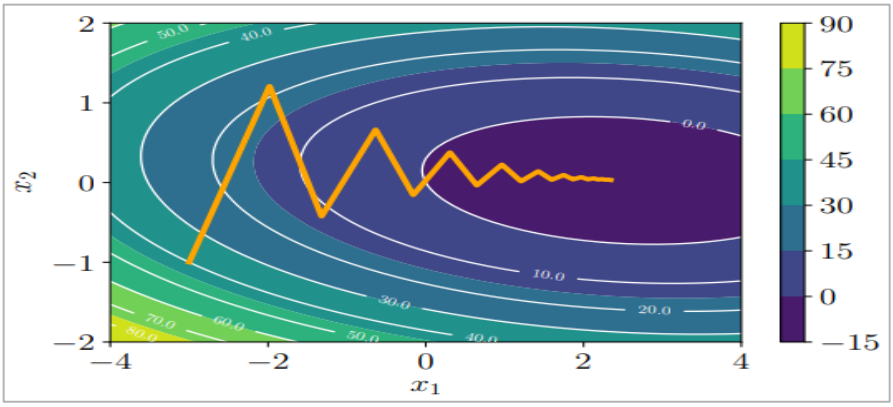


Figura 11-1: Gradiente descendiente bidimensional sobre una superficie

Fuente: Peter et al. 2021, p.228.

Donde el tamaño de paso es $\gamma \geq 0$, entonces $f(x_1) \leq f(x_0)$, así podemos proponer un sencillo algoritmo para encontrar el mínimo local $f(x_*)$ en una función $f: \mathbb{R}^n \to \mathbb{R}, x \mapsto f(x)$, así comenzamos proponiendo un x_0 y luego iteramos de acuerdo con la función.

$$\boldsymbol{x_{i+1} = x_0 - \gamma_i((\nabla f)(x_i))}$$

Ecuación 10-1: Descenso de la gradiente iterativa.

Fuente: Peter et al., 2021, p. 228.

Si proponemos un tamaño de paso o *learning rate* adecuado aseguraremos que la función se dirija hacia la convergencia, y este algoritmo es el responsable de disminuir la función de costo determinando los pesos indicados en la red neural (Peter et al. 2021, p. 229).

1.6.11.1. Paso o Learning rate

Este es uno de los parámetros más importantes en cuanto al descenso, este parámetro puede ser fijo en algunos casos, sin embargo, controlando dicho parámetro, se puede ajustar la rapidez a la que funciona el algoritmo teniendo en cuenta que si el en red es demasiado grande cuando nos acercamos a un mínimo local entonces este puede sobrepasarlo e ir más lejos de lo deseado.
Considerando la función:

$$\boldsymbol{f(x,y) = (\frac{3}{4}x - \frac{3}{2})^2 + (y-2)^2 + \frac{1}{4}xy}$$

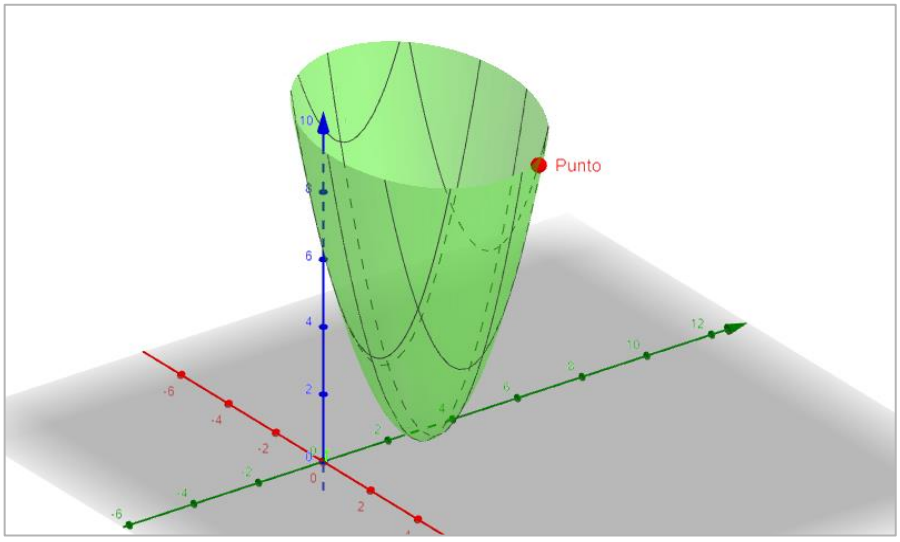


Figura 12-1: Función con punto (5,4,10).

Fuente: Luna Bryan, 2021.

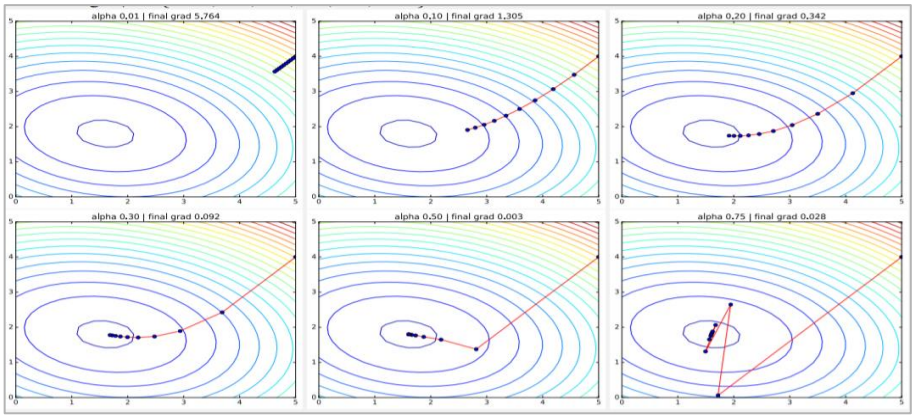


Figura 13-1: Gradientes con diferentes $\boldsymbol{\gamma}$

Fuente: Phillips, 2017, p.14.

Donde su gradiente es:

$$\nabla f(x, y) = \left(\frac{9}{8}x - \frac{9}{4} + \frac{1}{4}y\,, 2y - 4 + \frac{1}{4}x\right)$$

Tomaremos su gradiente descendiente desde la posición inicial (5,4), mientras se varía el Learning rate o paso $\gamma = [0.01, 0.1, 0.2, 0.3, 0.5, 0.75]$

Podemos darnos cuenta que en la figura 10-1 que, cuando el *Learning rate* tiene un valor pequeño este nos acerca al mínimo, al contrario cuando el *Learning rate* grande entonces el algoritmo se salta alrededor y corre riesgo de no converger (Phillips 2017, p. 14).

1.6.11.2. Stocasthic Gradient Descent

$$x_{i+1} = x_0 - \gamma_i((\nabla f)_i(x_0))$$

Ecuación 11-1: Gradiente Descendente Estocástico

Fuente: Peter et al., 2021, p. 229.

Mientras el descenso del gradiente realiza cálculos repetidos a un gran conjunto de datos como resultado, vuelve a calcular gradientes de ejemplos similares cada vez que se actualizan los parámetros, el descenso de gradiente estocástico (SDG, Stocasthic Gradient Descent por siglas en inglés) elimina esta redundancia, y como resultado suele ser mucho más rápido que el gradiente descendiente normal, en contraste con el gradiente descendiente el cual converge al mínimo de una cuenca la fluctuación de(SGD) permite saltar a nuevas potenciales cuencas o mínimos locales, sin embargo, esto dificulta la convergencia a un mínimo local exacto. El SGD muestra el mismo

comportamiento de convergencia que el exceso de gradiente por lotes cuando la tasa de aprendizaje es muy pequeña (Ruder 2016, p. 2).

1.6.11.3. Momentum

Cuando el SGD tiene problemas para determinar los mínimos en las zonas donde la superficie tienen curvas pronunciadas, el *momentum* es un método que ayuda SGD a encontrar una dirección adecuada amortiguando las oscilaciones para esto se añaden una fracción del valor η de acreditación del paso al tiempo del valor de producto actual. (Ruder 2016, p. 4)

$$x_{i+1} = \eta x_0 - \gamma_i((\nabla f)_i(x_0)) \quad (1)$$

Ecuación 12-1: Gradiente Descendente con momentum

Fuente: Ruder, 2016, p. 4.

Con el *momentum* adjudicamos un impulso como si se tratara de una pelota cuesta abajo todo que esta pelota como un impulso mientras baja haciéndose cada vez más rápido (Ruder 2016, p. 4).

1.6.11.4. Adam

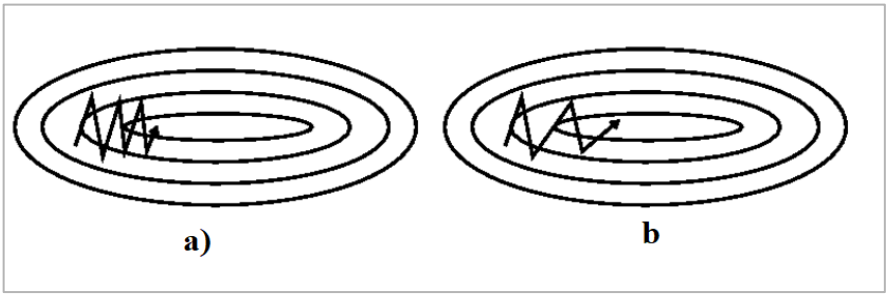


Figura 14-1: a).SGD sin momento b). SGD sin momento.

Fuente: Ruder, 2016, p. 4.

Adam es un método que se encarga de calcular las tasas de aprendizaje de manera adaptativa para cada uno de los parámetros, es un algoritmo de optimización basado en gradientes de primer orden con funciones o estocásticas, siendo este un método fácil de implementar y eficiente desde un punto de vista computacional puesto que tiene menos requisitos de memoria (Ruder 2016, p. 6).

1.6.12. Backpropagation

La técnica de *backpropagation* permite determinar el error de cada una de las capas aplicando derivadas parciales y la regla de la cadena debido que la salida de todo el compendio de funciones de cada una de las neuronas es precisamente el valor esperado con respeto al *ground true* o etiquetas.

$$y = f_k(f_{k-1}\left(f_{k-2}\left(f_{k-n}(\dots(f_1(x))\dots)\right)\right))$$

Ecuación 13-1: Backpropagation

Fuente: Ruder, 2016, p. 4.

Donde x son los valores de entrada, y son los resultados en los cuales intervienen cada una de las funciones $f_i = 1,2,\dots,k$ que poseen sus propios parámetros ya que estos parámetros dependerá específicamente de la función de activación que se elija y teniendo en cuenta que para modelos de múltiples capas necesitamos el gradiente de la función de perdida con respecto a cada uno de los parámetros, esto implicara que también necesitaremos los gradientes capa tras capa, obteniendo una composición de funciones aplicando derivadas parciales y regla de la cadena de las funciones de coste con respeto a cada uno de sus parámetros por capas (Peter et al. 2021, p. 160).

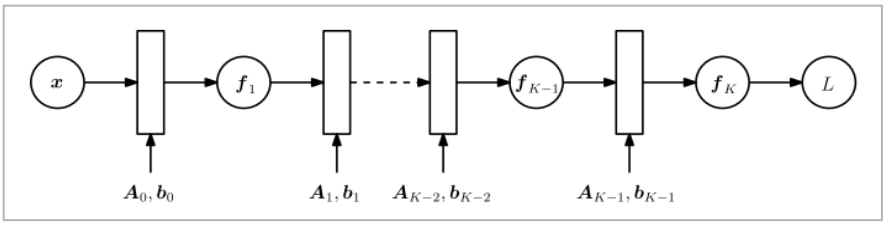


Figura 15-1: Diagrama de paso de una red multicapa.

Fuente: Peter et al. 2021, p 160

Considerando que θ se refiere a los parámetros A_i y b_i que se refieren a los pesos y bias respectivamente donde el valor de $i = 0,\dots,k-1$ es el índice de cada capa y L hace referencia a la función de costo.(Peter et al. 2021, p. 160)

Entonces mediante la regla de la cadena tenemos que:

.

$$\frac{\partial L}{\partial \theta_{K-1}} = \frac{\partial L}{\partial f_K}\frac{\partial f_K}{\partial \theta_{K-1}}$$

$$\frac{\partial L}{\partial \theta_{K-2}} = \frac{\partial L}{\partial f_K}\frac{\partial f_K}{\partial \theta_{K-1}}\frac{\partial f_{K-1}}{\partial \theta_{K-2}}$$

$$\frac{\partial L}{\partial \theta_{K-3}} = \frac{\partial L}{\partial f_K}\frac{\partial f_K}{\partial \theta_{K-1}}\frac{\partial f_{K-1}}{\partial \theta_{K-2}}\frac{\partial f_{K-2}}{\partial \theta_{K-3}}$$

$$\frac{\partial L}{\partial \theta_i} = \frac{\partial L}{\partial f_K}\frac{\partial f_K}{\partial \theta_{K-1}} \cdots \frac{\partial f_{i+2}}{\partial \theta_{i+1}}\frac{\partial f_{i+1}}{\partial \theta_i}$$

Ecuación 14-1: Regla de la Cadena del Algoritmo de backpropagation

Fuente: Peter et al., 2021, p. 161.

1.6.13. Conjunto de Datos

1.6.13.1. Train, Test y Validation

Es una buena práctica dividir los datos en 3 conjuntos de datos uno para el entrenamiento validación y prueba esto debido a que con el conjunto de datos de entrenamiento se entrena el algoritmo posterior a esto mediante el conjunto de datos de validación se realiza un ajuste de parámetros llegando a una etapa de, *fine-tunning* realizando los pasos anteriores de manera iterativa hasta llegar a un rendimiento óptimo y para finalizar se utilizan los datos de prueba evaluando nuevamente la red neuronal obteniendo un resultado final (Subramanian 2018, p. 73).

1.6.14. Capacidad Under and Over fitting

1.6.14.1. Underfitting

El underfitting se refiere a aquellas ocasiones cuando nuestro modelo no puede aprender ningún patrón de los datos de entrenamiento, lo que provoca que el modelo no funciona bien ni siquiera con el conjunto de datos con el que se ha entrenado una de las técnicas utilizadas para reducir el underfitting es el aumento de capas del modelo aumentando el número de pesos y parámetros. (Subramanian 2018, p. 73).

1.6.14.2. Overfiiting

Conocido también como sobreajuste es uno de los problemas más comunes en el ML y DP , cuando un modelo está en overfitting o también conocido sobreajuste el modelo ya no mejora su capacidad para resolver el problema, aunque se desempeña muy bien en el conjunto de datos de entrenamiento al evaluar el resultado de las métricas del conjunto de datos de validación no es

eficiente, eso se debe aquel modelo identifica patrones demasiado específicos del conjunto es decir el algoritmo memoriza el conjunto de datos para que funcione bien en el entrenamiento pero esto no ocurre en los datos no vistos como validación o de prueba (Subramanian 2018, p. 74).

1.6.14.3. Regularización

La regularización es un proceso que se da para evitar el overfitting a una red neuronal también conocido como tunning esta etapa es una de las más complicadas pesto que existen muchos parámetros para configurar.

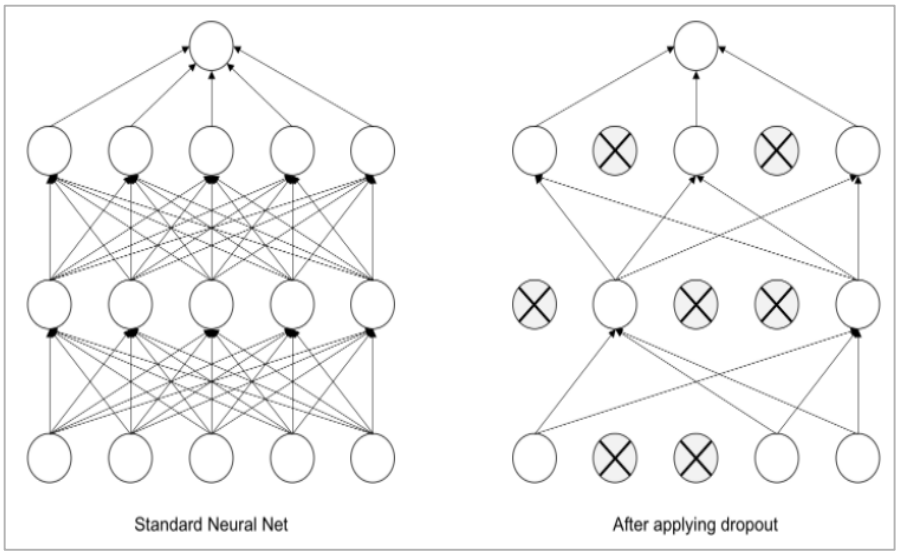


Figura 16-1: Dropout desactivación de neuronas en la red neuronal.

Fuente: Subramanian, 2018, p 76.

Entre los métodos usados para la regularización está el dropout, que es una técnica que hace que cierta cantidad de neuronas se "apaguen" de manera aleatoria para evitar un sobre entrenamiento.

.

1.6.15. Métricas

Las métricas permiten evaluar el resultado de la red neuronal una forma cuantitativa, para esto tenemos diferentes tipos de métricas entre las cuales como predecesora tendremos a la matriz de confusión de la cual se desprende la precisión, *recall* y F1-score (Ting 2010, p. 9).

1.6.15.1. Matriz de Confusión

La matriz de confusión es una herramienta útil que proporciona el rendimiento de la clasificación de manera cuantitativa de una red neuronal esta situación con respecto a los datos de prueba, esto permite observar de manera explícita cuando existe la confusión entre la clasificación de clases (Ting 2010, p. 10).

		Assigned Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

Figura 17-1: Matriz de confusión de una clasificación binaria.

Fuente: Ting, 2010, p 9

La matriz de confusión posee cuatro parámetros:

- Verdaderos positivos: Hace referencia a que la predicción es positiva y la etiqueta de la clase también lo es.
- Verdaderos negativos: hace referencia a que para predicción es negativa y la etiqueta también lo es
- Falsos negativos: Se cuando el valor real es positivo pero la predicción arroja un valor negativo.
- Falsos Positivos: Indica que el valor es real negativo, pero en la predicción resulta positivo.

1.6.15.2. Precision, Recall y F1-Score

Estas métricas se obtienen a partir de los parámetros de la matriz de confusión, siendo estos muy útiles a la hora de evaluar un modelo. La precisión se toma a partir del número de predicciones reales y positivas VP sobre los valores de verdaderos positivos VP mas falsos positivos FP, el *recall* se obtiene a partir de los valores de verdadero positivos VP sobre la suma de verdaderos positivos VP mas falsos negativos FN es decir, es la proporción de casos reales que son precedidos correctamente y el F1-score sea mediante 2 veces la precisión por el *recall* sobre la precisión más el *recall* e indica las combinaciones entre las métricas anteriores (Goutte y Gaussier 2005, p. 170).

$$Precision = \frac{VP}{VP + FP}$$

Ecuación 15-1: Precision

Fuente: Goutte y Gaussier, 2005, p. 170.

$$Recall = \frac{VP}{VP + FN}$$

Ecuación 16-1: Recall

Fuente: Goutte y Gaussier, 2005, p. 170.

$$F1 = 2\frac{Precision * Recall}{Precision + Recall}$$

Ecuación 17-1: F1 Score

Fuente: Goutte y Gaussier, 2005, p. 170.

1.6.16. Redes Neuronales Convolucionales

Las redes neuronales convolucionales se han hecho muy populares en el área de reconocimiento de imágenes y detección de objetos segmentación entre otras tareas en el campo de la visión por ordenador, todo esto gracias al reconocimiento de patrones y formas de la imagen aplicando capas *convolucionales*, capas de *pooling* y *fully coneccted* (Stevens, Antiga y Viehmann 2020, p. 200).

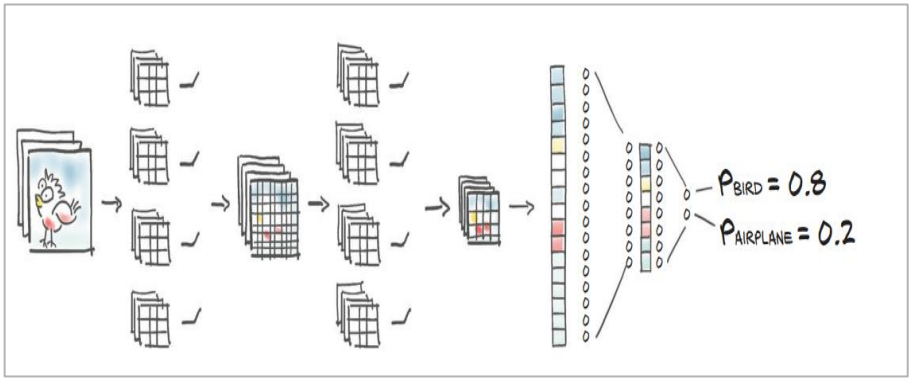


Figura 18-1: Diagrama de una red neuronal convolucional.

Fuente: Stevens, Antiga y Viehmann , 2020, p 206.

1.6.16.1. Convolución

La acumulación es una operación matemática de dos funciones audio esta operación es importante en la aplicación de procesamiento de señales en el campo de la infografía y en el procesamiento de imágenes se suelen trabajar con funciones discretas una convolución discreta en una imagen puede eliminar el río de una alta frecuencia afinar detalles detectar bordes o modular de otro modo del dominio de la frecuencia de imagen, para una imagen el proceso de convolución discreta de una imagen se da haciendo uso de un kernel discreto k (Novák, Liktor y Carsten Dachsbacher, 2019, párr. 4).

De esta forma:

$$(y * k)_{i,j} = \sum_n \sum_m y_{i-n,j-m}\, k_{n,m}$$

Ecuación 18-1: Convolución

Fuente: Novák, Liktor y Carsten Dachsbacher, 2019, párr. 4.

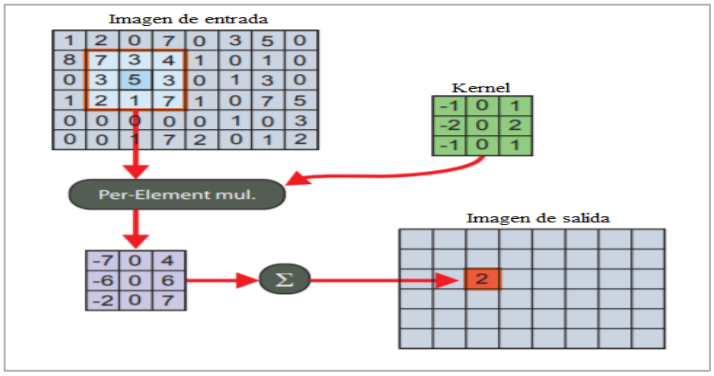


Figura 19-1: Convolución de una imagen con un filtro de 3x3.

Fuente: Novák, Liktor y Carsten Dachsbacher 2019, párr 4.

1.6.16.2. Max pooling

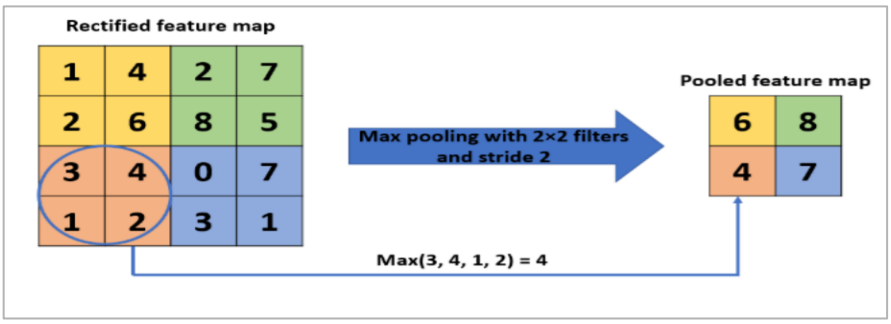


Figura 20-1: Ejemplo de Max-pooling.

Fuente: Christlein et al. 2019,p 2.

Las capas de Max pooling tiene como objetivo lograr la invariabilidad espacial reduciendo la resolución de los mapas de características o de las imágenes de una capa anterior.(Christlein et al. 2019, p. 2).

$$a_j = max_{N\times N}(a_i^{n\times n} u_{n,n})$$

Ecuación 19-1: Max Pooling

Fuente: Christlein et al., 2019, p. 2.

1.6.16.3. Strides

Es el paso que da la convolución en las imagen este parámetro es importante para manipular el tamaños de la imagen luego de la convolución. (Albawi, Mohammed y Al-Zawi 2018, p. 4)

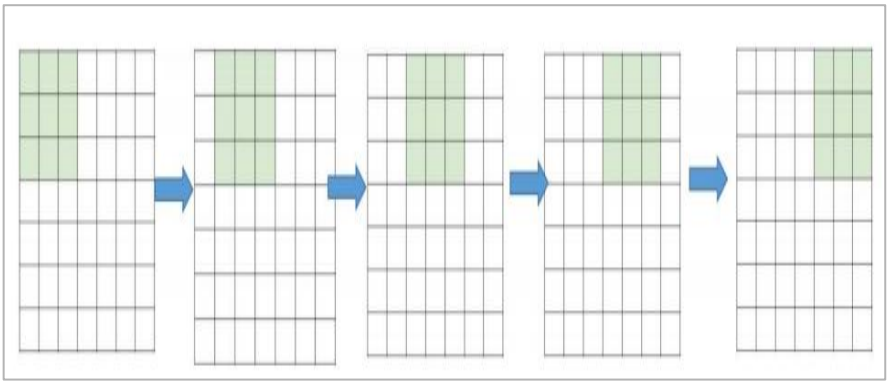


Figura 21-1: Stride de 1 paso.

Fuente: Albawi, Mohammed y Al-Zawi 2018, p 4.

1.6.16.4. Padding

El padding es un método que soluciona la perdida de información que puede existir en los bordes de la imagen , al momento de realizarse la convolución, es un método muy sencillo pero eficaz para solucionar este problema , esto se resuelve añadiendo filas y columnas de ceros en los bordes de la imagen, este es uno de los parámetros a tomar en cuenta para la salida de la imagen luego de la convolución (Albawi, Mohammed y Al-Zawi 2018, p. 4).

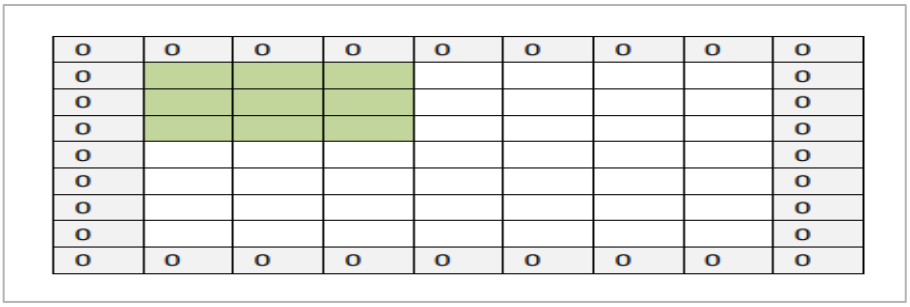


Figura 22-1: Padding o relleno de ceros alrededor de la imagen.

Fuente: Albawi, Mohammed y Al-Zawi 2018, p 4.

1.6.16.5. Fully connected

La capa de fully conected se encarga de aplanar matriz de imágenes 2D mediante haciendo un flaten para convertir las matrices a un vector de una sola dimensión 1D es de las capas finales que comprenden la mayor parte de los parámetros de la red teniendo un gran número de parámetros

entrenables son muy importantes debido a la necesidad de ajustar funciones no lineales que discriminan los errores (Basha, Dubey y Pulabaigari 2019, p. 5).

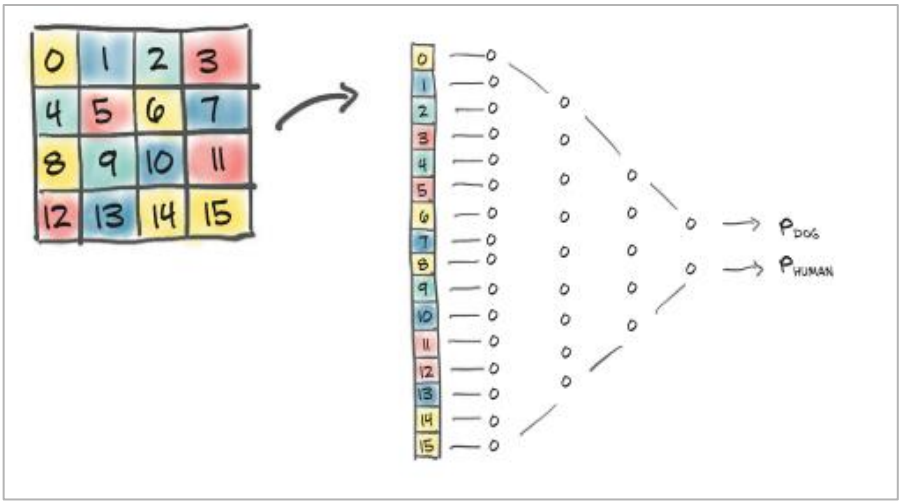


Figura 23-1: Capa densa aplicada a una transformación

Fuente: Stevens, Antiga y Viehmann, 2020, p 207.

1.6.17. Tranfer Learning

El transfer Learning es la técnica que hace uso de modelos previamente entenados, aprovechando los conocimientos posteriores en cuanto a los parámetros de la red se refieres como pesos *w* y bias *b* ,, entren las ventajas que esta técnica brinda está el rendimiento inicial conseguido de la red neuronal debido al conocimiento previamente adquirido, también está la reducción de la cantidad de tiempo que se tarda en aprender, puesto que debido a que mediante la existencia de parámetros *w* y por ultimo esto nos permite alcanzar una gran eficiencia en comparación a un modelo entrenado sin Transfer Learning (Olivas et al. 2010, p. 243).

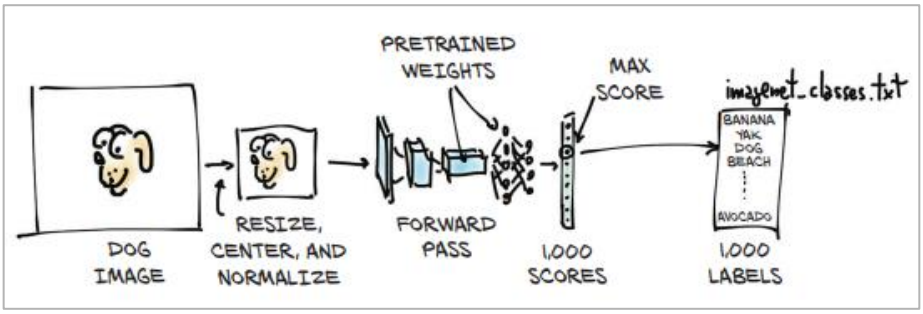


Figura 24-1: Diagrama de Transfer Learning en de una red neuronal convolucional.

Fuente: Stevens, Antiga y Viehmann 2020, p 422.

CAPÍTULO II

2. METODOLOGIA

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.

2.1. 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 de la provincia de Santo Domingo de los Tsáchilas, 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"*

Existe dos maneras para adquirir los datos del sistema PACS

a) 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.

b) 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.

2.2. 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 en el formato DCM.

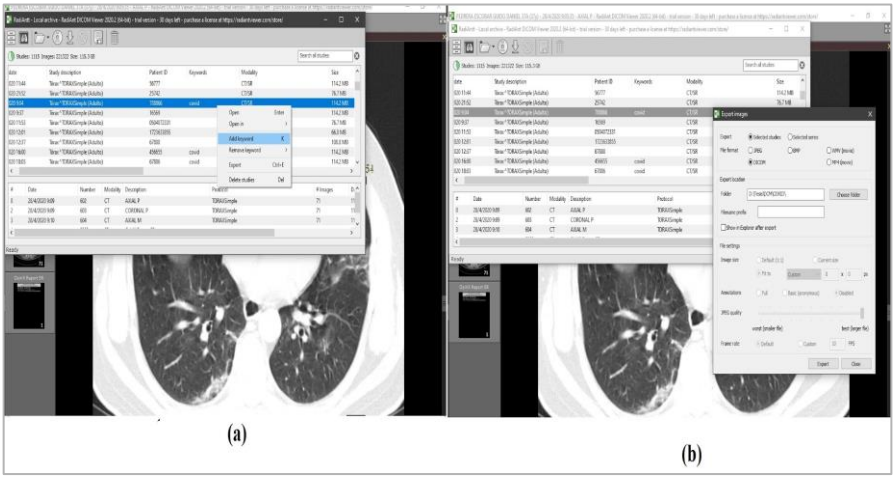


Figura 1-2: a) Colocación de etiqueta del estudio. b) Exportación de estudios.

Fuente: Radiant Dicom 2021.

2.3. 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".

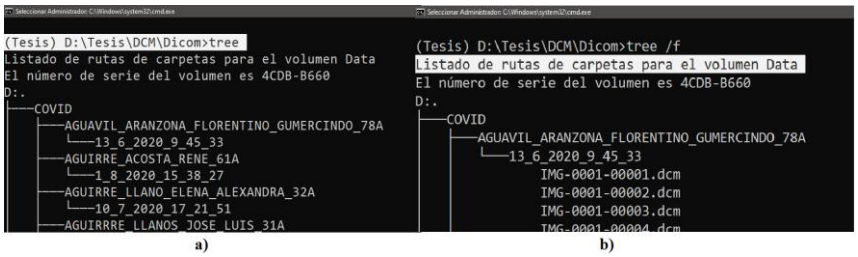


Figura 2-2: a) Listado de carpetas. b) Listado de archivos de imágenes médicas.

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 lista de rutas de los archivos DICOM y luego ser manipuladas.Para esto se lo guarda en una lista.

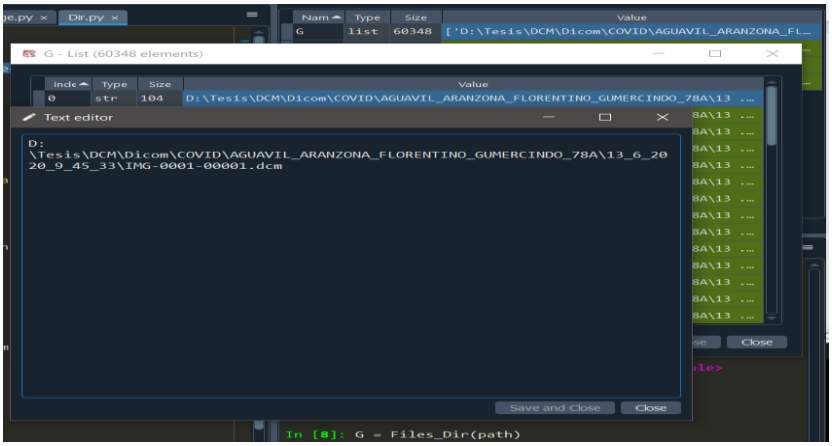


Figura 3-2: Resultado o salida de rutas de la función Files_Dir().

Fuente: Spyder 2021.

El algoritmo para la carga de lista de Subcarpetas y archivos se encuentra en el Anexo A

2.4. 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"*.

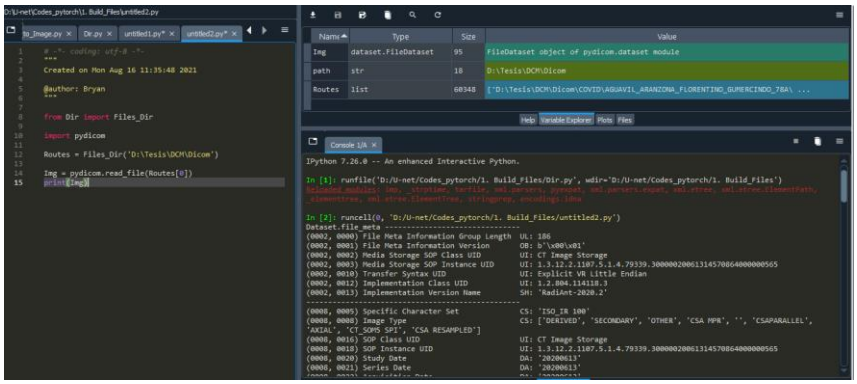


Figura 4-2: Lectura del archivo DICOM.

Fuente: Spyder 2021.

El algoritmo para leer datos DICOM se encuentra en el Anexo B

2.5. 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 valores inferior seran negros y el superiores serán blancos, para este cálculo se considera la siguiente formula:

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

Ecuación 1-2: Window max

Fuente: Ee. 2021,p 95.

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

Ecuación 2-2: Window min

Fuente: Ee. 2021,p 95.

$$y = X * m + b$$

Ecuación 3-2: Normalización

Fuente: Ee. 2021,p 96.

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.

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

Ecuación 4-2: Normalización escala Hounsfield

Fuente: Ee. 2021,p 95.

Si 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:

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

Ecuación 5-2: Normalización Png

Fuente: Ee. 2021,p 95.

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

El algoritmo de normalización a escala Hounsfield y *Arrays* de 8 bits se encuentra en el Anexo C.

2.6. 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.

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.

El algoritmo de creación de imágenes .png procesadas se encuentra en el Anexo D

2.7. Eliminación de imágenes médicas no necesarias

Tabla 1-2: Número de Archivos de Imagen.

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

Elaborado por: Luna, Bryan 2021.

2.8. 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.

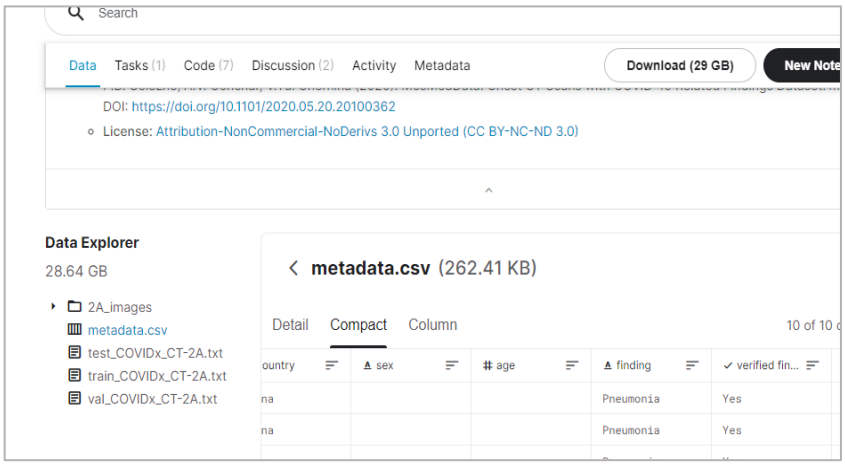
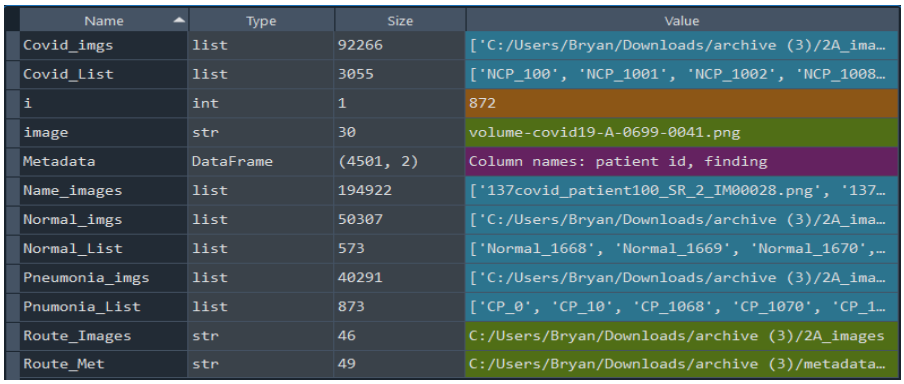


Figura 5-2: Dataset *COVIDx CT* -2 en Kaggle.

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 prosede a realizar un código en Python que ayude a mover los archivos a sus carpetas correspondientes según la *metadata.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…

Figura 6-2: Visualización de Imágenes a partir del archivo "csv".

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.

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 2-2: Número de imágenes de Kaggle

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

Elaborado por: Luna, Bryan 2021.

El algoritmo para la implementación del nuevo data set de Kaggle se encuentra en el Anexo E

2.9. 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 3-2: 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.

2.10. 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.

2.11. 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 4-2: 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

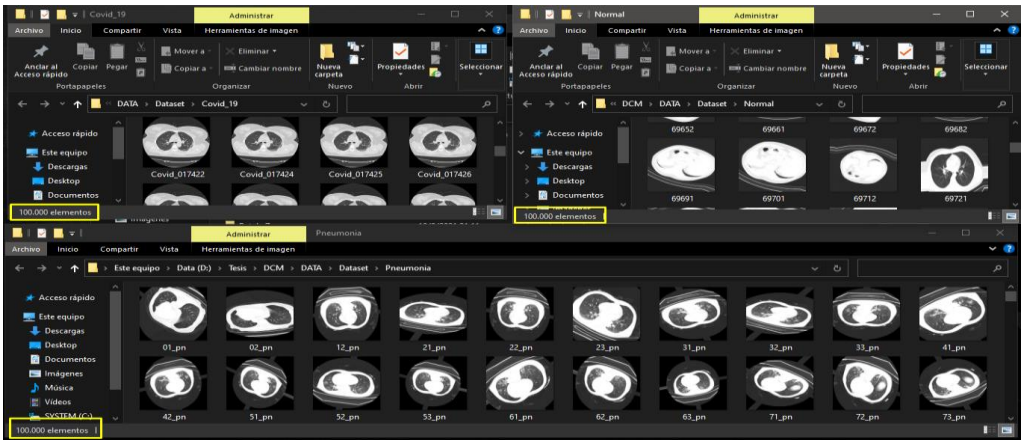


Figura 7-2: Carpetas para entrenamiento y validación: Covid_19, Normal y Pneumonia.

Fuente: Luna, Bryan 2021.

Algoritmo para el aumento de datos en el Anexo: F

2.12. 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.

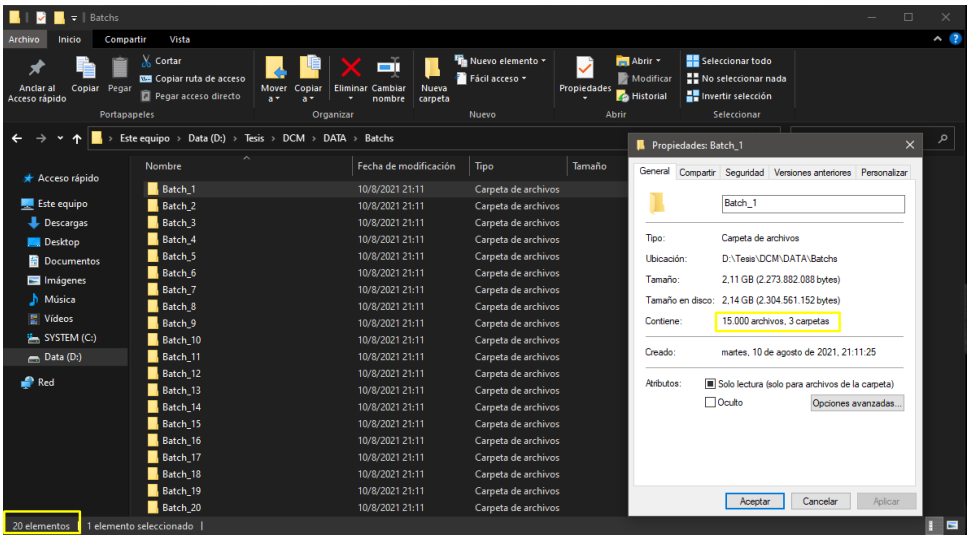


Figura 8-2: Carpetas con 15000 imágenes por batchs distribuidos en 3 carpetas.

Fuente: Luna Bryan, 2021

Algoritmo de creación de batchs o lotes y copiado aleatorio de datos se encuentra en el Anexo G

2.13. 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.

2.13.1. 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.

El algoritmo para tomar data set de entrenamiento se encuentra en el Anexo H

2.13.1.1. 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.

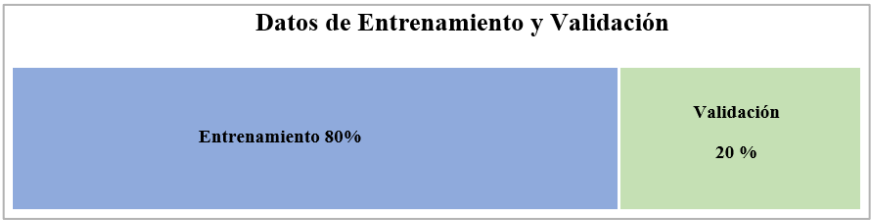


Figura 9-2: Split de datos para entrenamiento y Validación

Fuente: Luna Bryan, 2021

2.13.2. 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.

El algoritmo para tomar data set de validación se encuentra en el Anexo I

```
[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.
```

Figura 10-2: Datos de entrenamiento y Validación por batches

Fuente: Luna Bryan, 2021.

2.14. 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.

2.14.1. 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

2.14.2. 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

2.14.3. 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 5-2: 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.

2.14.4. 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.

2.14.5. 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.

2.14.5.1. 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.

2.14.6. 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.

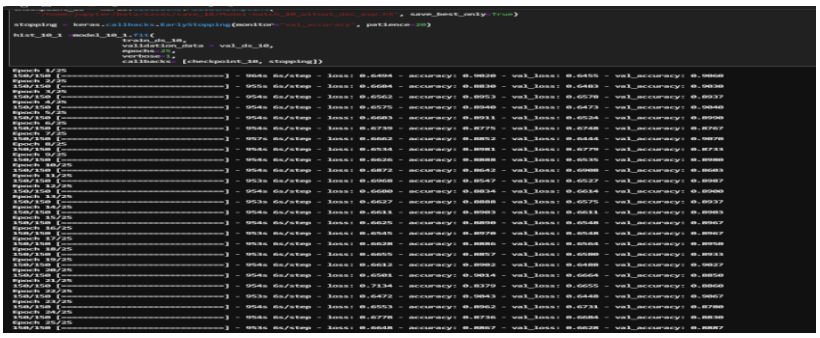


Figura 11-2: Progreso de entrenamiento de la red neuronal de un lote.

Fuente: Luna Bryan, 2021.

El Algoritmo de entrenamiento con su respectivo compilador en el Anexo J

2.14.7. 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 .

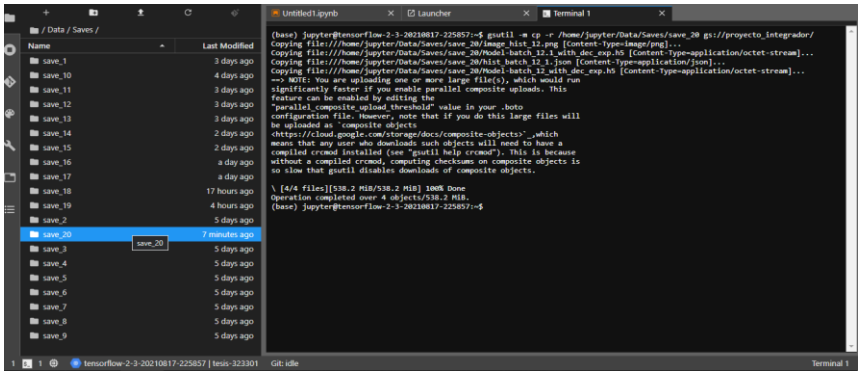


Figura 12-2: Copia de modelos entrenados al bucket de Google storage

Fuente: Luna Bryan, 2021.

2.15. 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.
El algoritmo de la interfaz se encuentra en el Anexo K.

CAPÍTULO III

3. MARCO DE ANÁLISIS E INTERPRETACIÓN DE RESULTADOS

3.1. Análisis de entrenamiento de Batchs

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

3.1.1. Loss y Acuraccy de todos los Batchs entrenados

En el gráfico 1-3 y 2-3 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 la 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.

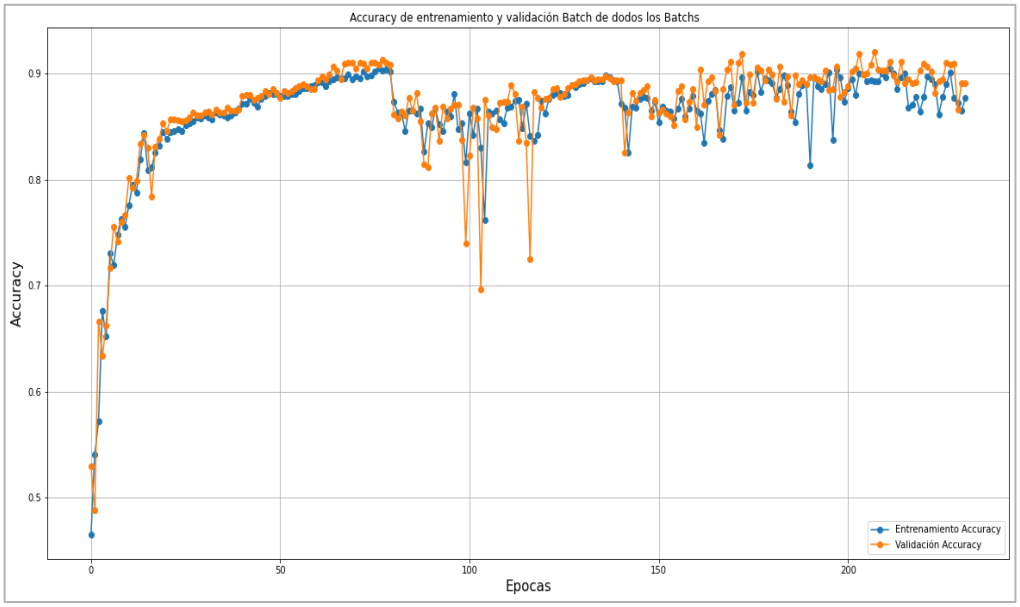


Grafico 1-3: Exactitud (Accuracy) de entrenamiento y validación los largo de 12 Batchs.

Fuente: Luna Bryan, 2021.

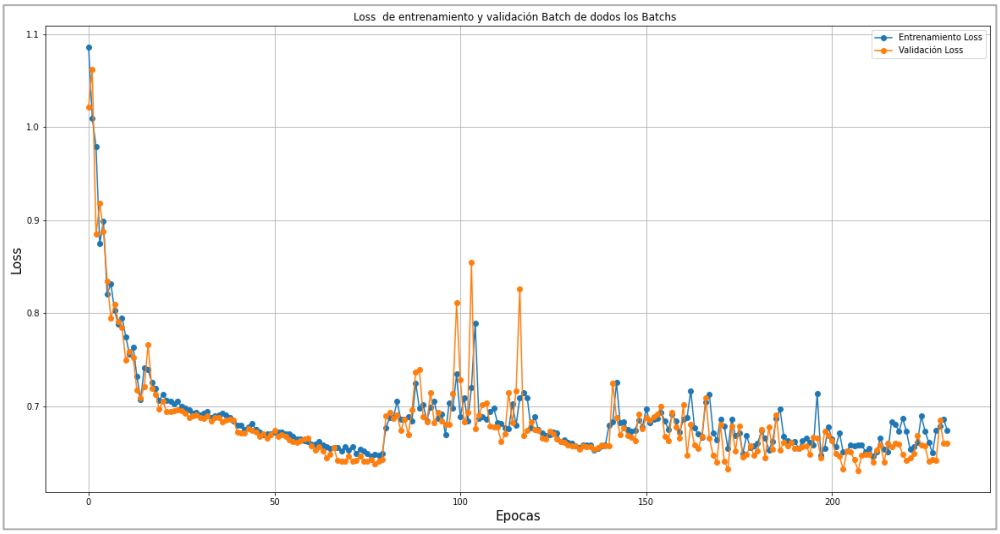


Grafico 2-3: Perdida (loss) entrenamiento y validación los largo de 12 Batchs.

Fuente: Luna Bryan, 2021.

Cada uno de los análisis de los lotes o batchs se encuentran en el Anexo B.

3.1.2. Decaimiento exponencial de la función de perdida

Se propone un decaimiento exponencial a los valores de *Learning rate el la figura 3-3*, 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

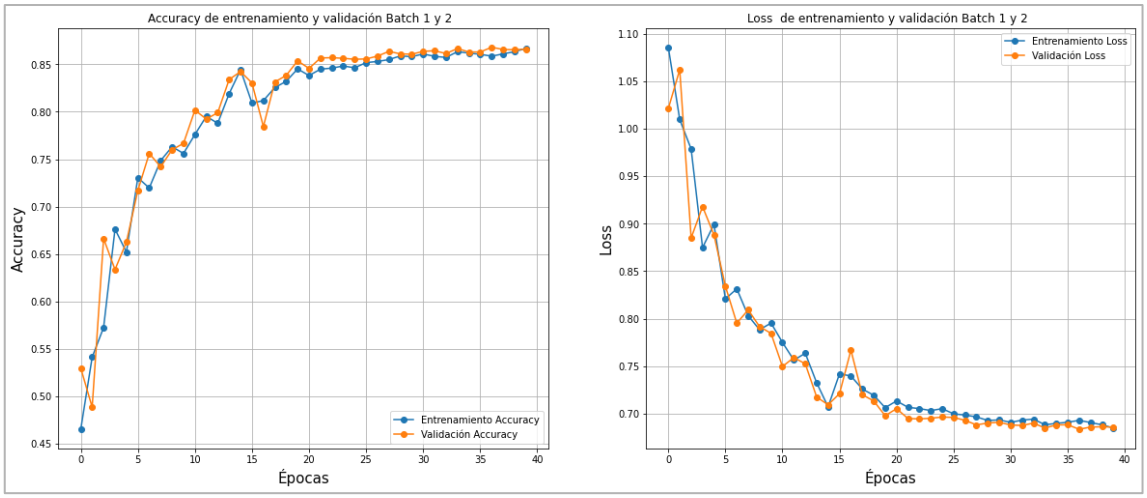


Grafico 3-3: Lote 1 sin decaimiento exponencial con lote 2 con decaimiento exponencial.

Realizado por: Luna, Bryan 2021.

3.1.3. Interpretación de underfitting

Debido al ajuste con el decaimiento exponencial figura 4-3, 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.

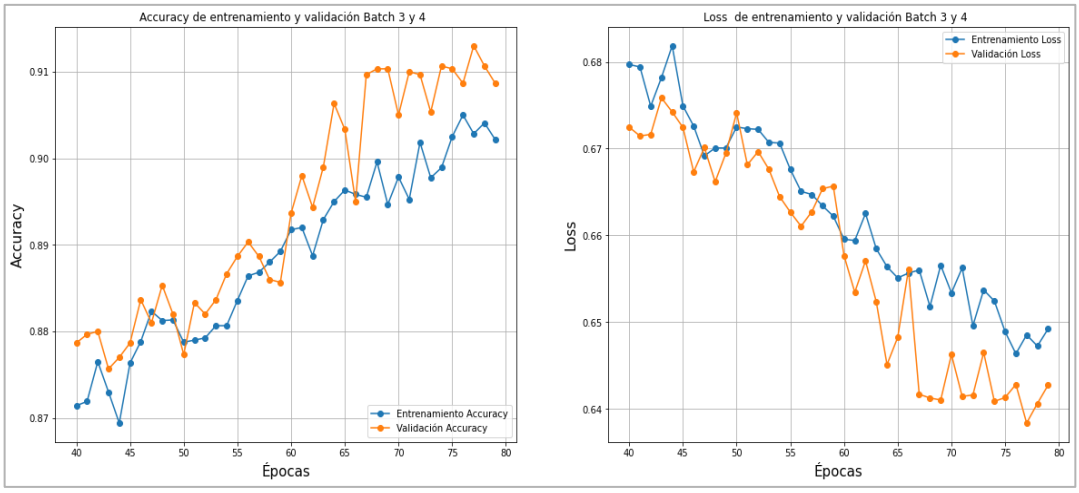


Grafico 4-3: Presencia de underfitting en el lote 3 y 4 debido a la aplicación del decaimiento exponencial.

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 la figura 5-3.

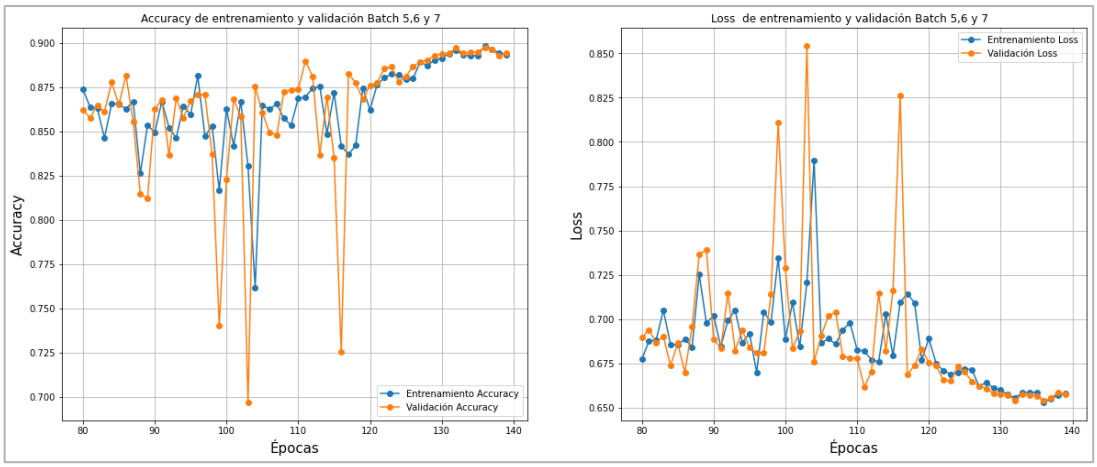


Grafico 5-3: Batch 5 y 6 sin decaimiento exponencial , Batch 7 con Decaimiento exponencial en el compilador.

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.

3.1.4. 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 6-3, 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 optimo.

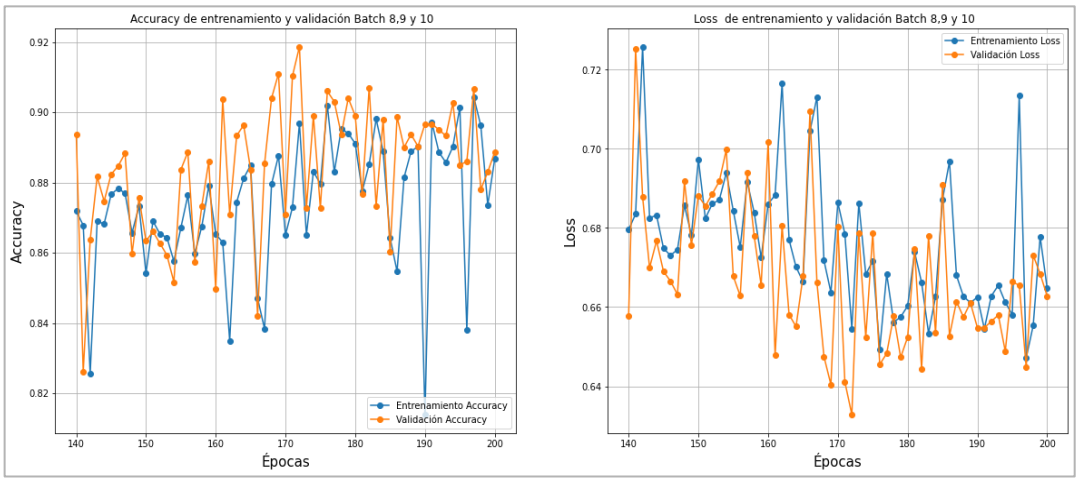


Grafico 6-3: . Batch 8,9 y 10 sin decaimiento exponencial.

Realizado por: Luna, Bryan 2021.

3.1.5. Decrecimiento de Accuracy y Aumento de Loss

La grafica 7-3 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.

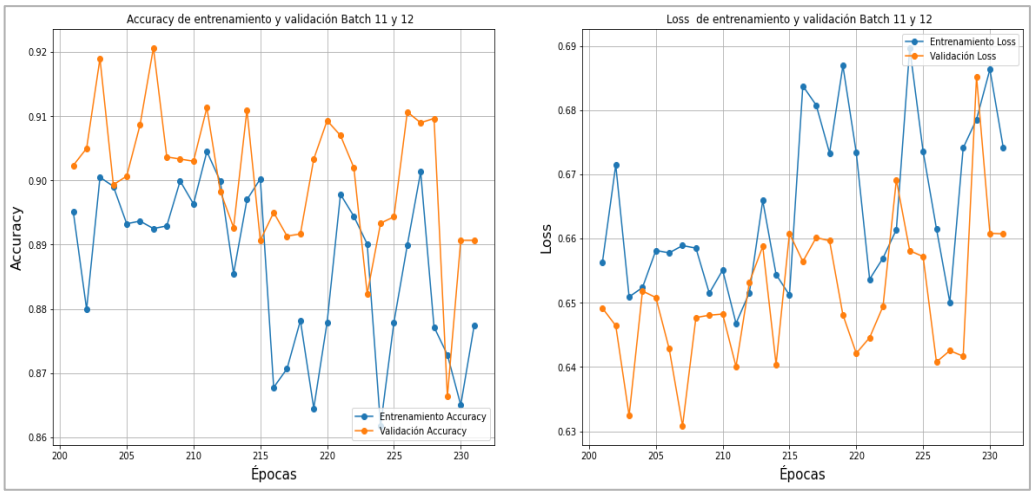


Grafico 7-3: Batch 11 y 12 decremento de accuracy y aumento de loss

Realizado por: Luna, Bryan 2021

3.2. 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 de cada uno de los modelos evaluando los falsos positivos y falsos negativos así como también las métricas : *precisión* , *recall*, *f1 score* y *Accuracy*.

Tabla 1-3: El Mejor modelos 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 1-3 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.

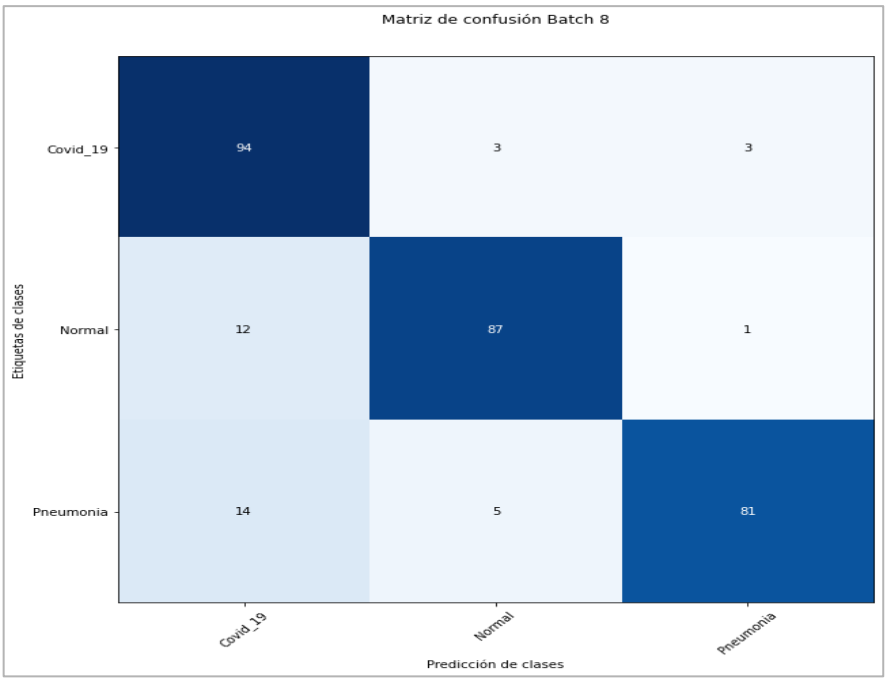


Grafico 8-3: Matriz de confusión del mejor modelo

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 que permita dar una interfaz gráfica al usuario para su mejor comodidad haciendo uso de Streamlit en Python

Tabla 2-3: 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.

Figura 1-3: Interfaz aplicación web.

Realizado por: Luna Bryan 2021.

En la figura 1-3 muestra la portada de la interfaz de la aplicación web se sistema de reconocimiento o clasificación de imágenes médicas

Figura 2-3: Interfaz aplicación web.

Realizado por: Luna Bryan 2021.

En la figura 2-3 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

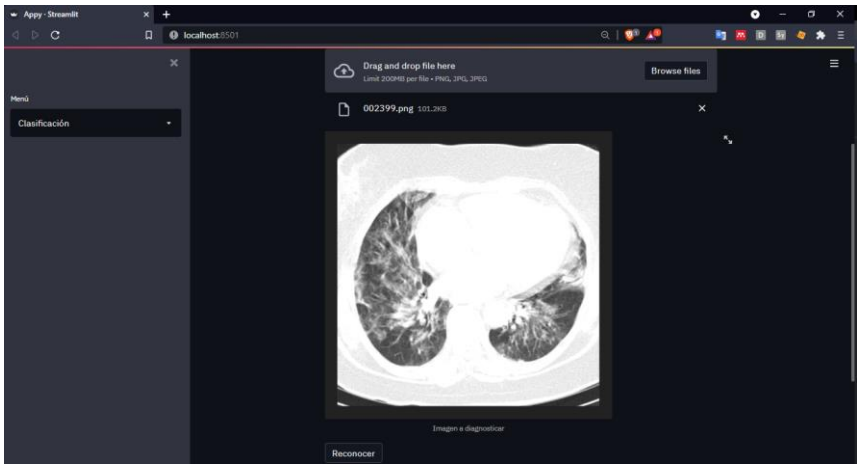


Figura 3-3: Carga de imagen en formato PNG.

Realizado por: Luna Bryan 2021.

En la figura 3-3 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

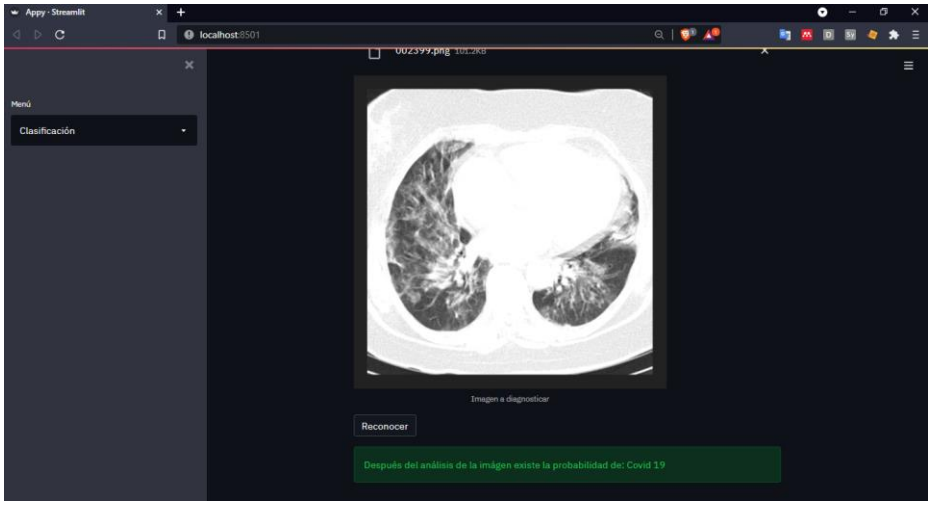


Figura 4-3: Reconocimiento de la imagen

Realizado por: Luna Bryan 2021.

En laa Figura 4-3 muestra el resultado final de la clasificación de imágenes medicas , este resultado muestra una eficacia del 88 % con una tasa de error del 12 %

CONCLUSIONES

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 adoptó 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.

RECOMENDACIONES

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.

GLOSARIO

Array: Conjunto de datos similares o de igual origen ubicados de una manera ordenada.(Avery y Kristensen, 2017, p.3.)

Tensor: Objeto de origen matemático que permite almacenar valores numéricos para su análisis y que puede tomar n dimensiones según el análisis correspondiente.(Georganas et al. 2021, parr 4)

GPU: Unidad de procesamiento grafico es una arquitectura multi.hilo que hace uso del paralelismo utilizada para cálculos gráficos y no gráficos.(Schwabe, 2014, p.57)

Machine Learning: También llamado Aprendizaje automático hace referencia a la capacidad de los sistemas para aprender partiendo de un conjunto de datos que automatizan procesos y construyen modelos analíticos para resolver tareas asociadas (Chahal y Gulia, 2019,p 4410).

Deep Learning: Concepto de aprendizaje automático basado en redes neuronales artificiales, que superan las capacidades de los modelos de aprendizaje automáticos con enfoques tradicionales de análisis de datos y poco profundos. (Chahal y Gulia, 2019,pp 4410).

Perceptrón: Es un algoritmo de clasificación lineal utilizado en el aprendizaje supervisado, formado redes neuronales(Goodfellow, Bengio y Courville 2016, p. 78).

Learning Rate: Hiperparametro que controla como puede cambiar un modelo de red neuronal en respuesta al error ajustando los pesos de la red.(Chahal y Guli, 2019, p 76).

Gradiente: Operador que se aplica a funciones vectoriales tridimensionales, útil para determinar los máximos u mínimos de dicha función.(Ruder, 2016, p 39).

BIBLIOGRAFÍA

ALBAWI, Saad, MOHAMMED, Tareq Abed y AL-ZAWI, Saad, Understanding of a convolutional neural network. *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, no. August, pp. 1-6. 2018. DOI 10.1109/ICEngTechnol.2017.8308186.

ANDREIEVA, Valeria y SHVAI, Nadiia, Generalization of Cross-Entropy Loss Function for Image Classification. *Mohyla Mathematical Journal* [en línea], vol. 3, pp. 3-10. 2021. DOI 10.18523/2617-7080320203-10. Disponible en: https://doi.org/10.18523/2617-7080320203-10.

ANGERAMI, Martín, Aplicaciones y beneficios de la TAC helicoidal y la reconstrucción 3D. *Universidad Nacional General San Martín Tecnicatura*, 2016.,pp. 2.

AVERY, James y KRISTENSEN, Mads, Array streaming for array programming. *International Journal of Computational Science and Engineering*, vol. 1, no. 1, pp. 1. 2017. ISSN 1742-7185. DOI 10.1504/ijcse.2017.10011354.

BASHA, S.H. Shabbeer, DUBEY, Shiv Ram y PULABAIGARI, Viswanath, *Impact of Fully Connected Layers on Performance of Convolutional Neural Networks for Image Classification*. ScienceDirect, vol. 378. 2019. DOI https://doi.org/10.1016/j.neucom.2019.10.008.

BORDILS, Francisco y CHAVARRÍA, Miguel, *Almacenamiento y transmisión de imágenes. PACS.* Monográfico: Radiología Digital, vol. 23, pp. 54-58. 2008.

CEDER, Naomi, *The quick Python book*. Greenwich2010. : Manning Publications Co. 2010. ISBN 9781935182207.

CHAHAL, Ayushi y GULIA, Preeti, *Machine learning and deep learning. International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 12, pp. 4910-4914. 2019. ISSN 22783075. DOI 10.35940/ijitee.L3550.1081219.

CHRISTLEIN, Vincent, SPRANGER, Lukas, SEURET, Mathias, NICOLAOU, Anguelos, KRÁL, Pavel y MAIER, Andreas, *Deep Generalized Max Pooling*. [en línea], 2019. [Consulta: 25 agosto 2021]. Disponible en: https://github.com/VChristlein/dgmp.

DAVIS, James, *The Python Book* [en línea]. California2019. : s.n. 2019. [Consulta: 12 agosto 2021]. Disponible en: https://www.softcover.io/downloads/92780ad5/python_book.

DU, Ke-lin, *Neural Networks and Statistical Learning*. London2014. : Springer. 2014. ISBN 9781447155706.

ENYINNA NWANKPA, Chigozie, IJOMAH, Winifred, GACHAGAN, Anthony y MARSHALL, Stephen, Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. *arXiv preprint arXiv:1811.03378.*, 2018.

FOS GUARINOS, Belén, *Diseño de técnicas de inteligencia artificial aplicada a imágenes médicas de rayos X para la detección de estructuras anatomicas de los pulmones y sus alteraciones*. [en línea], 2016. Disponible en: https://riunet.upv.es/bitstream/handle/10251/70103/73657675Z_TFG_14733814699245714722695301721227.pdf?sequence=2.

GEEWAX, JJ, *Google Cloud Platform in Action*. Shelter Island, NY 2018. : Manning Publications Co. 2018. ISBN 9781617293528.

GEORGANAS, Evangelos, KALAMKAR, Dhiraj, AVANCHA, Sasikanth, ADELMAN, Menachem, *Tensor processing primitives: A programming abstraction for efficiency and portability in deep learningworkloads*. S.l.2021. : Association for Computing Machinery. 2021. ISBN 9781450384421.

GHEBLAWI, Mahmoud, WANG, Kaiming, VIVEIROS, Anissa, NGUYEN, Quynh, ZHONG, Jiu Chang, *SARS-CoV-2 Receptor and Regulator of the Renin-Angiotensin System: Celebrating the 20th Anniversary of the Discovery of ACE2.* Circulation Research, pp. 1456-1474. 2020. ISSN 15244571. DOI 10.1161/CIRCRESAHA.120.317015.

GOODFELLOW, Ian, BENGIO, Yoshua y COURVILLE, Aaron, *Deep Learning*. S.l.2016. : MIT Press. 2016.

GOUTTE, Cyril y GAUSSIER, Eric, *A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. Lecture Notes in Computer Science* [en línea]. S.l.2005. : Springer Berlin Heidelberg, 2005. pp. 345-359. Disponible en: https://doi.org/10.1007/978-3-540-31865-1_25.

HENNIG, Christian y KUTLUKAYA, Mahmut, *Some Thoughts About The Design Of Loss Functions.* REVSTAT-STAT J, vol. 5, no. 1, pp. 19-39. 2007. ISSN 1645-6726.

HINOSTROZA, Luis, *Fisiopatología de la imagen en la infección por sars-cov-2.* Revista Peruana de Radiología [en línea], vol. 20, pp. 16-29. 2020. Disponible en: https://www.socpr.org.pe/revistas/REVISTA SOCPR EDICIÓN ESPECIAL.pdf.

HOPE, Tom, RESHEFF, Yehezkel S. y LIEDER, Itay, *Learning TensorFlow*. England2017. : O´Reilly. 2017. ISBN 9781491978511.

HURWITZ, Judith y KIRSCH, Daniel, *Machine Learning for Dummies* [en línea]. Hoboken2018. : John Wiley & Sons, Inc. 2018. [Consulta: 6 julio 2021]. ISBN 9781119454953. Disponible en: http://www.wiley.com/go/permissions.

IZQUIERDO, Jesús, *Caracterización de canal no lineal usando modelos de Volterra-Parafac* [en línea]. Sevilla2012. : Universidad de Sevilla. 2012. [Consulta: 12 agosto 2021]. Disponible en: http://bibing.us.es/proyectos/abreproy/12082/fichero/Capítulo3.pdf.

JUNG, Haijo, *Basic Physical Principles and Clinical Applications of Computed Tomography.* Progress in Medical Physics, vol. 32, no. 1, pp. 1-17. 2021. ISSN 2508-4445. DOI 10.14316/pmp.2021.32.1.1.

MARTÍNES, Javier Llamas, *Reconocimiento de Imágenes mediante Redes Neuronales Convolucionales.* Universidad Politécnica de Madrid: ETSI Sistemas Informáticos [en línea], vol. 1st ed, pp. 1 a 50. 2015. Disponible en: http://oa.upm.es/53050/1/TFG_JAVIER_MARTINEZ_LLAMAS.pdf.

MASON, D., ***SU-E-T-33:*** *Pydicom: An Open Source DICOM Library* [en línea]. 2011. S.l.2011. : s.n. 2011. Disponible en: https://aapm.onlinelibrary.wiley.com/doi/abs/10.1118/1.3611983.

MAULUD, Dastan y ABDULAZEEZ, Adnan M., *A Review on Linear Regression Comprehensive in Machine Learning.* Journal of Applied Science and Technology Trends, vol. 1, no. 4, pp. 140-147. 2020. DOI 10.38094/jastt1457.

MORENO, Artola, *Clasificación de imágenes usando redes neuronales convolucionales en Python.* Universidad de Sevilla, pp. 80. 2019.

NOVÁK, Dipl. Ing Jan, LIKTOR, Dipl. Inf Gábor y CARSTEN DACHSBACHER, Ing, *GPU Computing: Image Convolution.* Karlsruhe Institute of Technology, 2019.

OLIPHANT, Travis E., *Guide to NumPy* [en línea]. Massachusetts2006. : Massachusetts Institute of Technology. 2006. [Consulta: 13 agosto 2021]. Disponible en: http://www.trelgol.com.

OLIVAS, Emilio Soria, GUERRERO, José David Mart\'\in, MARTINEZ-SOBER, Marcelino, MAGDALENA-BENEDITO, Jose Rafael y LÓPEZ, Antonio José Serrano, *Handbook of Research on Machine Learning Applications and Trends* [en línea]. S.l.2010. : {IGI} Global. 2010. Disponible en: https://doi.org/10.4018/978-1-60566-766-9.

ORTIZ, Felipe Herrera, CORREO, Bosque-, CARLOS, Juan, CORREO, Colombia-, FRANCISCO, Marlio, CASTA, Motta, MAURICIO, Carlos, ALEXIS, Fredy, NI, M, *Patrones característicos de COVID-19 en tomografía de tórax: una revisión de la literatura.* Neuronum [en línea], vol. 6, no. 4, pp. 350-368. 2020. Disponible en: http://eduneuro.com/revista/index.php/revistaneuronum/article/view/298.

PETER, Marc, ALDO, Deisenroth A., CHENG, Faisal y ONG, Soon, *Mathematics For Machine Learning* [en línea]. S.l.2021. : Cambridge University Press. 2021. [Consulta: 6 julio 2021]. Disponible en: https://mml-book.com.

PHILLIPS, Dusty, *Python 3 Object Oriented Programming*. Birmingham2010. : Packt Publishing. 2010. ISBN 9781849511261.

PHILLIPS, Jeff M., *An Introduction to Data Analysis through a Geometric Lens*. S.l.2017. : utah.edu. 2017.

RAMESH, A.N., KAMBHAMPATI, C., MONSON, Jrt y DREW, P.J., *Artificial intelligence in medicine*. Ann R Coll Surg Engl, vol. 86, pp. 334-338. 2004. DOI 10.1308/147870804290.

RAMÍREZ GIRALDO, Juan Carlos, ARBOLEDA CLAVIJO, Carolina y MCCOLLOUGH, Cynthia, *TOMOGRAFÍA COMPUTARIZADA POR RAYOS X: FUNDAMENTOS Y ACTUALIDAD*. *Revista Ingeniería Biomédica*, vol. 2, no. 4, pp. 54-66. 2008. ISSN 1909-9991.

RENCHER, Alvin C. y SCHAALJE, G. Bruce, *Linear Model In Statistics* [en línea]. 2nd ed. S.l.2008. : John Wiley & Sons, Inc. 2008. [Consulta: 18 agosto 2021]. ISBN 9780471754985. Disponible en: www.wiley.com.

RICHARDS, Tyler, *Getting started with Streamlit for data science create streamlit applications from scratch*. Birmingham2021. : Packt Publishing. 2021. ISBN 9781800565500.

RUDER, Sebastian, An overview of gradient descent optimization algorithms. [en línea], pp. 1-14. 2016. Disponible en: http://arxiv.org/abs/1609.04747.

SANDE, Abhijeet y RAMDURG, Praveenkumar, Comparison Of Hounsfield Unit Of CT With Grey Scale Value Of CBCT For Hypo And Hyperdense Structure. *European Journal of Molecular & Clinical Medicine*, vol. 07, pp. 4654-4658. 2020. ISSN 2515-8260.

SCHWABE, Peter, Graphics processing units. *Secure Smart Embedded Devices, Platforms and Applications*, vol. 9781461479, no. 1, pp. 179-200. 2014. DOI 10.1007/978-1-4614-7915-4_8.

SENÉN, Victor, Diseño, implementación y evaluación de una red neuronal convolucional de regresión en clasificación de naranjas. [en línea], 2019. Disponible en: https://riunet.upv.es/bitstream/handle/10251/155034/20901959L_TFG_1593611105995799415 0375869586263.pdf?sequence=1&isAllowed=y.

SERNA, Walter y TRUJILLO, Juan, Descripción del estándar DICOM para un acceso confiable a la información de las imágenes médicas. , vol. 2, no. 45, pp. 289-294. 2010. DOI 10.22517/23447214.347.

SHARMA, Siddharth y ATHAIYA, Anidhya, Activation Funtions In Neural Networks. *International Journal of Engineering Applied Sciences and Technology* [en línea], vol. 4, pp. 310-316. 2020. [Consulta: 25 agosto 2021]. Disponible en: http://www.ijeast.com.

STEVENS, Eli, ANTIGA, Luca y VIEHMANN, Thomas, *Deep Learning with PyTorch*. Shelter Island, NY2020. : Manning Publications Co. 2020. ISBN 9781617295263.

SUBRAMANIAN, Sanjay, WANG, Lucy Lu, MEHTA, Sachin, BOGIN, Ben, VAN ZUYLEN, Madeleine, PARASA, Sravanthi, SINGH, Sameer, GARDNER, Matt y HAJISHIRZI, Hannaneh, MedICaT: A Dataset of Medical Images, Captions, and Textual References. *Findings of the Association for Computational Linguistics* [en línea], pp. 2112-2120.

2020. [Consulta: 3 septiembre 2021]. Disponible en: https://github.com/allenai/medicat.

SUBRAMANIAN, Vishnu, *Deep Learning with PyTorch*. S.l.2018. : Packt Publishing Ltd. 2018. ISBN 9781788624336.

TABASSUM, Lubna, Fundamentals of Artificial Intelligence and Deep Learning Techniques. , vol. 6, no. 4, pp. 700. 2020.

THAKURRATAN, Ranjit S., *Google Cloud Platform Administration Design*. Birmingham2018. : Packt Publishing. 2018. ISBN 9781788624350.

TING, Kai Ming, Confusion Matrix. En: C. SAMMUT y G.I. WEBB (eds.), *Encyclopedia of Machine Learning* [en línea]. Boston, MA2010. : Springer US, 2010. pp. 209. ISBN 978-0-387-30164-8. Disponible en: https://doi.org/10.1007/978-0-387-30164-8_157.

TUTORIALSPOINT, Artificial Intelligence Whit Python. [en línea]. S.l.2016. : [Consulta: 6 julio 2021]. Disponible en: https://www.tutorialspoint.com/artificial_intelligence_with_python/artificial_intelligence_with_python_tutorial.pdf.

USMANI, Zeeshan, 2016. What is Kaggle, Why I Participate, What is the Impact? | Data Science and Machine Learning. [en línea]. [Consulta: 3 septiembre 2021]. Disponible en: https://www.kaggle.com/getting-started/44916.

VARMA, Dandu Ravi, Managing DICOM images: Tips and tricks for the radiologist. *Indian Journal of Radiology and Imaging* [en línea], vol. 22, no. 01, pp. 4-13. 2012. ISSN 0971-3026. DOI 10.4103/0971-3026.95396. Disponible en: http://www.thieme-connect.de/DOI/DOI?10.4103/0971-3026.95396.

VIERA MAZA, Gabriela Isamar, *Procesamiento de imágenes usando OpenCV aplicado en Raspberry Pi para la clasificación del cacao* [en línea]. Piura2017. : Universidad de Piura. 2017. [Consulta: 3 septiembre 2021]. Disponible en: https://pirhua.udep.edu.pe/bitstream/handle/11042/2916/IME_218.pdf?sequence=1&isAllowed=y.

LEONARDO FABIO MEDINA NUSTE

Firmado digitalmente por LEONARDO FABIO MEDINA NUSTE
Nombre de reconocimiento (DN): c=EC, o=BANCO CENTRAL DEL ECUADOR, ou=ENTIDAD DE CERTIFICACION DE INFORMACION-ECIBCE, l=QUITO, serialNumber=0000621485, cn=LEONARDO FABIO MEDINA NUSTE
Fecha: 2022.01.31 15:53:49 -05'00'

ANEXOS

ANEXO A: ALGORITMO PARA LA CARGA DE LISTA DE SUBCARPETAS Y ARCHIVOS.

```
"""
Files_Dir(Path): devuelve  los últimos archivos de un directorio
de carpetas
Fol_Dir(Path): devuelve subcarpeta de nivel 2
"""
#Función que retorna las rutas de los archivos .dcm.
def Files_Dir(path):
  list_rut_files = []
  for root , dirs, files in os.walk(path):
    for names in files:
      list_rut_files.append(os.path.join(root,names))
  return list_rut_files
#Función que retorna las rutas de las subcarpetas que contienen
a los archivos .dcm
def Fol_Dir(path):
  A = os.listdir(path)
  List_of_Pads = []
  for root, dirs, files in os.walk(path):
    for names in dirs:
      List_of_Pads.append(os.path.join(root,names))
  Rute_of_Pads = List_of_Pads[len(A): ]
  return Rute_of_Pads
def main():

  Files_Dir(path)
  Fol_Dir(path)
if __name__ == "__main__":
  main()
```

ANEXO B: ALGORITMO PARA LEER DATOS DICOM

```
from Dir import Files_Dir
import pydicom
Routes = Files_Dir('D:\Tesis\DCM\Dicom')
Img = pydicom.read_file(Routes[0])
print(Img)
```

ANEXO C: ALGORITMO DE NORMALIZACIÓN A ESCALA HOUNSFIELD Y ARRAYS DE 8 BITS.

```
# importamos las librerías para trabajar con arrays e imágenes DICOM
import numpy as np
import pydicom as pd
Path = '...\IMG-0001-00001.dcm'
def Dicom2array(Path):
  Dcm_Img = pd.read_file(Path)
  rows = Dcm_Img.get(0x00280010).value
```

```
  cols = Dcm_Img.get(0x00280011).value
  Instance_Number = int(Dcm_Img.get(0x00200013).value)
  Window_Center = int(Dcm_Img.get(0x00281050).value[0])
  Window_Width = int(Dcm_Img.get(0x00281051).value[0])

  Window_Max = int(Window_Center + Window_Width / 2)
  Window_Min = int(Window_Center - Window_Width / 2)

  if (Dcm_Img.get(0x00281052) is None ):
    Rescale_Intercept = 0
  else:
    Rescale_Intercept = int(Dcm_Img.get(0x00281052).value)

  if (Dcm_Img.get(0x00281053) is None):
    Rescale_Slope = 1
  else:
    Rescale_Slope = int(Dcm_Img.get(0x00281053).value)

  New_Img = np.zeros((rows,cols), np.uint8)
  Pixels = Dcm_Img.pixel_array

  for i in range(0, rows):
    for j in range(0,cols):
      Pix_Val = Pixels[i][j]
      Rescale_Pix_Val = Pix_Val * Rescale_Slope + Rescale_Intercept

      if (Rescale_Pix_Val > Window_Max):
        New_Img[i][j] = 255
      elif (Rescale_Pix_Val < Window_Min):
        New_Img[i][j] = 0
      else:
        New_Img[i][j] = int(((Rescale_Pix_Val -
Window_Min)/(Window_Max - Window_Min)) * 255)

  return New_Img
def main():

  Dicom2array(Path)
if __name__ == "__main__":
  main()
```

ANEXO D: ALGORITMO DE CREACIÓN DE IMÁGENES .PNG PROCESADAS

```
#librerias
import cv2 as cv
import os
# modulos y previamente creados
import Dir
import Dicom_to_Image as DI
Path0 = "…\COVID"
Output_F = "…\Imagenes_png_Covid"

List_Files = Dir.Files_Dir(Path0)
List_Folders = Dir.Fol_Dir(Path0)

#Arrays = []
if os.path.isdir(Output_F) is False:
  os.mkdir(Output_F)
T = 0
for i in range(0, len(List_Files)):
```

```
  Arrays = (DI.Dicom2array(List_Files[i]))

  cv.imwrite( Output_F + '/'+ str(int(i)+1).zfill(6) +'.png', Arrays)

  T +=1
print(f"La tranformación esta completada :D se and creado {T} imagenes
.png \nen la ruta {Output_F}")
```

ANEXO E: ALGORITMO PARA LA IMPLEMENTACIÓN DEL NUEVO DATA SET DE KAGGLE:

```
import pandas as pd
#librerias para navegar en los datos
import os
#import pathlib
import shutil
# Rutas de imagenes y Metadatos
Route_Images = 'C:/Users/Bryan/Downloads/archive (3)/2A_images'
Route_Met = 'C:/Users/Bryan/Downloads/archive (3)/metadata.csv'

# lista del path de imagenes
Name_images = os.listdir(Route_Images)

#leer metadatos csv
Metadata = pd.read_csv(Route_Met)
# indetifica las fila de trastorno y el ID del paciente
Metadata = Metadata[['patient id','finding']]

# listas para guardar las ID del los sujetos con los posibles
transtornos
Pnumonia_List = []
Covid_List= []
Normal_List = []
# comparo listas de transtorno y guerdo en las listas anteriores
for i in range(len(Metadata)):
  if Metadata['finding'][i] == 'Normal' :
    Normal_List.append(Metadata.iloc[i][0])

  elif Metadata['finding'][i] == 'Pneumonia':
    Pnumonia_List.append(Metadata.iloc[i][0])

  elif Metadata['finding'][i] == 'COVID 19' :
    Covid_List.append(Metadata.iloc[i][0])

Ahora crearemos nuevas listas para alojar a cada una de las imágenes
correspondientes al sujeto y a su trastorno
""" Imágenes Normales """
Normal_imgs = []
for image in Name_images:
  for i in range(len(Normal_List)):
    if image.startswith(Normal_List[i], 0, len(Normal_List[i])) is
True:
      Normal_imgs.append(Route_Images+'/'+image)
Normal_imgs = list(set(Normal_imgs))
Normal_imgs.sort()

Y crearemos las carpetas correspondientes a sus trastornos: Normal,
Covid_19 y Pneumonia
Normal_Folder = '/run/media/bdluna/Data/Tesis/DCM/Normal'# ruta para
guardar imágenes Normales
```

```
if os.path.isdir(Normal_Folder) is False:
  os.mkdir(Normal_Folder)
  for i in range(len(Normal_imgs)):
    shutil.copy2(Normal_imgs[i],
Normal_Folder+'/'+'Normal_'+str(i).zfill(6)+'.png')

""" Imágenes con Covid """

Covid_imgs = []
for image in Name_images:
  for i in range(len(Covid_List)):
    if image.startswith(Covid_List[i], 0, len(Covid_List[i])) is True:
      Covid_imgs.append(Route_Images+'/'+image)
Covid_imgs = list(set(Covid_imgs))
Covid_imgs.sort()

Covid_Folder = '/run/media/bdluna/Data/Tesis/DCM/Covid_19'# carpeta
para img de ovid
if os.path.isdir(Covid_Folder) is False:
  os.mkdir(Covid_Folder)
  for i in range(len(Covid_imgs)):
    shutil.copy2(Covid_imgs[i], Covid_Folder
+'/'+'Covid_'+str(i).zfill(6)+'.png')

""" Imágenes Con Pneumonia """
Pneumonia_imgs = []
for image in Name_images:
  for i in range(len(Pnumonia_List)):
    if image.startswith(Pnumonia_List[i], 0, len(Pnumonia_List[i])) is
True:
      Pneumonia_imgs.append(Route_Images+'/'+image)
Pneumonia_imgs = list(set(Pneumonia_imgs))
Pneumonia_imgs.sort()

Pneumonia_Folder = '/run/media/bdluna/Data/Tesis/DCM/Pneumonia' # ruta
de pnuemonias
if os.path.isdir(Pneumonia_Folder) is False:
  os.mkdir(Pneumonia_Folder)
  for i in range(len(Pneumonia_imgs)):
    shutil.copy2(Pneumonia_imgs[i],
Pneumonia_Folder+'/'+'Pneumonia_'+str(i).zfill(6)+'.png')
```

ANEXO F: ALGORITMO PARA EL AUMENTO DE DATOS

```
import os
import numpy as np
import cv2
from keras.preprocessing.image import ImageDataGenerator, load_img,
image, img_to_array

DG_folder='Imagenes nuevas'
images_increased = 5

try:
  os.mkdir(DG_folder)
except:
  print("")

train_datagen = ImageDataGenerator(
  rotation_range=20,
```

```
  zoom_range=0.2,
  width_shift_range=0.1,
  height_shift_range=0.1,
  horizontal_flip=True,
  vertical_flip=False)


data_path = "D:\Tesis\DCM\DATA\Dataset\Covid_19"
data_dir_list = os.listdir(data_path)


width_shape, height_shape = 512,512

i=0
num_images=0
for image_file in data_dir_list:
  img_list=os.listdir(data_path)

  img_path = data_path + '/'+ image_file

  imge=load_img(img_path)

  imge=cv2.resize(image.img_to_array(imge), (width_shape,
height_shape), interpolation = cv2.INTER_AREA)
  x= imge/255
  x=np.expand_dims(x,axis=0)
  t=1
  for output_batch in train_datagen.flow(x,batch_size=1):
    a=image.img_to_array(output_batch[0])
    imagen=output_batch[0,:,:]*255
    imgfinal = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)
    cv2.imwrite(DG_folder+"/%i%i.png"%(i,t), imgfinal)
    t+=1

    num_images+=1
    if t>images_increased:
      break
  i+=1

print("images generated",num_images)
```

ANEXO G: ALGORITMO DE CREACIÓN DE BATCHS O LOTES Y COPIADO ALEATORIO DE DATOS :

```
import os
import shutil
from random import sample
from distutils.dir_util import copy_tree
DIRECTORIO_ORIGEN = "D:/Tesis/DCM/DATA/Upload"
DIRECTORIO_DESTINO = "D:/Tesis/DCM/DATA/Upload2"
# creo copia de archivos por si acaso
"""
print("Copiando...")
copy_tree(DIRECTORIO_ORIGEN, DIRECTORIO_DESTINO)
print("Copiado")

"""
# creo carpetas para diferentes batches
ruta = 'D:/Tesis/DCM/DATA/Batch'
rutas = []
for i in range(20):
```

```
  rutas.append( ruta +'_'+ str(i+1))
folder = ['Covid_19', 'Normal', 'Pneumonia']
for t in rutas:
  try:
    os.mkdir(t )
    for i in folder:
      os.mkdir(t+'/'+i)

  except:
    print("")

Covid_19 = 'D:/Tesis/DCM/DATA/Upload2/Covid_19'
Normal = 'D:/Tesis/DCM/DATA/Upload2/Normal'
Pneumonia = 'D:/Tesis/DCM/DATA/Upload2/Pneumonia'

for t in range(20):#range(len(rutas)):
  while len(os.listdir(rutas[t] +'/'+'Covid_19'))<5000:
    datos = os.listdir(Covid_19)
    datos = sample(datos, 5000)
    for i in datos:
      shutil.move(Covid_19 +'/'+i, rutas[t] +'/'+'Covid_19'+'/'+ i)

for t in range(20):#range(len(rutas)):
  while len(os.listdir(rutas[t] +'/'+'Normal'))<5000:
    datos = os.listdir(Normal)
    datos = sample(datos, 5000)
    for i in datos:
      shutil.move(Normal +'/'+i, rutas[t] +'/'+'Normal'+'/'+ i)

for t in range(20):#range(len(rutas)):
  while len(os.listdir(rutas[t] +'/'+'Pneumonia'))<5000:
    datos = os.listdir(Pneumonia)
    datos = sample(datos, 5000)
    for i in datos:
      shutil.move(Pneumonia +'/'+i, rutas[t] +'/'+'Pneumonia'+'/'+ i)

print(f'trabajo terminado, se han creado 20 batches de 5k de imágenes
en sus correspondientes carpetas {folder}')
```

ANEXO H: ALGORITMO PARA TOMAR DATA SET DE ENTRENAMIENTO

```
# Especificamos tamaño de imagen y de batch Size

(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)
```

ANEXO I: ALGORITMO PARA TOMAR DATA SET DE VALIDACIÓN

```
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)
```

ANEXO J: ALGORITMO DE ENTRENAMIENTO CON SU RESPECTIVO COMPILADOR

```
import tensorflow.keras as keras
#carga el modelo
model_1 = tf.keras.models.load_model(
  '/home/jupyter/Data/Saves/save_14/Model-batch_1.h5', compile=False)
initial_learning_rate = 0.0001
# Decaimiento exponencial
lr_schedule =
keras.optimizers.schedules.ExponentialDecay(initial_learning_rate,
                              decay_steps=100000,
                              decay_rate=0.96,
                              staircase=True)
# compila el modelo
model_1.compile(optimizer=tf.keras.optimizers.Adam(learning_rate =
lr_schedule),

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
        metrics=['accuracy'])
# Definimos una carpeta para ir salvando el mejor modelo por batch
checkpoint_1 = keras.callbacks.ModelCheckpoint(
  "/home/jupyter/Data/Saves/save_1/Model-batch_1.h5",
save_best_only=True)
# Se Define un Early Stopping
stopping = keras.callbacks.EarlyStopping(monitor="val_accuracy",
patience=10)

# Se entrena en Modelo
hist_1 =model_1.fit(
          train_ds_1,
          validation_data = val_ds_1,
          epochs=20,
          verbose=1,
          callback)
```

ANEXO K: ALGORITMO DE CREACIÓN DE INTERFAZ GRAFICA

```
# -*- coding: utf-8 -*-
"""
Created on Sun Aug 29 23:01:51 2021

@author: Bryan Luna
"""

import streamlit as st
from PIL import Image
import numpy as np
import os
# usa la CPU
os.environ['CUDA_VISIBLE_DEVICES'] = '-1'
import tensorflow as tf
import tensorflow.keras as keras
from skimage.transform import resize


MODEL_PATH = 'D:/Tesis/DCM/DATA/Saves/save_17/Model-
batch_9_witout_dec_exp.h5'
```

```
@st.cache(allow_output_mutation=True)
def cargar_modelo(MODEL_PATH):
  model0 = keras.models.load_model(MODEL_PATH)
  return model0
model0 = cargar_modelo(MODEL_PATH)


width_shape = 512
height_shape = 512

def model_prediction(image, model):
  #img_height, img_width = 512,512
  #img = keras.preprocessing.image.load_img(image,
target_size=(img_height, img_width))
  names= ['Covid 19', 'Normal', 'Pnuemonia']
  img_array = keras.preprocessing.image.img_to_array(image)
  img_array = tf.expand_dims(img_array, 0)
  predictions = model.predict(img_array)
  score = tf.nn.softmax(predictions[0])
  value = names[np.argmax(score)]
  print("probablmente implique {} ".format(names[np.argmax(score)]))
  return value

def main():

  menu = ["Portada","Clasificación"]#,"Segmentacion","Dicomtopng"]
  choice = st.sidebar.selectbox("Menú",menu)
  #foto = 'D:/Tesis/DCM/DATA/UPload2/Covid_19/segmentacion/002822.png'
  #img_height, img_width = 512,512
  #foto = 'D:/Tesis/DCM/Prueba/Covid_19/002375.png'#

  if choice == "Portada":

    caratula = Image.open('D:/Proyectos/Entornos/App/AHORA SI.png')
    st.image(caratula, width=700)

    hide = """
        <h1
        style='text-align: center; color:#FFFFFF;'>Escuela Superior
Politécnica del Chimborazo
        </h1>

        <h2
        style='text-align: center; color:#FFFFFF;'>Proyecto de
Investigación
        </h2>
        <h3
        style='text-align: center; color:#FFFFFF;'>Bryan Darwin Luna
Bravo
        </h3>

        <h4
        style='text-align: justify;font-size:25px;
color:#FFFFFF;'>RECONOCIMIENTO DE LA PRESENCIA DE SARS-COV-2 EN
PULMONES A TRAVÉS DE IMÁGENES DE RADIODIAGNÓSTICO HACIENDO USO DE
MACHINE LEARNIG CON LENGUAJE DE PROGRAMACIÓN PYTHON
        </h4>

        """
    st.markdown(hide, unsafe_allow_html=True)
```

```
    #st.title('Escuela Superior Politecnica del Chimborazo')
    #st.subheader('           Bryan Luna ')


    #st.write('Objetivos ')

  elif choice == "Clasificación":
    caratula = Image.open('D:/Proyectos/Entornos/App/clas_im.png')
    st.image(caratula, width=700)


    st.header(' Usted puede realizar un diagnóstico previo de manera
computacional gracias a un modelo de Machine Learning ')

    foto_2 = st.file_uploader("Carga una imagen", type=["png", "jpg",
"jpeg"])  #

    if foto_2 is not None:
      image2 = Image.open(foto_2)

      mostrar = np.array(image2)
      image_2 = image2.resize((512,512))
      image_2 = image_2.convert("RGB")

      img_array2 = keras.preprocessing.image.img_to_array(image_2)
      st.image(image2, caption="Imagen a diagnosticar",
use_column_width=False)
    # predicción
    if st.button("Reconocer"):
      impr = model_prediction(image_2, model0)
      if impr == 'Covid 19':
        st.success(f'Después del análisis de la imágen existe la
probabilidad de: {impr}')
      if impr == 'Pnuemonia':

        st.success(f'Después del análisis de la imágen existe la
probabilidad de: {impr} producto de Covid 19' )
      elif impr == 'Normal':
        st.success(f'Después del análisis de la imágen existe la
probabilidad de que el pasiente este en un estado : {impr} ' )



if __name__ == '__main__':
  main()
```

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

DIRECCIÓN DE BIBLIOTECAS Y RECURSOS DEL APRENDIZAJE

UNIDAD DE PROCESOS TÉCNICOS Y ANÁLISIS BIBLIOGRÁFICO Y DOCUMENTAL

REVISIÓN DE NORMAS TÉCNICAS, RESUMEN Y BIBLIOGRAFÍA

Fecha de entrega: *20 / 01 / 2022*

INFORMACIÓN DEL AUTOR/A (S)
Nombres – Apellidos: *Bryan Darwin Luna Bravo*
INFORMACIÓN INSTITUCIONAL
Facultad: *Ciencias*
Carrera: *Física*
Título a optar: *Físico*
f. Analista de Biblioteca responsable: *Ing. Leonardo Medina Ñuste MSc.*

LEONARDO FABIO MEDINA NUSTE

Firmado digitalmente por LEONARDO FABIO MEDINA NUSTE
Nombre de reconocimiento (DN): c=EC, o=BANCO CENTRAL DEL ECUADOR, ou=ENTIDAD DE CERTIFICACION DE INFORMACION-ECIBCE, l=QUITO, serialNumber=0000621485, cn=LEONARDO FABIO MEDINA NUSTE
Fecha: 2022.01.20 11:22:43 -05'00'

2240-DBRA-UTP-2021