UNIVERSIDAD NACIONAL JOSÉ MARÍA ARGUEDAS

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Presentado por

JOSÉ LUIS BARBOZA GONZALES

EFECTIVIDAD DE UN MODELO DE CLASIFICACIÓN BASADO EN DEEP LEARNING EN LA DETECCIÓN DE COVID-19

Asesor

MSc. HERWIN ALAYN HUILLCEN BACA

Co - Asesor

MSc. FLOR DE LUZ PALOMINO VALDIVIA

TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE

INGENIERO DE SISTEMAS

ANDAHUAYLAS – APURÍMAC – PERÚ

2023

APROBACION DEL ASESOR

Quién suscribe:
MSc. Herwin Alayn Huillcen Baca por la presente:

CERTIFICA,
Que, el Bachiller en **Ingeniería de Sistemas, José Luis Barboza Gonzales** ha culminado satisfactoriamente el informe final de tesis intitulado: "**EFECTIVIDAD DE UN MODELO DE CLASIFICACIÓN BASADO EN DEEP LEARNING EN LA DETECCIÓN DE COVID-19"** para optar el Título Profesional de Ingeniero de Sistemas.

Andahuaylas, 06 de noviembre de 2023.

MSc. Herwin Alayn Huillcen Baca
Asesor

Br. José Luis Barboza Gonzales
Tesista

ANEXO 32

DECLARACIÓN JURADA DE AUTENTICIDAD

Yo, Barboza Gonzales José Luis, identificado (a) con DNI N°**70669118** de la Escuela Profesional de **Ingeniería de Sistemas** Declaro bajo juramento que el Proyecto Titulado: (Trabajo de Investigación/ Tesis / Trabajo de Suficiencia Profesional) **EFECTIVIDAD DE UN MODELO DE CLASIFICACIÓN BASADO EN DEEP LEARNING EN LA DETECCIÓN DE COVID-19** Es auténtico y no vulnera los derechos de autor. Además, su contenido es de entera responsabilidad del autor (es) del proyecto, quedando la UNAJMA exenta de toda responsabilidad en caso de atentar contra la Ley de propiedad intelectual y derechos de autor.

Andahuaylas, 06 de noviembre de 2023

.....
Firma
N° DNI: 70669118
E-mail: 100522017Z@unajma.edu.pe
N° Celular: 972742928

.....
Firma del Asesor
N° DNI: 30861096
E-mail: hhuillcen@unajma.edu.pe
N° Celular: 978483096

ACTA DE SUSTENTACIÓN DE TESIS

En la Av. José María Arguedas del Local Académico SL01 (Ccoyahuacho) en el auditorio de la Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional José María Arguedas ubicado en el distrito de San Jerónimo de la Provincia de Andahuaylas, siendo las 11:00 horas del día 29 de setiembre del año 2023, se reunieron los docentes: Dra. Norma Lorena Catacora Flores, M.Sc. Iván Soria Solís, Dr. Humberto Silvera Reynaga, en condición de integrantes del Jurado Evaluador del Informe Final de Tesis intitulado: "EFECTIVIDAD DE UN MODELO DE CLASIFICACIÓN BASADO EN DEEP LEARNING EN LA DETECCIÓN DE COVID 19", cuyo autor es el Bachiller en Ingeniería de Sistemas **JOSÉ LUIS BARBOZA GONZALES**, el asesor M.Sc. Herwin Alayn Huillcen Baca y la coasesora M.Sc. Flor de Luz Palomino Valdivia, con el propósito de proceder a la sustentación y defensa de dicha tesis.

Luego de la sustentación y defensa de la tesis, el Jurado Evaluador **ACORDÓ**: APROBAR por UNANIMIDAD al Bachiller en Ingeniería de Sistemas **JOSÉ LUIS BARBOZA GONZALES,** obteniendo la siguiente calificación y mención:

Nota escala vigesimal		Mención
Números	Letras	
18	DIECIOCHO	EXCELENTE

En señal de conformidad, se procedió a la firma de la presente acta en 03 ejemplares.

Dra. Norma Lorena Catacora Flores
Presidente del Jurado Evaluador

M.Sc. Iván Soria Solís
Primer Miembro del Jurado Evaluador

Dr. Humberto Silvera Reynaga
Segundo Miembro del Jurado Evaluador

APROBACIÓN DEL JURADO DICTAMINADOR

LA TESIS: "Efectividad de un modelo de clasificación basado en Deep Learning en la detección de Covid 19"; para optar el Título Profesional de Ingeniero de Sistemas, ha sido evaluada por el Jurado Dictaminador conformado por:

PRESIDENTE: Dra. Norma Lorena Catacora Flores
PRIMER MIEMBRO: M.Sc. Ivan Soria Solis
SEGUNDO MIEMBRO: Mag. Humberto Silvera Reynaga

Habiendo sido aprobado por **UNANIMIDAD**, en la ciudad de Andahuaylas el día 29 del mes de setiembre de 2023

Andahuaylas, 03 de noviembre de 2023.

Dra. Norma Lorena Catacora Flores
PRESIDENTE DEL JURADO DICTAMINADOR

M.Sc. Ivan Soria Solis
PRIMER MIEMBRO DEL JURADO DICTAMINADOR

Mag. Humberto Silvera Reynaga
SEGUNDO MIEMBRO DEL JURADO DICTAMINADOR

Unidad de Investigación de la Facultad Ingeniería
C022-2023

Andahuaylas, 14 de noviembre de 2023

La Unidad de Investigación de la Facultad de Ingeniería, expide la:

Constancia

De porcentaje de similitud (24%) según el software Turnitin, al informe final de investigación: Efectividad de un modelo de clasificación basado en DEEP LEARNING en la detección de Covid 19. Presentado por el Bach. José Luis Barboza Gonzales cuyo Asesor y Coasesora son: MSc. Herwin Alayn Huillcen Baca y MSc. Flor de Luz Palomino Valdivia

Dra. María del Carmen Delgado Laime
Presidente de la Unidad de Investigación de la
Facultad de Ingeniería

MSc. Fidelia Tapia Tadeo
Miembro de la
Unidad de Investigación de la Facultad de Ingeniería

M.Sc. Richard Carrión Abollaneda
Miembro de la
Unidad de Investigación de la Facultad de Ingeniería

C.c
Archivo.

NOMBRE DEL TRABAJO

INFORME FINAL JOSE LUIS BARBOZA (2) (4).docx

RECUENTO DE PALABRAS

12949 Words

RECUENTO DE CARACTERES

73489 Characters

RECUENTO DE PÁGINAS

104 Pages

TAMAÑO DEL ARCHIVO

8.2MB

FECHA DE ENTREGA

Nov 6, 2023 6:48 AM GMT-5

FECHA DEL INFORME

Nov 6, 2023 6:49 AM GMT-5

24% de similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para cada base c

- 23% Base de datos de Internet
- 12% Base de datos de publicaciones
- Base de datos de Crossref
- Base de datos de contenido publicado de Crossr

UNIVERSIDAD NACIONAL JOSÉ MARÍA ARGUEDAS
Ing. Herwin Alayn Huillcen Baca
DOCENTE

Resumen

DEDICATORIA

A Dios, por guiarme y darme un propósito
A mis padres, Walter y Martha, por brindarme su amor y apoyo incondicional.

AGRADECIMIENTOS

A Dios, por dame vida y salud.
A mi familia, por motivarme y confiar en mí.
A mis estimados profesores, compañeros y personas que me apoyaron directa e indirectamente. Cada uno de ustedes ha contribuido significativamente a mi aprendizaje y crecimiento personal.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Glosario siglas y Abreviaturas

PCR	Reacción en cadena de la polimerasa
CNN	Red Neuronal Convolucional
NAAT	Pruebas de amplificación de ácido nucleico
CXR	Radiografía de pecho
PCAF	Módulo de Fusión de Características de Atención de Canal Paralelo
OMS	Organización mundial de la salud
TP	Verdaderos positivos
FP	Falsos positivos
TN	Verdaderos negativos
FN	Falsos negativos
MSP	Minera Shouxin Perú
M	Millones

RESUMEN

La enfermedad del Coronavirus (COVID-19) es una enfermedad extremadamente contagiosa y de rápida propagación; por lo tanto, su diagnóstico temprano es de suma importancia. Ante la problemática de no contar con una herramienta eficaz, además de las pruebas para ayudar en la detección, se plantea este trabajo.

Este trabajo de investigación plantea evaluar la efectividad de un modelo de clasificación hecho con Deep Learning para la identificación de COVID-19, con el objetivo de determinar qué tan efectivo y eficiente es el modelo propuesto para la detección del virus. Para ello, se hizo uso de tres conjuntos de datos públicos (COVID Chest X-ray, COVID Chest X-ray y COVID-19 Radiography Database) y un conjunto de datos propio utilizado para la validación.

Para llevar a cabo la investigación, se utilizaron diversos modelos pre-entrenados basados en la arquitectura DenseNet y se propone una modificación en la arquitectura DenseNet201, donde se realiza una modificación en la primera fase de la convolución y en las "Transition Layers". Este modelo, a su vez, utiliza transfer learning con el Dataset de ImageNet. Los resultados indican que el modelo propuesto llega a una efectividad del 98%, y cuenta con una eficiencia de 18.1M parámetros y 3.2GFLOPs, superando otras propuestas del estado del arte. Se concluye que el modelo propuesto es fiable, efectivo y de bajo costo para la detección del COVID-19.

Palabras Clave: Deep Learning, DenseNet, Covid, Rendimiento, Eficiencia, Eficacia.

ABSTRACT

The Coronavirus disease (COVID-19) is an extremely contagious and rapidly spreading disease, therefore early diagnosis is of utmost importance. Recognizing the problem of not having an effective tool, in addition to tests to aid in detection, this was the reason for proposing this work.

This research focuses on evaluating the effectiveness of a Deep Learning-based classification model for the detection of COVID-19, with the aim of determining how effective and efficient the proposed model is for virus detection, using chest X-ray images. For this purpose, three public datasets were used (COVID Chest X-ray Dataset, COVID Chest X-ray Dataset, and COVID-19 Radiography Database).

To carry out the research, various pre-trained models based on the DenseNet architecture are used, and a modification is proposed in the DenseNet201 architecture where a modification is made in the first phase of convolution and in the "Transition Layers". This model, in turn, uses transfer learning using the ImageNet Dataset. The results show that the proposed model has an effectiveness of 98%, with an efficiency of 18.1M params and 3.2GFLOPs, surpassing other state-of-the-art proposals. It is concluded that the proposed model is reliable, effective, and low-cost for the detection of COVID-19.

Keywords: Deep Learning, DenseNet, Covid, Performance, efficiency, Effectiveness.

CHUMASQA

Coronavirus unquyqa (COVID-19) sinchi contagioso, usqhayta mast’arikuq unquy, chayrayku ñawpaqmanta riqsiyqa aswan chaniyuq. Mana allin yanapakuyniyuq kay sasachakuy riqsichispa, pruebakuna tariypi yanapanapaq yapasqa, kaymi karqa kay llamkayta yuyaychakunankupaq.

Kay mask’ayqa, juk “modelo de clasificación de Deep Learning” nisqapi ruwasqa allin kayninta chaninchaypi churakun, kay COVID-19 unquyta riqsinapaq, chaywantaq yachanapaq mayk’a allin chanta allinchus kay “modelo” yuyasqa “virus” tarinapaq, kay radiografía de sinqamanta siq’ikunata apaykachaspa. Chaypaqmi kimsa “Dataset” willakuykunata llamkachirqaku

(COVID Chest X-ray Dataset, COVID Chest X-ray Dataset y COVID-19 Radiography Database).

Kay maskay ruwanapaq, imaymana ñawpaqmanta yachachisqa modelokuna DenseNet arquitectura kaqpi ruwasqa, chaymanta huk tikrayta “DenseNet201 arquitectura” kaqpi yuyaychakun maypichus huk tikray ruwakun ñawpaq fase convolución kaqpi chaymanta "Capas de Transición" kaqpi. Kay “modelo”, kutichiyninpi, ImageNet Dataset kaqwan t'inkisqa yachayta llamk'achin. Chay ruwasqakunam qawarichin chay modelo propuesto nisqa 98% efectividad nisqayuq kasqanmanta, 18.1M parametros nisqawan 3.2GFLOPs nisqawan eficiencia nisqawan, chaymi huk propuestas de última generación nisqakunatapas atipan. Tukuchikunmi chay modelo propuesto nisqa confiable, efectivo, pisi qolqella COVID-19 nisqa tarinapaq.

Rimarinamanta: Deep Learning, DenseNet, Covid, allin llamkay, kusa kaq, Allin ruway.

INTRODUCCIÓN

La aparición de la pandemia del coronavirus SARS-CoV-2 en 2019 desató una crisis sanitaria global sin precedentes, presentando amenazas inmediatas y duraderas a la vida y supervivencia del ser humano. Este estudio se centra en una de las necesidades fundamentales derivadas de esta crisis: la necesidad de pruebas rápidas, eficientes y precisas para la detección del COVID-19. La importancia de un diagnóstico rápido y certero de COVID-19 se hace evidente cuando consideramos la velocidad a la que se propaga el virus y la necesidad crítica de aislar a las personas infectadas para contener su propagación. Sin embargo, las pruebas existentes presentan ciertas limitaciones en términos de velocidad y precisión. Las pruebas de ácido nucleico, aunque precisas, pueden tardar días en dar resultados, lo que puede retrasar las medidas de aislamiento. Las pruebas de antígenos, por otro lado, son más rápidas, pero menos precisas. Frente a este escenario, surge la necesidad de explorar nuevas alternativas para la detección eficiente y precisa del COVID-19. En este contexto, la Inteligencia Artificial (IA) emerge como una herramienta prometedora, ya que ha demostrado mejorar la calidad y la atención médica en diversos campos. En particular, las técnicas de aprendizaje profundo se han mostrado prometedoras para la detección y prevención temprana de enfermedades. Por lo tanto, esta tesis busca evaluar la efectividad de un modelo de clasificación basado en Deep Learning en la detección del COVID-19 a través de la evaluación de eficiencia y la eficacia. El estudio se justifica en la necesidad de explorar nuevas alternativas de detección del virus que sean fiables, eficaces y de fácil acceso.

Dado el impacto devastador de la pandemia en países latinoamericanos como Perú es crucial buscar soluciones innovadoras que puedan contribuir a la detección temprana.

CAPITULO I: PROBLEMA DE INVESTIGACIÓN

1.1. Descripción del problema

A lo largo de la historia ha existido amenazas contra la vida y supervivencia del ser humano y recientemente en el año 2019 empezó la pandemia mundial del coronavirus SARS-CoV-2 esta enfermedad extremadamente contagiosa y que actualmente habita entre las personas.

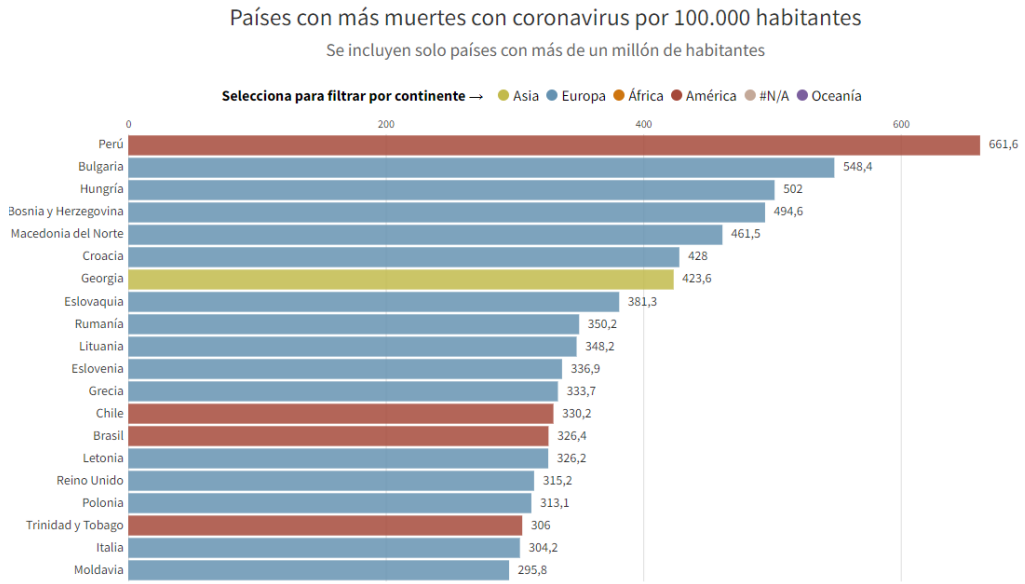
A partir del inicio con el pasar de los días y meses el crecimiento de contagios se tornaba exponencial por tanto las personas con síntomas debían estar en aislamiento hasta que llegue el momento para tomarles una prueba y evitar el contagio masivo, entonces se vio la importancia de las pruebas o test de COVID.

A nivel mundial se hacen pruebas para ello se menciona que "existen dos categorías primordiales de test de virus: los exámenes de amplificación de ácido nucleico (conocidos como NAAT) y los test de antígenos." (Centros para el control y la prevención de enfermedades, 2022) , No obstante, este tipo de pruebas pueden tardar en dar los resultados entre minutos y días. Según la MINSA (2022) menciona que "en el caso de las pruebas moleculares la entrega de resultados tarda entre 3 y 4 días, en el caso de las pruebas de antígenos entre 15 a 30 minutos porque no se necesitan laboratorios para su procesamiento".

Países como Perú han sido de los más afectados por el coronavirus con mayor número de muertes por millón de habitantes como se muestra en la siguiente figura (ver figura 1).

Figura 1

Muertes por COVID-19 por millón de habitantes



Fuente: (RTVE, 2021)

Según las estadísticas que muestra Edouard Mathieu (2020)

> Perú presenta una tasa de mortalidad de 5,996.87 defunciones por cada millón de residentes, lo que la coloca por encima de naciones como Bulgaria, Brasil, Estados Unidos, Reino Unido, Francia, Alemania, Canadá e India. A pesar de que estos países han enfrentado una elevada incidencia de casos de COVID-19, el índice de mortalidad ha sido inferior al que se ha observado en el territorio peruano.

Afectando no solo vidas, sino que también en lo económico “Desde una perspectiva económica, Perú ha experimentado uno de los impactos más significativos a nivel global, evidenciado por una disminución del 11.1% en su Producto Bruto Interno (PBI) durante el año 2020, según los datos proporcionados por el Instituto Nacional de Estadística e Informática.” (UNICEF Perú, 2021).

Por otro lado, la Inteligencia artificial en la actualidad tiene un gran impacto en el campo de la medicina, según Medinaceli Díaz et al. (2021) “La Inteligencia Artificial ha contribuido a elevar la calidad y el cuidado en el ámbito de la salud, ya que, al agilizar el análisis de datos clínicos mediante algoritmos computacionales como el aprendizaje automático y el aprendizaje profundo, se ha posibilitado la detección temprana y la prevención de enfermedades.” (p.93), por lo tanto se puede considerar la inteligencia artificial como una alternativa para la detección de enfermedades de forma eficiente mediante el uso de recursos como imágenes radiológicas como lo menciona Sanofi campos (2020) “actualmente la inteligencia artificial Se encuentra involucrada en el proceso de identificación de medicamentos novedosos, así como en la interpretación de imágenes de diagnóstico por radiología y en el análisis del genoma de un individuo, lo que contribuye a la comprensión de la evolución de una enfermedad”.

Si bien existen pruebas de detección como PCR (molecular) que son bastante eficaces; pero tienen un tiempo considerable en obtener resultados en días; por otro lado, también existen pruebas rápidas (antígenos), pero no son lo bastante fiables. Por lo tanto, el problema radica en que no existen alternativas que sean fiables y a la vez eficaces, tampoco son de fácil acceso, especialmente en la población de bajos recursos.

1.2. Formulación del problema

1.2.1. Problema general

¿Cuál es la efectividad de un modelo de clasificación basado en Deep Learning en la detección de COVID 19?

1.2.2. Problemas específicos

- ¿Cuál es la eficiencia de un modelo de clasificación basado en Deep Learning en la detección de COVID 19?
- ¿Cuál es la eficacia de un modelo de clasificación basado en Deep Learning en la detección de COVID 19?

1.3. Objetivos

1.3.1. General

Evaluar la efectividad de un modelo de clasificación basado en Deep Learning en la detección de COVID 19.

1.3.2. Específicos

- Evaluar la eficiencia de un modelo de clasificación basado en Deep Learning en la detección de COVID 19.
- Evaluar la eficacia de un modelo de clasificación basado en Deep Learning en la detección de COVID 19.

1.4. Justificación

La propuesta que se presenta nace de la necesidad de explorar nuevas alternativas, ampliar las opciones para la detección de COVID 19 y con esto aportar una nueva herramienta complementaria que viene a ser eficiente y eficaz.

Viendo la problemática y en un contexto desfavorable se toma como justificación plantear una nueva alternativa para la detección de este virus de manera eficiente y con un grado de exactitud confiable para evitar la propagación masiva. Además, aportar a la comunidad científica con una nueva alternativa de la arquitectura DenseNet disponible.

Los beneficios de este modelo incluyen una mayor confianza en los resultados de detección, una reducción en los tiempos de diagnóstico y una mayor eficiencia en el uso de recursos. Al mejorar la precisión y eficiencia en la detección, se pueden tomar decisiones más rápidas y efectivas en el tratamiento y control de la enfermedad. Además, este modelo puede tener aplicaciones más amplias en la detección de otras enfermedades respiratorias, lo que ampliaría su utilidad en el campo de la medicina.

1.5. Limitaciones

Las radiografías son una herramienta de diagnóstico comúnmente utilizada en entornos clínicos, pero hay limitaciones en la disponibilidad en imágenes digitales debido a que generalmente el resultado es entregado en una placa radiográfica y no en una imagen como tal, asimismo hay limitaciones con respecto al personal capacitado para interpretar las imágenes.

En esta investigación no se cuenta con un equipo de cómputo de alto rendimiento (GPU de alto desempeño), solamente se hizo uso de entorno de GPU basado en Google Colaboratory en la capa gratuita, el cual ciertamente afecta algunas pruebas y modificaciones mayores en la arquitectura.

CAPITULO II: ANTECEDENTES

2.1. Antecedentes o trabajos previos

2.1.1. Antecedentes internacionales

Narin et al. (2021) en su trabajo titulado "Automatic detection of coronavirus disease (COVID-19) using X-ray images and deep convolutional neural networks" viendo el incremento de la propagación del virus consideraron necesario implementar un sistema automático de detección como una rápida alternativa para prevenir la propagación entre personas con COVID-19. En este estudio proponen cinco modelos pre entrengados (ResNet50, ResNet101, ResNet152, InceptionV3 y Inception-ResNetV2) para la detección de pacientes con coronavirus y neumonía usando radiografías de tórax. Obteniendo como resultado de que el modelo pre entrenado ResNet50 ofrece el más alto rendimiento con una exactitud del 96.1% en comparación con los otros cuatro. La Investigación de Narin et al. (2021) es tomada como referencia para hacer una comparativa con el resultado de se obtiene en este estudio.

Afshar et al. (2020) en su trabajo de investigación titulado "COVID-CAPS: A Capsule Network-based Framework for identification of COVID-19 cases from x-ray Images" vieron la urgencia de desarrollar una red neuronal profunda basada en redes neuronales convolucionales para facilitar la identificación de casos COVID, este trabajo presentan una alternativa modelo de framework basado en redes de capsula llamado COVID-CAPS siendo capaz de manejar pequeños datasets con lo cual alcanzaron una precisión del 95.7% y sensibilidad de 90% con 259 488 número de parámetros entrenables, Esta investigación sirve para hacer la comparativa con el modelo propuesto, así mismo para la construcción de la propuesta.

Rehman et al. (2020) en su trabajo de investigación titulado “Improving coronavirus diagnostic using deep transfer learning” su objetivo de su trabajo fue desarrollar un eficiente diagnosis de COVID-19 diferenciando este de viral neumonía, bacterial neumonía y casos sanos, usando métodos de Deep learning, en esta investigación hicieron uso de modelos pre entrenados (AlexNet, SqueezeNet, GoogLeNet, VGG, MobileNet, ResNet18, etc), para mejorar el rendimiento usando técnicas de transfer learning y comparando el rendimiento de diferentes arquitecturas de redes neuronales convolucionales. El resultado que se obtuvo fue una precisión de. 98.75% con un numero de parámetros 20.2 M.

Moura et al. (2020) En su trabajo titulado “Deep Convolutional Approaches for the Analysis of COVID-19 Using ChestX-Ray Images From Portable Devices” presentan nuevos enfoques completamente automáticos específicamente diseñados para la clasificación de imágenes de rayos X de tórax adquiridas por equipos portátiles en 3 categorías clínicas diferentes: normal, patológica y COVID-19. Para este propósito, se presentan 3 enfoques de aprendizaje profundo complementarios basados en una arquitectura densa en las redes neuronales convolucionales. La respuesta conjunta de todos los abordajes permite potenciar la diferenciación entre pacientes infectados por COVID-19, pacientes con otras enfermedades que manifiestan características similares a la COVID-19 y casos normales. Los enfoques propuestos se validaron con un conjunto de datos recuperado específicamente para esta investigación. A pesar de la mala calidad de las imágenes de rayos X de tórax que es inherente a la naturaleza del equipo portátil, los enfoques propuestos proporcionaron valores de precisión global de 79,62 %, 90,27 % y 79,86 %, respectivamente, lo que permitió un análisis confiable de las radiografías portátiles para facilitar la decisión clínica.

Rahaman et al. (2020) en el trabajo titulado “Identification of COVID-19 samples from chest X-Ray images using deep learning: A comparison of transfer learning approaches”, viendo que uno de los factores críticos detrás de la rápida propagación de la pandemia de COVID-19 es un tiempo prolongado de prueba clínica. La herramienta de imágenes, como la radiografía de tórax (CXR), puede acelerar el proceso de identificación. Por lo tanto, su objetivo fue desarrollar un sistema automático para la detección de COVID-19 usando como dataset imágenes de radiografía del tórax para la realización del trabajo examinaron 15 modelos pre entrenados diferentes de CNN para encontrar el más adecuado para la tarea. La distribución del dataset fue 70% de las imágenes para en entrenamiento, 15 % para la validación y el resto para la prueba.

Llegaron a una precisión más alta con el modelo VGG19 obteniendo un 89.3% con parámetros de 143M. Este trabajo es considerado para hacer la comparativa con el modelo propuesto, aunque en este antecedente solamente se haga uso de la comparativa de modelos pre entrenados.

Xu et al. (2021) en su trabajo de investigación titulado “MANet: A two-stage deep learning method for classification of COVID-19 from Chest X-ray images” distingue casos positivos de COVID-19 de otras cuatro clases incluyendo normal, tuberculosis, neumonía bacterial y neumonía viral usando imágenes CXR, Entre los modelos de clasificación evaluados, ResNet50 con MA alcanza la mayor precisión media en las pruebas, con un 96,32% con parámetros de 23.51M y numero de FLOPs de 1.059GFLOPs. Esta investigación se toma en cuanta debido a que mide la eficiencia y eficacia del modelo.

Sahinbas & Catak (2021) en su trabajo de investigación titulado “Transfer learning-based convolutional neural network for COVID-19 detection with X-rayimages” En este estudio, se utilizaron las cinco arquitecturas diferentes de modelos CNN profundos preentrenados bien conocidos (VGG16, VGG19, ResNet, DenseNet e InceptionV3) para el aprendizaje por transferencia empleando imágenes de rayos X de pacientes con COVID-19 para el método propuesto. Cada uno de los modelos CNN profundos pre entrenados puede analizar una imagen de rayos X para distinguir un caso de COVID-19 y un grupo sano debido a su estructura más profunda. Su modelo alcanza una exactitud de clasificación del 80% para VGG16, que ofrece el mejor rendimiento, donde llegaron con la arquitectura ResNet50 a 3.8GFLOPs y ResNet101 y ResNet152 con 11.3GFLOPs. Esta investigación se toma en cuanta debido a que mide la eficiencia y eficacia del modelo, aunque solamente hace uso de modelos pre entrenados para dicho fin sin modificaciones en las arquitecturas.

Wang et al. (2021) en su trabajo de investigación “Detecting COVID-19 in Chest X-Ray Images via MCFF-Net” diseñaron el Módulo de Fusión de Características de Atención de Canal Paralelo (PCAF) y una nueva estructura de red neuronal convolucional llamada MCFF-Net son empleados para aumentar la eficiencia del proceso de reconocimiento. La red utiliza tres clasificadores: 1-FC, GAP-FC y Conv1-GAP con el objetivo de mejorar la precisión. Los resultados obtenidos en los experimentos demuestran que el modelo MCFF-Net66-Conv1-GAP logra una precisión general del 94.66% en la clasificación. Al mismo tiempo, se logra una precisión del 100% en la clasificación, así como en la precisión, sensibilidad, especificidad y puntuación F1 para la detección de COVID-19 con parámetros de 45.78M y

5.9GFLOPs. Esta investigación se toma en cuenta debido a que mide la eficiencia y eficacia del modelo.

2.1.2. Antecedentes nacionales

Caya Pérez (2020) en su trabajo de investigación titulado “evaluación de modelos de redes neuronales convolucionales aplicado a radiografías de tórax, para apoyar al proceso de diagnóstico de neumonía asociada al covid-19” La propuesta planteó la evaluación de tres modelos aplicados a radiografías de tórax, con el propósito de respaldar el procedimiento de diagnóstico de la neumonía relacionada con el COVID-19 mediante la clasificación de imágenes.

Los modelos utilizados en esta tesis son ResNet50, InceptionV3 y particular. El desarrollo de estos modelos demandó la utilización de transfer learning. Asimismo, se empleó la técnica de aumento de datos (data augmentation) con el propósito de evaluar su efectividad y su impacto en el proceso de entrenamiento de los tres modelos. Se utilizó un conjunto de datos que incluyó imágenes de radiografías de tórax tanto de casos confirmados de COVID-19 como de casos normales para el entrenamiento y la validación de dichos modelos. Por último, se obtuvo la mayor exactitud de 98% con el modelo InceptionV3 y el modelo particular que propone con una exactitud de 97%, sin embargo, no calcula la eficiencia de su modelo y tampoco se encontró la cantidad de parámetros de su propuesta.

CAPITULO III: MARCO TEÓRICO

3.1 Diagnóstico de COVID19

3.1.1 COVID19

La enfermedad COVID-19 es provocada por el coronavirus identificado como SARS-CoV-2. La Organización Mundial de la Salud (OMS) tomó conocimiento de la existencia de este virus por primera vez el 31 de diciembre de 2019, cuando se le informó acerca de un grupo de casos de "neumonía viral" que habían surgido en Wuhan, China (OMS, 2020).

3.1.2 Tipos de pruebas COVID 19

Existen dos categorías principales de test virales: las pruebas de amplificación de ácido nucleico (conocidas como NAAT, según sus siglas en inglés) y las pruebas de antígenos. En determinados casos, podría sugerirse la elección de un tipo de prueba en lugar del otro. Es importante señalar que todas las pruebas deben llevarse a cabo cumpliendo con los estándares establecidos por la Administración de Alimentos y Medicamentos (FDA).

A. Las NAAT, igual a las pruebas de PCR (reacción en cadena de la polimerasa), por lo general se hacen en un entorno de laboratorio y vienen a ser las pruebas más precisas para individuos con o sin síntomas. Estas pruebas tienen la capacidad de identificar el material genético del virus, el cual puede persistir en el cuerpo durante un periodo de hasta 90 días después de un resultado positivo.

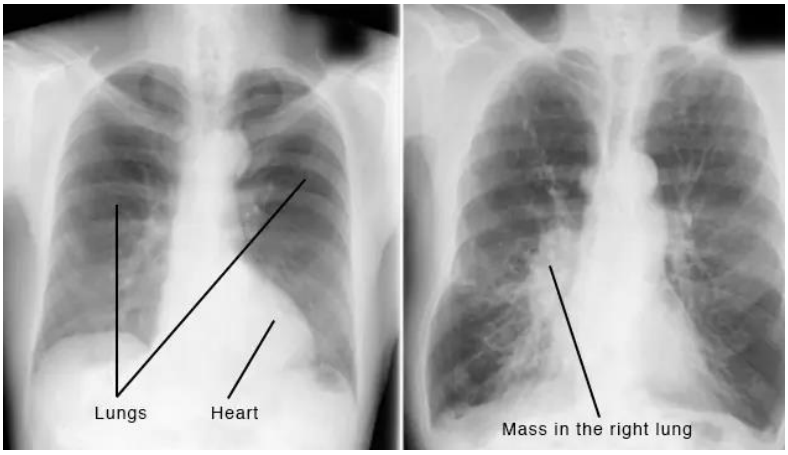
B. Las pruebas de antígenos son exámenes de rápida ejecución que proporcionan resultados en un período de 15 a 30 minutos. Son menos fiables que las pruebas de amplificación de ácido nucleico (NAAT), especialmente cuando se aplican en personas asintomáticas. Un resultado negativo en una única prueba de antígeno no es suficiente para descartar la posibilidad de infección. Para lograr una detección más precisa de la infección, se aconseja repetir la prueba de antígeno con resultado negativo después de al menos 48 horas, lo que se denomina pruebas secuenciales. (NCIRD, 2022).

3.1.3 Radiografia del torax

Las radiografías de tórax, conocidas como rayos X, emplean una mínima cantidad de radiación ionizante para generar imágenes del interior del área torácica. Se utilizan con el propósito de evaluar el estado del corazón, los pulmones y la pared del pecho, y son útiles en la detección y diagnóstico de problemas como la dificultad para respirar, fiebre, una tos persistente, , dolor en el pecho o lesiones. Asimismo, se aplican para contribuir en la identificación y seguimiento de diversas afecciones pulmonares, como la neumonía, el enfisema y el cáncer. Dado que las radiografías de tórax son procedimientos rápidos y sencillos, se convierten en una herramienta particularmente beneficiosa para situaciones de diagnóstico y tratamiento de emergencia.(RSNA, 2022).

Figura 2

Radiografía de tórax



Nota: Adaptado de https://www.mayoclinic.org/es-es/tests-procedures/chest-x-rays/about/pac-20393494

3.2 Tecnologías de clasificación basadas en Deep learning

3.2.1 Deep learning

Según SAS (2023) menciona que:

Deep learning viene a ser un subcampo del machine learning que enseña a una computadora a realizar tareas similares a las humanas, como el reconocimiento de imágenes, del habla y la elaboración de predicciones. Mejora la habilidad para categorizar, identificar, identificar y explicar a través del uso de información y redes neuronales.

3.2.2 Redes neuronales

Según la Documentación de IBM, (2021) menciona que:

Una red neuronal es una representación simplificada que imita la manera en que el cerebro humano lleva a cabo el procesamiento de información. Opera al

mismo tiempo, empleando una gran cantidad de unidades de procesamiento interconectadas que se asemejan a versiones abstractas de las neuronas.

Las unidades de procesamiento se disponen en estratos o capas. Típicamente, una red neuronal consta de tres componentes: una capa de entrada, que contiene unidades que representan las variables de entrada; una o múltiples capas ocultas; y una capa de salida, compuesta por una o varias unidades que representan las variables de salida. Estas unidades están conectadas mediante ponderaciones o fuerzas de conexión variables. Los datos de entrada se introducen en la primera capa, y los valores se transmiten desde cada neurona hacia las neuronas de la capa siguiente. Al final, se obtiene un resultado a partir de la capa de salida.

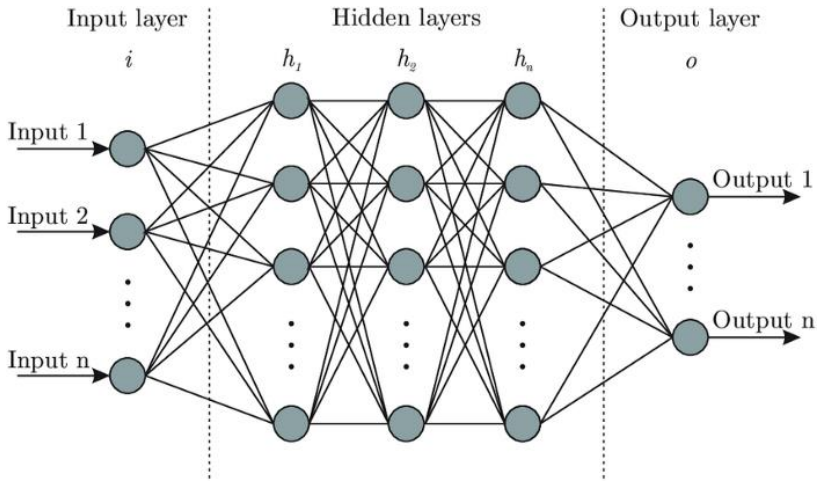
El aprendizaje de la red se produce mediante la revisión de registros individuales, lo que conlleva la generación de una predicción para cada registro y la realización de ajustes en las ponderaciones en caso de que la predicción sea incorrecta. Este ciclo se repite en múltiples ocasiones, permitiendo que la red continúe perfeccionando sus predicciones hasta que cumpla con uno o varios criterios de finalización establecidos.

Inicialmente, todas las ponderaciones se establecen de manera aleatoria, y las respuestas generadas por la red pueden ser poco coherentes. La red adquiere conocimiento a través de un proceso de entrenamiento en el cual se le presentan continuamente ejemplos con resultados conocidos. Las respuestas de la red se comparan con estos resultados conocidos, y la información resultante de esta comparación se retroalimenta a través de la red, ajustando gradualmente las ponderaciones. Conforme avanza el proceso de entrenamiento, la red mejora

progresivamente su capacidad para reproducir resultados previamente conocidos con mayor precisión. Una vez finalizado el entrenamiento, la red puede aplicarse a casos futuros en los que el resultado es desconocido.

Figura 3

Estructura básica de una red neuronal



Nota. Adaptado de estructura básica de redes neuronales, de Lavanya Shukla, 2019, https://towardsdatascience.com/designing-your-neural-networks-a5e4617027ed.

3.2.3 Tipos de redes neuronales

A. Redes neuronales prealimentadas

Las redes neuronales de alimentación directa dirigen el flujo de datos en una única dirección, desde el nodo de entrada hacia el nodo de salida. Cada nodo en una capa se conecta con todos los nodos de la capa subsecuente. Estas redes de alimentación directa recurren a un proceso de retroalimentación para perfeccionar sus predicciones con el transcurso del tiempo. (*¿Qué Es Una Red Neuronal? Guía de IA y ML - AWS*, 2023)

B. Algoritmo de retro programación

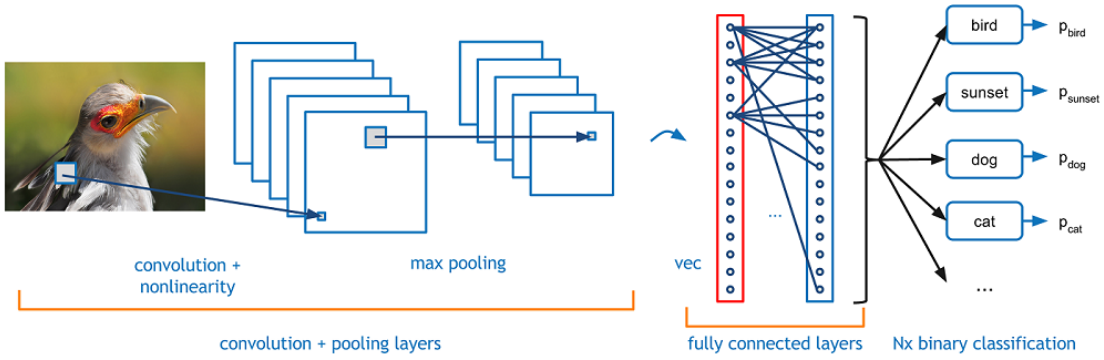
Las redes neuronales artificiales aprenden de manera continua mediante el uso de ciclos de corrección o épocas en los que perfeccionan su capacidad predictiva. En resumen, podemos considerar que los datos se desplazan desde el nodo de entrada hasta el nodo de salida a través de múltiples rutas dentro de la red neuronal. Sin embargo, solo una de estas rutas es la correcta, aquella que vincula el nodo de entrada con el nodo de salida adecuado. Para identificar esta ruta correcta, la red neuronal recurre a un ciclo de corrección. (*¿Qué Es Una Red Neuronal? Guía de IA y ML - AWS*, 2023)

C. Red neuronal convolucional

Las Redes neuronales convolucionales vienen a ser un tipo de redes neuronales donde las neuronas se asemejan a las células en la corteza visual primaria (V1). Este tipo de red es una variación de un perceptrón multicapa, sin embargo, dado a que su aplicación es realizada en matrices bidimensionales, son buenas para tareas de visión por computadora, como en la segmentación y clasificación de imágenes, entre otras aplicaciones.(Juan Barrios, 2023)

Figura 4

Arquitectura de una red neuronal convolucional



Nota. Adaptado de arquitectura típica de una red neuronal convolucional, Redacción KeepCoding, 2023, https://keepcoding.io/blog/arquitectura-tipica-red-neuronal-convolucional/

3.2.4 Arquitectura DenseNet

DenseNet es un tipo de red neuronal convolucional que se caracteriza por su empleo de conexiones densas entre capas, a través de bloques densos, en los que se conectan todas las capas (con mapas de características del mismo tamaño) directamente entre sí. Para preservar la naturaleza de avance, cada capa obtiene entradas que son adicionadas de todas las capas precedentes y estas transmiten sus propios mapas de características a todas las capas posteriores.(*DenseNet Explained | Papers With Code*, n.d.)

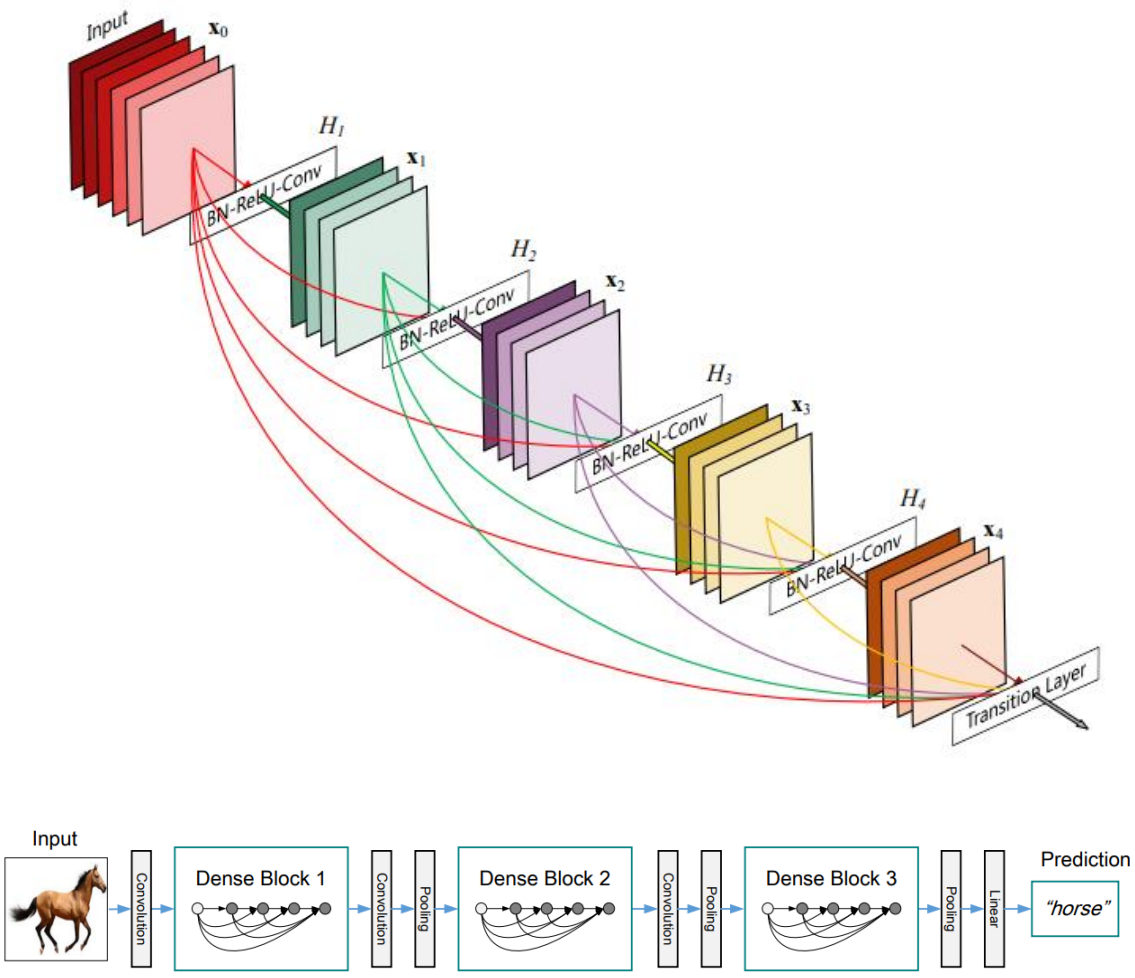
El modelo DenseNet121 ganó el premio CVPR 2017 Best Paper Award y fue desarrollado por investigadores de la Universidad de Cornell, la Universidad de Tsinghua y FaceBook Research. La red neuronal de convolución contiene conexiones más cortas entre las capas de entrada y salida para que la red pueda ser más profunda, más eficiente y más precisa. Con base en estas observaciones,

Gao Huang et al. introdujo DenseNet, que conecta cada capa siguiendo el principio de diseño profundo. Con cada capa, se usa un mapa de características de todas las capas como entrada y luego se usa su propio mapa de características para las siguientes capas. La arquitectura DenseNet tiene muchos puntos fuertes: reduce el descenso del gradiente, mejora la propagación de funciones, promueve la reutilización de funciones y reduce una gran cantidad de parámetros.(Apicella et al., 2022).

DenseNet viene a ser una arquitectura de red en la que cada capa está conectada directamente a todas las demás de forma feed-forward (dentro de cada bloque denso). Para cada capa, los mapas de características de todas las capas anteriores son tratados como entradas que son independientes, mientras que sus propios mapas de características son transmitidas como entradas a todas las capas posteriores. Este modelo de conectividad ofrece las mejores precisiones en datasets como CIFAR10/100 (con o sin aumento de datos) y SVHN (Street View House Numbers). En el conjunto de datos a gran escala ILSVRC (ImageNet Large Scale Visual Recognition Challenge), DenseNet logra una precisión similar a ResNet, pero utilizando menos de la mitad de parámetros y aproximadamente la mitad de FLOPs. (Zhuang, 2022)

Figura 5

Arquitectura DenseNet



Nota. Adaptado de Densely Connected Convolutional Networks, Kilian Q. Weinberger,2018, Donde x representa los DenseBlocks y H la funcion transition layer.

3.2.5 Transfer learning

Es la reutilización de un modelo preentrenado en un nuevo problema. En el aprendizaje por transferencia, una máquina explota los conocimientos adquiridos en una tarea anterior para mejorar la generalización sobre otra. Por ejemplo, al entrenar un clasificador para predecir si una imagen contiene comida, podría utilizar los conocimientos adquiridos durante el entrenamiento para reconocer bebidas.(Donges Niklas, 2022)

3.2.6 ImageNet

ImageNet es un conjunto de datos de imágenes que se estructura conforme a la jerarquía de WordNet, concentrándose en términos sustantivos. En esta base de datos, cada nodo de la jerarquía se encuentra asociado con una gran cantidad de imágenes, en números que alcanzan cientos o miles. El proyecto ha sido decisivo para el avance de la investigación en visión por ordenador y aprendizaje profundo. Los datos están a disposición gratuita de los investigadores para uso no comercial.(ImageNet, 2021).

Figura 6

Imagenes de ejemplo del dataset ImageNet

Nota: Tomado de (https://cs.stanford.edu/people/karpathy/cnnembed/)

ImageNet es una gran base de datos con más de 14 millones de imágenes. Fue diseñada por académicos para la investigación de la visión por ordenador. Fue la primera de su clase en términos de escala. Las imágenes se organizan y etiquetan en una jerarquía.

En el aprendizaje automático y las redes neuronales profundas, las máquinas se entrenan en un amplio conjunto de datos de varias imágenes. Las máquinas deben aprender características útiles de estas imágenes de entrenamiento. Una vez aprendidas, pueden utilizar estas características para clasificar imágenes y realizar muchas otras tareas asociadas a la visión por ordenador. ImageNet ofrece a los investigadores un conjunto común de imágenes para evaluar sus modelos y algoritmos.(Devopedia, 2021)

Tabla 1

Opciones para Definir Modelos en Pytorch

	PyTorch nativo	**PyTorch Lightning**	**TorchVision**
Beneficios	Flexibilidad total para definir modelos personalizados y ciclo de entrenamiento.	Estructura modular de código para una mayor comprensión y mantenibilidad de modelos. Funcionalidades adicionales para el entrenamiento más escalable.	Ofrece modelos pre-entrenados para tareas de visión por computadora, Esto abarca tareas como clasificación de imágenes, identificación de objetos y la segmentación semántica.
Desventajas	La estructura de código puede ser más compleja y difícil de mantener para proyectos grandes y complejos.	No es tan flexible para la definición de modelos personalizados como PyTorch nativo.	La biblioteca se enfoca principalmente en la visión por computadora y puede no ser adecuada para otros tipos de modelos.

Figura 7

Arquitectura DenseNet para ImageNet

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Nota. El grafico representa las distintas arquitecturas con un grow rate de toda la red de 32. Tomado de Densely Connected Convolutional Networks (p.4), por Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger. https://arxiv.org/abs/1608.06993

Figura 8

Definición de un modelo pre-entrenado en Pytorch

```
from torchvision.models import DenseNet121_Weights, DenseNet161_Weights, DenseNet169_Weights, DenseNet201_Weights
def CNN_Models(pretrained=True):
  model = models.densenet121(weights=DenseNet121_Weights.IMAGENET1K_V1)
  num_ftrs = model.classifier.in_features
  model.classifier = nn.Linear(num_ftrs, len(class_names))
  model = model.to(device)
  return model
```

Nota. Se muestra el código de cómo se define un modelo pre-entrenado con ImageNet usando el paquete de torchvision.

3.3 Métricas para medir la eficiencia de un modelo de clasificación

3.3.1 Eficiencia

Según la RAE (2022) “Se refiere a la habilidad de alcanzar los resultados esperados utilizando la menor cantidad de recursos necesaria”.

Se puede definir como la proporción entre los recursos empleados en un proyecto y los resultados alcanzados a partir de este. Esto se manifiesta cuando se logra alcanzar un objetivo utilizando una cantidad reducida de recursos o cuando se logran varios objetivos empleando la misma cantidad de recursos o incluso menos. (GESTIÓN, 2022).

3.3.2 Número de parámetros de una red neuronal

Los parámetros en general son pesos que se aprenden durante el entrenamiento. Son matrices de pesos que contribuyen al poder predictivo del modelo y que se modifican durante el proceso de retropropagación. La cantidad es de acuerdo al algoritmo de entrenamiento que se elija, en particular la estrategia de optimización hace que cambien sus valores. (Vasudev Rakshith, 2019).

3.3.3 FLOPs

Representa una métrica de rendimiento utilizada en operaciones de punto flotante por segundo que un procesador puede ejecutar. Esta métrica es fundamental para el funcionamiento de los procesadores y abarca la manipulación de números de diversos tamaños y fracciones. Su importancia radica en que los procesadores disponen de una capacidad limitada para almacenar los números con los que operan. (Santos Manuel, 2018).

La notación científica en coma flotante es una forma de representar números reales, ya sean muy grandes o muy pequeños, de manera eficaz y concisa en computadoras, permitiendo realizar operaciones aritméticas. El estándar vigente para esta representación es el IEEE 754.(Wikepedia, 2022).

Según Sooryakiran Pallikulathil (2021) menciona que en el contexto de Deep Learning "FLOPs, significa simplemente el número total de operaciones en coma flotante necesarias en una sola pasada de avance o en una época. Cuanto mayor sea el número de FLOPs, más lento será el modelo y, por tanto, menor será el rendimiento".

3.4 Métricas para medir la eficacia de un modelo de clasificación

3.4.1 Eficacia

Según RAE (2022) es la "Capacidad de lograr el resultado previsto o el efecto que se desea o se espera".

Según GESTIÓN (2022) "La eficacia se refiere al grado de cumplimiento de metas y objetivos. Hace alusión a nuestra habilidad para alcanzar lo que nos proponemos".

3.4.2 Efectividad

Si consideramos que la eficacia se relaciona con el logro de metas específicas y la eficiencia se relaciona con lograr esas metas con el uso más eficaz de recursos limitados, la efectividad abarca ambos conceptos. En otras palabras, ser efectivo implica alcanzar las metas de manera eficaz y eficiente al mismo tiempo, y buscar la optimización de los recursos disponibles.(Galia Puerto, 2019).

3.4.3 Exactitud o Accuracy

La exactitud se relaciona con la proximidad del resultado de una medición al valor real. Desde una perspectiva estadística, la exactitud guarda relación con el sesgo en una estimación. Se expresa como la proporción de resultados precisos, incluyendo verdaderos positivos (TP) y verdaderos negativos (TN), dividida por el total de casos examinados, que comprende verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos.(Barrios, 2019).

Figura 9

Fórmula para el cálculo del Accuracy

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

Nota. Donde verdaderos positivos (TP), falsos positivos (FP), verdaderos negativos (TN), falsos negativos (FN) de la matriz con fusión, adaptado de Medium por Harikishan NB, 2019

3.4.4 Precisión

Según Barrios (2019) La precisión se relaciona con la variabilidad en los valores obtenidos a través de múltiples mediciones de una magnitud. Cuando esta variabilidad es reducida, la precisión es mayor. Se representa como la proporción de verdaderos positivos en relación con el total de resultados positivos, lo que incluye tanto verdaderos positivos como falsos positivos.

Figura 10

Fórmula para cálculo de la precisión

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

Nota. Adaptado de Medium por Harikishan NB, 2019

3.4.5 Sensibilidad o recall

Según Barrios (2019) menciona que "Esta medida también es reconocida como la tasa de verdaderos positivos (True Positive Rate) o TP. Representa la proporción de casos positivos que fueron acertadamente detectados por el algoritmo."

Figura 11

Fórmula para el cálculo de la sensibilidad

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

Nota. Adaptado de Medium por Harikishan NB, 2019

3.4.6 f1-score

Esta métrica es ampliamente utilizada ya que combina tanto la precisión como la sensibilidad en una sola medida. Por esta razón, resulta especialmente útil cuando existe un desequilibrio en la distribución de las clases, como ocurre en situaciones donde, por ejemplo, el 15% de los casos corresponden a una condición y el 85% restante a la otra, lo cual es una situación común en el ámbito de la salud.(Barrios, 2019)

Figura 12

Fórmula para el cálculo del f1-score

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

Nota. Adaptado de Medium por Harikishan NB, 2019

CAPITULO IV: METODOLOGÍA DE LA INVESTIGACIÓN

4.1 Tipo de investigación

Dentro de las rutas de investigación este trabajo se encuentra en el enfoque cuantitativo ya que según Hernández Sampieri & Mendoza Torres (2018) “brinda una gran posibilidad de repetición y un enfoque sobre puntos específicos de los fenómenos, además de que facilita la comparación entre estudios similares”.

4.2 Nivel de investigación

El nivel de investigación que se toma es de tipo descriptiva ya que el objetivo es describir y analizar las características y propiedades de un modelo de deep learning en la detección de COVID-19 para luego determinar la eficiencia y eficacia del modelo.

4.3 Diseño de la investigación

Diseño de la investigación de tipo experimental y según la clasificación de Campbell y Stanley (1996) estaría dentro del tipo preexperimental debido a que se cuenta con un solo grupo único de datos.

4.4 Población

La población para esta investigación fue compuesta por las imágenes de radiografías de COVID 19 y normales que están disponibles en los diferentes datasets.

4.5 Muestra

Para la muestra en este estudio se hace uso de dos datasets para entrenamiento "COVID Chest X-ray Dataset" obtenido de un repositorio público de GitHub y "Chest X-ray Dataset (Pneumonia)" entre ambos llegando a contar con 642 imagenes. Para la evaluación se usa el dataset "COVID-19 Radiography Database", obtenido de la plataforma Kaggle se toma 200 imágenes, asimismo se añade un dataset propio del investigador con 120 imágenes normales obtenido de personas que trabajan en Minera Shouxin Perú S.A (MSP).

El primer dataset cuenta imágenes de distintos tipos de enfermedades.

Figura 13

Resumen del Dataset de Imagenes covid-chestxray-dataset

Type	Genus or Species	Image Count
Viral	COVID-19 (SARSr-CoV-2)	468
	SARS (SARSr-CoV-1)	16
	MERS-CoV	10
	Varicella	5
	Influenza	4
	Herpes	3
Bacterial	*Streptococcus* spp.	13
	Klebsiella spp.	9
	Escherichia coli	4
	Nocardia spp.	4
	Mycoplasma spp.	5
	Legionella spp.	7
	Unknown	2
	Chlamydophila spp.	1
	Staphylococcus spp.	1
Fungal	*Pneumocystis* spp.	24
	Aspergillosis spp.	2
Lipoid	Not applicable	8
Aspiration	Not applicable	1
Unknown	Unknown	59

Nota. Adaptado de (https://github.com/ieee8023/covid-chestxray-dataset)

El segundo dataset cuenta con imágenes de radiografías de pulmones normales y con neumonía, de los cuales se hace uso solo de las imágenes normales.

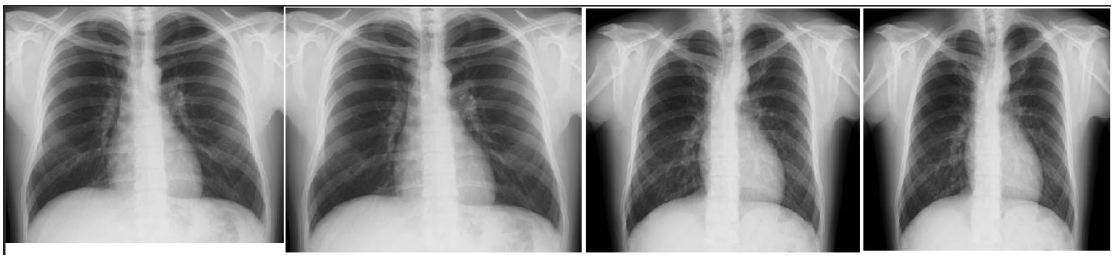
Figura 14

Ejemplos Ilustrativos de Pacientes con Neumonia del Dataset Chest X-Ray Images

Nota. Obtenido de (https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia)

Figura 15

Imágenes del Dataset Propio



Los dataset públicos fueron elegidos por ciertas razones en esta investigación:

- Kaggle y GitHub contienen grandes volúmenes de información que ahorra considerablemente tiempo y esfuerzo en la recopilación de la data.

- Tener datos de este tipo de fuentes proporciona una visión más global de un problema en campos de las ciencias de la computación, Marchine Learning, Inteligencia Artificial.

- Reproducibilidad de la investigación, cabe mencionar que otros investigadores pueden acceder y validar el resultado de la investigación.

4.6 Procedimiento de la investigación

El presente trabajo de investigación sigue las siguientes etapas:

Etapa 1: Recolección de datos

Para la obtención de datos se procedió a poner a disposición los datasets de las fuentes anteriormente descritas, para luego poder ser cargadas en el entorno de desarrollo.

Etapa 2: Preprocesamiento de datos

En esta etapa se hizo el filtro de los datos para poder tener la muestra con el cual se desarrolla el proyecto, en ese sentido se obtienen las imágenes etiquetadas como COVID 19 y Normal. También se hace la normalización de los datos para el proceso de entrenamiento.

Etapa 3: Desarrollo del modelo

Dentro del desarrollo se propone un modelo de Deep Learning usando redes neuronales convolucionales (CNN) basado en la arquitectura DenseNet201 para ello se realizan pruebas de muchas variantes de DenseNet para conseguir un modelo eficiente y eficaz.

Etapa 4: Entrenar el Modelo

En esta etapa se entrena el modelo con los datos configurando la cantidad de épocas (epochs) en lo cual se realiza varias pruebas usando nuevos parámetros y haciendo modificaciones en el código del modelo.

Etapa 5: Evaluación

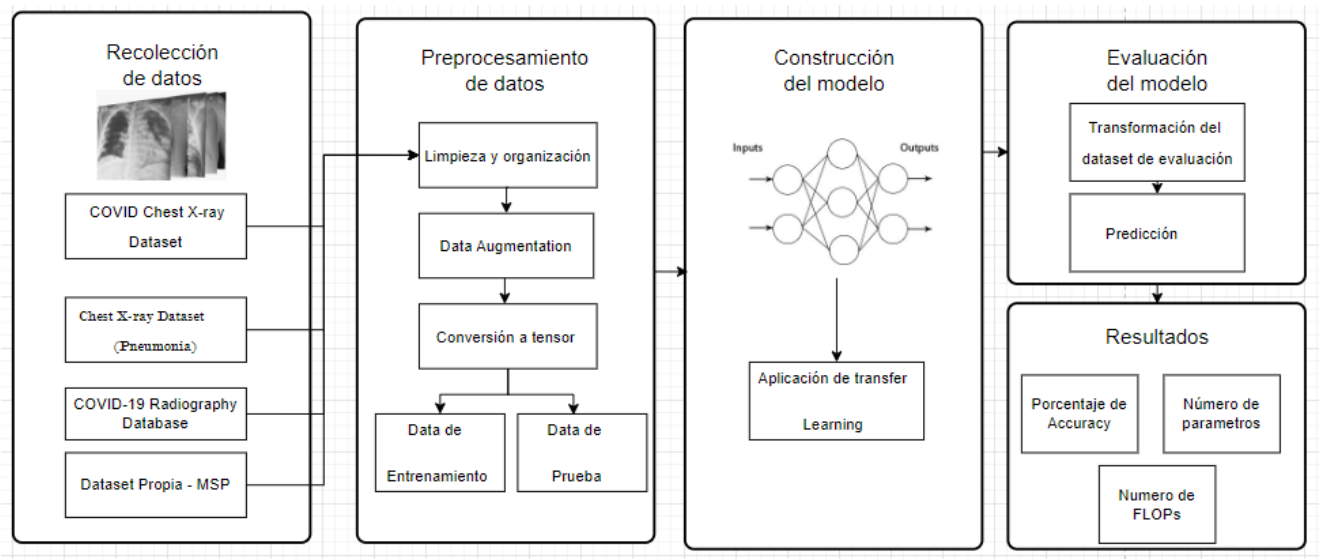
En esta etapa se implementa las métricas de evaluación para evaluar la exactitud, precisión, el rendimiento (número de parámetros y FLOPs) esto para determinar la efectividad mediante la eficiencia y eficacia.

CAPITULO V: RESULTADOS

5.1 Implementación de la propuesta

Figura 16

PipeLine de la propuesta



Nota. Esta figura muestra el proceso de la implementación de la propuesta

5.2 Recolección de datos

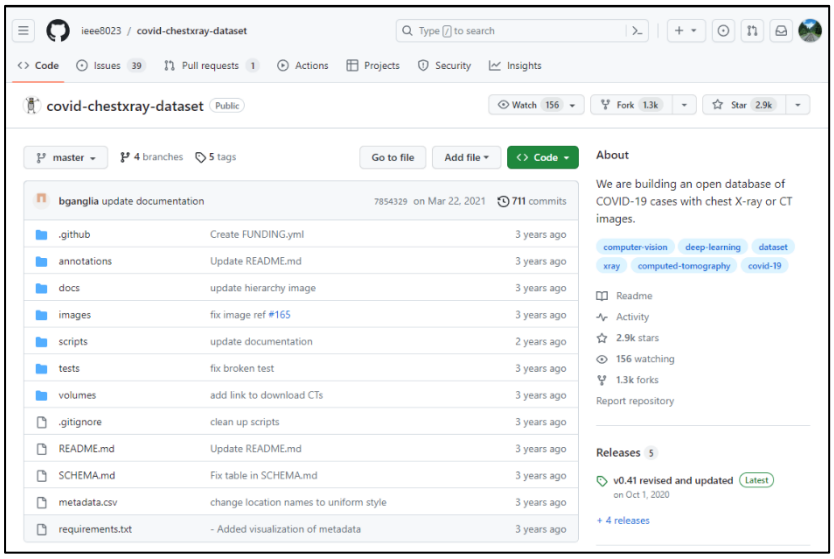
Si bien es cierto la etapa de recopilación de datos es una de las más importantes en la investigación, ya que los resultados dependerán en gran medida de la calidad y cantidad de los datos obtenidos. Para garantizar una recopilación de datos efectiva, es importante definir claramente los objetivos de la investigación y el tipo de datos que se necesitan para responder a las preguntas planteadas en el trabajo de investigación. Además, es fundamental elegir las técnicas y herramientas más adecuadas para recolectar los datos, teniendo en cuenta las características de la población objeto de estudio y los recursos disponibles.

Para esta investigación se hace uso de cuatro datasets que permitan entrenar, validar y evaluar el modelo propuesto:

- COVID Chest X-ray Dataset
- Chest X-ray Dataset (Pneumonia)
- COVID-19 Radiography Database
- Dataset Propio de imágenes normales - MSP

Figura 17

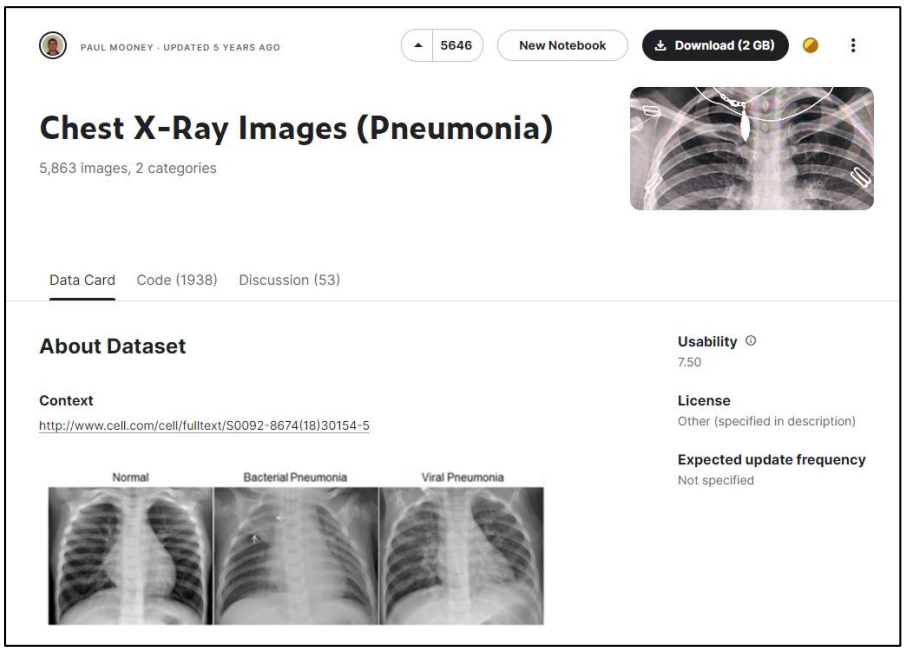
Covid-chestxray-dataset GitHub



Nota. Adaptado de covid-chestxray-dataset en GitHUb, (Cohen et al., 2020)(https://github.com/ieee8023/covid-chestxray-dataset)

Figura 18

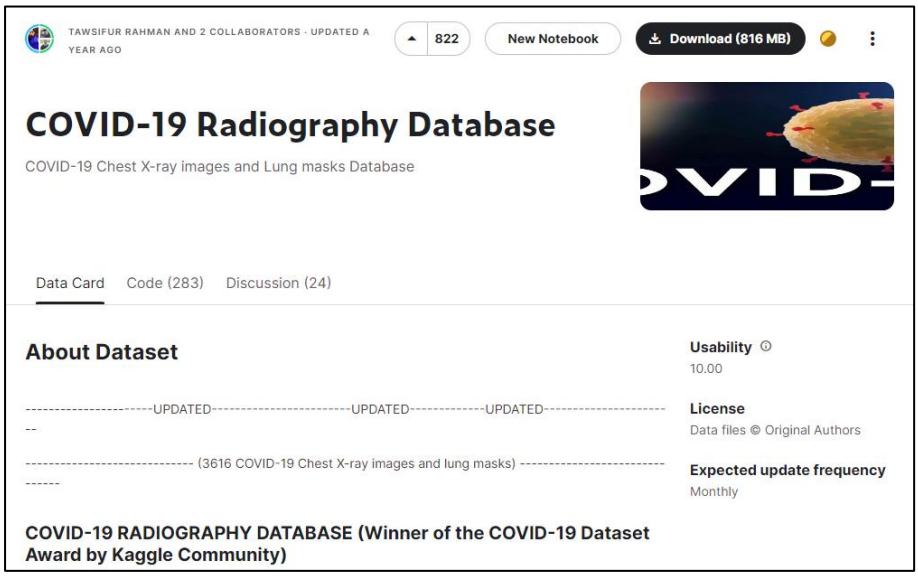
Chestx-ray images (Pneumonia) Dataset



Nota. Adaptado de Chest X-ray images, Mooney, 2018, , (https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia)

Figura 19

Covid-19 Radiography Database



Nota. Adaptado de COVID-19 Radiography Database, Rahman et al., 2020, (https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database)

Para poder hacer uso de los datos de los datasets anteriores estos se descargaron y guardaron en carpetas de Google drive esto para facilitar el acceso a estos mediante la herramienta Google Colaboratory, eso se realizó con los datasets de "COVID-19 Radiography Database" y "Chest X-Ray Images (Pneumonia)" debido a que los datos están organizados correctamente dentro según a las necesidades de la investigación, sin embargo en el caso del dataset "covid-chestxray-dataset" se cuenta con muchos tipos de imágenes de diferentes enfermedades en la misma carpeta por ello se procedió a filtrar solamente los datos etiquetados como covid-19.

Figura 20

Clonando el Repositorio del Dataset Covid Chestxray desde GitHub

```
✓ 39 s  [3]  !git clone https://github.com/ieee8023/covid-chestxray-dataset.git

Cloning into 'covid-chestxray-dataset'...
remote: Enumerating objects: 3641, done.
remote: Total 3641 (delta 0), reused 0 (delta 0), pack-reused 3641
Receiving objects: 100% (3641/3641), 632.96 MiB | 18.21 MiB/s, done.
Resolving deltas: 100% (1450/1450), done.
Updating files: 100% (1174/1174), done.
```

Una vez hecho el proceso de traer los datos del repositorio de GitHub se obtiene una carpeta con el nombre "covid-chestxray-dataset" en el cual se encuentra una carpeta "images" que contiene todas las imágenes de diferentes tipos (ver figura 21).

Figura 21

Carpetas del Dataset covid Chestxray

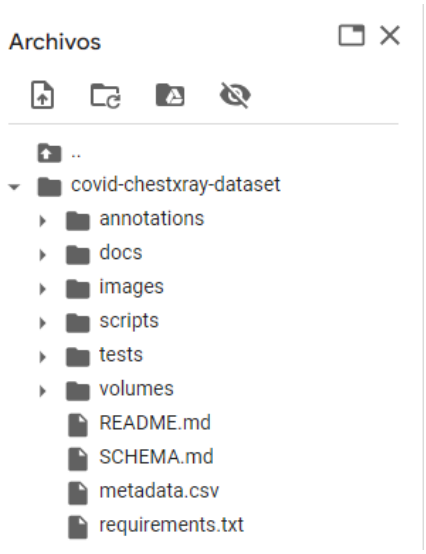


Figura 22

Resumen de Imágenes del Dataset Covid-chestxray-dataset

Type	Genus or Species	Image Count
Viral	COVID-19 (SARSr-CoV-2)	468
	SARS (SARSr-CoV-1)	16
	MERS-CoV	10
	Varicella	5
	Influenza	4
	Herpes	3
Bacterial	*Streptococcus* spp.	13
	Klebsiella spp.	9
	Escherichia coli	4
	Nocardia spp.	4
	Mycoplasma spp.	5
	Legionella spp.	7
	Unknown	2
	Chlamydophila spp.	1
	Staphylococcus spp.	1
Fungal	*Pneumocystis* spp.	24
	Aspergillosis spp.	2
Lipoid	Not applicable	8
Aspiration	Not applicable	1
Unknown	Unknown	59

Nota: Obtenido de (https://github.com/ieee8023/covid-chestxray-dataset)

Para filtrar los datos solamente necesarios de covid-19 se procedió a crear un algoritmo, que haga la búsqueda dentro del archivo “csv” del dataset, para filtrar solamente de la categoría covid-19 además de ello se hizo otro filtro para incluir solo

las filas donde la columna "view" tiene el valor "AP" o "PA", que representan las vistas frontal-anterior y posterior-anterior, respectivamente.

Figura 23

Algoritmo de Filtrado de Imágenes de covid-19 del Dataset chestxray

```
[4] pathfolder = '/content/drive/MyDrive/DeepLearning'
```

```
[5] df = pd.read_csv('./covid-chestxray-dataset/metadata.csv')
    selected_df = df[df.finding=="Pneumonia/Viral/COVID-19"]
    selected_df = selected_df[(selected_df.view == "AP") | (selected_df.view == "PA")]
    print(selected_df.head(2))
    len(selected_df)
```

```
  patientid  offset sex   age                  finding RT_PCR_positive  \
0         2     0.0   M  65.0  Pneumonia/Viral/COVID-19               Y
1         2     3.0   M  65.0  Pneumonia/Viral/COVID-19               Y

  survival intubated intubation_present went_icu  ...              date  \
0        Y         N                  N        N  ...  January 22, 2020
1        Y         N                  N        N  ...  January 25, 2020

                                   location  folder  \
0  Cho Ray Hospital, Ho Chi Minh City, Vietnam  images
1  Cho Ray Hospital, Ho Chi Minh City, Vietnam  images

                                        filename                   doi  \
0  auntminnie-a-2020_01_28_23_51_6665_2020_01_28_...  10.1056/nejmc2001272
1  auntminnie-b-2020_01_28_23_51_6665_2020_01_28_...  10.1056/nejmc2001272

                                         url  license  \
0  https://www.nejm.org/doi/full/10.1056/NEJMc200...      NaN
1  https://www.nejm.org/doi/full/10.1056/NEJMc200...      NaN

                                clinical_notes other_notes Unnamed: 29
0  On January 22, 2020, a 65-year-old man with a ...         NaN         NaN
```

El resultado del filtrado fue de 342 imágenes de covid-19 los cuales fueron almacenados en una carpeta de Google drive para su posterior uso, igualmente para las imágenes normales se hace uso del dataset “chest_xray” una cantidad de 300 imágenes. El procedimiento para ello fue crear una carpeta “COVID19-DATASET” en el cual se almacena los datos de entrenamiento y pruebas.

Figura 24

Creación y Copia de los Dataset para Entrenamiento a la Carpeta COVID19-DATASET

```
[11] os.makedirs(pathfolder + '/COVID19-DATASET/train/covid19')
     os.makedirs(pathfolder + '/COVID19-DATASET/train/normal')

[12] COVID_PATH = pathfolder + '/COVID19-DATASET/train/covid19'
     NORMAL_PATH = pathfolder + '/COVID19-DATASET/train/normal'

[9]  for image in images:
         shutil.copy(os.path.join('./covid-chestxray-dataset/images', image), os.path.join(COVID_PATH, image))#copy c

     for image in os.listdir(pathfolder + '/chest_xray/train/NORMAL')[:300]: # normal from the another dataset that i
         shutil.copy(os.path.join(pathfolder + '/chest_xray/train/NORMAL', image), os.path.join(NORMAL_PATH, image))
```

Figura 25

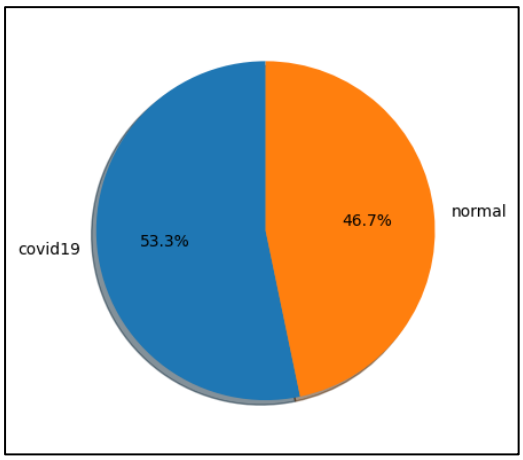
Datasets en Google Drive

5.3 Preprocesamiento de los datos

En esta etapa, se realizó el procesamiento de los datos para que puedan ser utilizados para entrenar y validar el modelo para ello se tiene la siguiente proporción de datos para entrenamiento:

Figura 26

Proporción de la data para entrenamiento

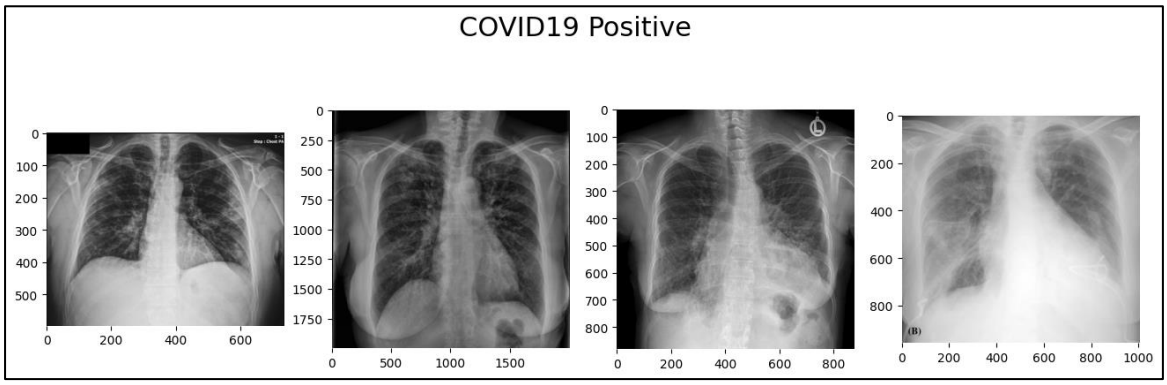


Nota. Para el entrenamiento del modelo se cuenta con 342 y 300 imágenes, este es el resultado en porcentajes.

Se cargó las imágenes aleatoriamente para mostrarlo en un gráfico utilizando la biblioteca matplotlib. También se agrega un título a la figura y configura el tamaño de la figura.

Figura 27

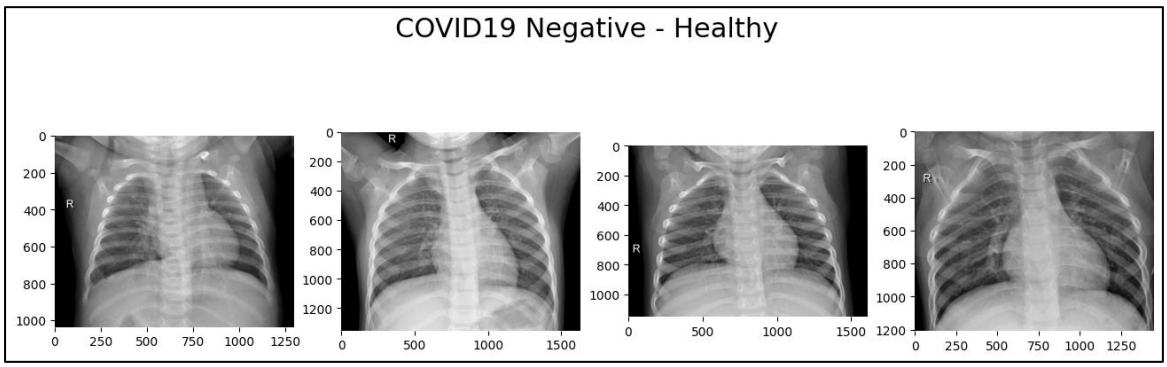
Imágenes de Muestra con covid-19



Nota. Imágenes tomadas aleatoriamente del dataset de entrenamiento

Figura 28

Imágenes Aleatorias Normales



Nota. Imágenes tomadas aleatoriamente del dataset de prueba

Si bien es cierto las computadoras no entienden imágenes, sin embargo, estos están representados mediante pixeles que en ello guardan valores numéricos entre [0-255] que representan la intensidad de luz o color en ese punto de la imagen.

Debido a ello se debe realizar una transformación y normalización de la imagen para esto nos ayuda Pytorch a través de su clase "Transforms.Compose" que nos permite encadenar múltiples transformaciones juntas, así se pueden aplicar secuencialmente a una entrada de imágenes.

Las transformaciones para el conjunto de datos de entrenamiento incluyeron:

- Cambiar el tamaño de la imagen se consideró una imagen de (150x150) pixeles
- Una rotación aleatoria en cual está configurado para rotar hasta un máximo de 10 grados.
- Aplicación de "RandomHorizontalFlip" esta transformación voltea la imagen horizontalmente con una probabilidad de 0.4, este se aplica como una técnica de "Data Augementation".

- Conversión de las imágenes a tensores para ello se usa “transforms.ToTensor” en pytorch que una estructura de datos multidimensional similar a un array o matriz.
- Normalización de la imagen, se realiza restando la media (mean_nums) y dividiendo por la desviación estándar (std_nums). Estos valores de media y desviación estándar se han calculado previamente a partir de los datos de ImageNet que a su vez ayuda a realizar transfer learning.

Las transformaciones para el conjunto de validación incluyen:

- Cambio del tamaño de la imagen a (150,150) pixeles
- Recorte de la imagen en su centro y luego una redimensión a una forma cuadrada de 150x150 pixeles, esto con el motivo de reducir el ruido y enfocar la atención en la información más importante así ayudar a generalizar al modelo mediante esta técnica de Data Augementation.
- Transformación a tensor las imágenes
- Normalización de la imagen

La división de los datos tanto en entrenamiento y validación es de una proporción de 80% y 20% respectivamente, para lo cual de manera aleatoria se dividen los datos para luego para aplicar a los datos de entrenamiento y prueba la clase SubsetRandomSampler que se utiliza para muestrear aleatoriamente los índices de las imágenes para los conjuntos de entrenamiento y validación.

Finalmente, se utilizan los objetos train_sampler y test_sampler que vienen a ser el resultado de aplicar SubsetRandomSampler para crear objetos DataLoader para el

conjunto de entrenamiento y validación, respectivamente. Estos objetos DataLoader proporcionan un iterador que devuelve los datos en lotes (batch_size=8). El resultado de la función es trainloader, valloader y dataset_size, que son los objetos DataLoader para el conjunto de entrenamiento, validación y el tamaño del conjunto de datos.

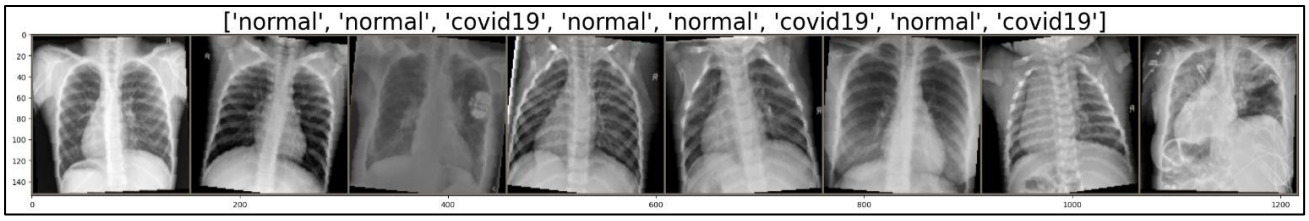
Figura 29

Código de la Aplicación de las Transformaciones y el muestreo de datos

```
def load_split_train_test(datadir, valid_size = .2):
    train_data = datasets.ImageFolder(datadir,
                      transform=data_transforms['train']) #Toma la ruta de las imagenes de su respectivo
nombre y aplica la transformación.
    print(train_data)
    test_data = datasets.ImageFolder(datadir,
                       transform=data_transforms['val'])
    print(test_data)
    num_train = len(train_data)
    num_test = len(test_data)
    indices = list(range(num_train))
    split = int(np.floor(valid_size * num_train))
    np.random.shuffle(indices)
    train_idx, test_idx = indices[split:], indices[:split]
    dataset_size = {"train":len(train_idx), "val":len(test_idx)}
    train_sampler = SubsetRandomSampler(train_idx) # muestra aleatoriamente elementos de un conjunto de
datos
    test_sampler = SubsetRandomSampler(test_idx)
    trainloader = torch.utils.data.DataLoader(train_data,
                      sampler=train_sampler, batch_size=8) # Dataloader proporciona datos de entrenamiento
y validación en batches de 8
    testloader = torch.utils.data.DataLoader(test_data,
                       sampler=test_sampler, batch_size=8)
    return trainloader, testloader, dataset_size
trainloader, valloader, dataset_size = load_split_train_test(DATA_PATH, .2)
dataloaders = {"train":trainloader, "val":valloader}
data_sizes = {x: len(dataloaders[x].sampler) for x in ['train','val']}
class_names = trainloader.dataset.classes
```

Figura 30

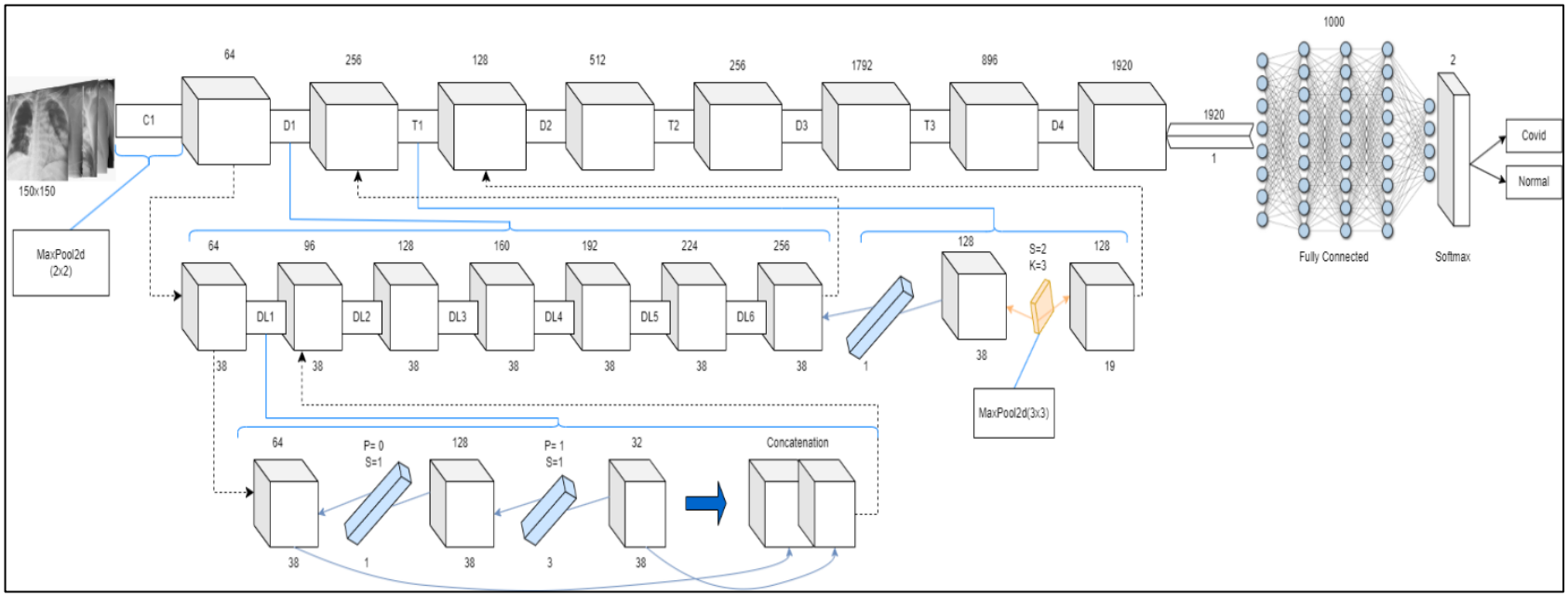
Imágenes de Entrenamiento Etiquetadas



Nota. Obtenido después de separar los datos validación y entrenamiento

Figura 31

Modelo de la Arquitectura DenseNet201 Propuesta



Nota. Modelo propuesto basado en DenseNet201 los cambios hechos en la arquirectura fueron en la primera capa de convolución (C1) cambiado de un max-pool de 3x3 a un max-pool de 2x2, en las capas de transición (T1, T2, T3) se hizo un cambio de Average Pooling 2x2 a Max pooling 3x3 stride 2.

En esta investigación se realizó la construcción del modelo desde una perspectiva donde pueda ser flexible y modificable la arquitectura por completo. La arquitectura usada en esta investigación está basada en DenseNet201 que viene a ser una arquitectura de capas densas donde cada capa recibe una concatenación de las características de las capas anteriores.

Para el desarrollo del modelo se tomó como referencia la forma en como están construidos los modelos pre-entrenados dentro del paquete “torchvision” esto permitió aplicar el concepto de transfer learning en el cual podemos inicializar los pesos de nuestro nuevo modelo con los pesos pre-entrenados del modelo base y ajustarlos en función de nuestro conjunto de datos específico que puede ayudar a mejorar el rendimiento del modelo en conjunto de datos con poca disponibilidad de datos.

La arquitectura DenseNet propone realizar una convolución con un kernel de 7x7, stride de 2 y una capa de max pooling de 3x3 a los inputs como primera parte de la arquitectura, Seguidamente de los bloques densos, donde viene a ser una clase que recibe en su constructor el número de capas, número de características, el tamaño del cuello de botella (Bottleneck), tasa de crecimiento (grow rate), etc.

Dentro del DenseBlock, se aplican varias operaciones, como convoluciones 1x1 para reducir la dimensionalidad de la entrada, con esto reduciendo el número de parámetros requeridos y convoluciones de 3x3 para extraer las características de las imágenes, batch normalization y una función de activación ReLU para introducir no linealidades. La salida de cada capa convolucional se concatena con la entrada del bloque denso para formar la entrada a la siguiente capa convolucional.

Seguidamente se tiene la capa de transición (Transition) que viene hacer una convolución de 1x1 seguido de un average pooling de 2x2 esta capa realiza dos funciones principales:

- **Reducir la dimensionalidad**: la capa de transición se utiliza para reducir la dimensionalidad de los bloques densos (Dense Blocks) que la preceden. Esto se hace para limitar el número de características que se transmiten a través de la red, lo que a su vez reduce el costo computacional y mejora la eficiencia de la red.

- **Controlar la resolución espacial**: la capa de transición también se utiliza para controlar la resolución espacial de la salida de la red. Esto se hace a través del uso de operaciones de agrupamiento (pooling) que reducen la resolución espacial de la salida, lo que ayuda a evitar la sobrerrepresentación y el sobreajuste.

Los datos una vez pasado por los diferentes Denseblock y Transition layers pasan por el proceso “Flatten” donde las entradas de dimensiones son convertidas a una matriz unidimensional lo que permite que las capas fully connected procesen toda la información en una sola matriz unidimensional.

En la fase de clasificación esta la capa “Fully connected” creada mediante la clase de Pytorch nn.Linear donde se le provee la matriz unidimensional para luego pasar a través de una función de activación softmax y clasificar así según las clases que se le provea.

Figura 32

Código de la Arquitectura DenseNet201 Modificada

```
class DenseNet(nn.Module):
    def __init__(self, growth_rate=32, block_config=(6, 12, 48, 32),
                 num_init_features=64, bn_size=4, drop_rate=0, num_classes=2, memory_efficient=False,
):

        super(DenseNet, self).__init__()

        # Convolution and pooling
        self.features = nn.Sequential(OrderedDict([
            ('conv0', nn.Conv2d(3, num_init_features, kernel_size=7, stride=2,
                                padding=3, bias=False)),
            ('norm0', nn.BatchNorm2d(num_init_features)),
            ('relu0', nn.ReLU(inplace=True)),
            ('pool0', nn.MaxPool2d(kernel_size=2, stride=2, padding=1)), # before kernel_size = 3
        ]))

        # Add multiple denseblocks based on config
        # for densenet-201 config: [6,12,48,32]
        num_features = num_init_features
        for i, num_layers in enumerate(block_config):
            block = _DenseBlock(
                num_layers=num_layers,
                num_input_features=num_features,
                bn_size=bn_size,
                growth_rate=growth_rate,
                drop_rate=drop_rate,
                memory_efficient=memory_efficient
            )
            self.features.add_module('denseblock%d' % (i + 1), block)
            num_features = num_features + num_layers * growth_rate
            if i != len(block_config) - 1:
                # add transition layer between denseblocks to
                # downsample
                trans = _Transition(num_input_features=num_features,
                                    num_output_features=num_features // 2)
                self.features.add_module('transition%d' % (i + 1), trans)
                num_features = num_features // 2

        # Final batch norm
        self.features.add_module('norm5', nn.BatchNorm2d(num_features))

        # Linear layer
        self.classifier = nn.Linear(num_features, num_classes)

        # Official init from torch repo.
        for m in self.modules():
            if isinstance(m, nn.Conv2d):
                nn.init.kaiming_normal_(m.weight)
            elif isinstance(m, nn.BatchNorm2d):
                nn.init.constant_(m.weight, 1)
                nn.init.constant_(m.bias, 0)
            elif isinstance(m, nn.Linear):
                nn.init.constant_(m.bias, 0)

    def forward(self, x):
        features = self.features(x)
        out = F.relu(features, inplace=True)
        out = F.adaptive_avg_pool2d(out, (1, 1))

        out = torch.flatten(out, 1)
        out = self.classifier(out)
        return out
```

Nota. Código de la arquitectura DenseNet201 con la modificación en la primera capa convolucional donde se cambia el kernel de Max Pooling de 3x3 a 2x2.

Figura 33

Código del DenseBlock en la arquitectura DenseNet

```
class _DenseBlock(nn.ModuleDict):
    _version = 2

    def __init__(self, num_layers, num_input_features, bn_size, growth_rate, drop_rate,
memory_efficient=False):
        super(_DenseBlock, self).__init__()
        for i in range(num_layers):
            layer = _DenseLayer(
                num_input_features + i * growth_rate,
                growth_rate=growth_rate,
                bn_size=bn_size,
                drop_rate=drop_rate,
                memory_efficient=memory_efficient,
            )
            self.add_module('denselayer%d' % (i + 1), layer)

    def forward(self, init_features):
        features = [init_features]
        for name, layer in self.items():
            new_features = layer(features)
            features.append(new_features)
        return torch.cat(features, 1)
```

Figura 34

Código del DenseLayer de la arquitectura DenseNet

```
class _DenseLayer(nn.Module):
    def __init__(self, num_input_features, growth_rate, bn_size, drop_rate, memory_efficient=False):
        super(_DenseLayer, self).__init__()
        self.add_module('norm1', nn.BatchNorm2d(num_input_features)),
        self.add_module('relu1', nn.ReLU(inplace=True)),
        self.add_module('conv1', nn.Conv2d(num_input_features, bn_size *
                                           growth_rate, kernel_size=1, stride=1,
                                           bias=False)),
        self.add_module('norm2', nn.BatchNorm2d(bn_size * growth_rate)),
        self.add_module('relu2', nn.ReLU(inplace=True)),
        self.add_module('conv2', nn.Conv2d(bn_size * growth_rate, growth_rate,
                                           kernel_size=3, stride=1, padding=1,
                                           bias=False)),
        self.drop_rate = float(drop_rate)
        self.memory_efficient = memory_efficient

    def bn_function(self, inputs):
        "Bottleneck function"
        concated_features = torch.cat(inputs, 1)
        bottleneck_output = self.conv1(self.relu1(self.norm1(concated_features)))
        return bottleneck_output

    def forward(self, input):
        if isinstance(input, Tensor):
            prev_features = [input]
        else:
            prev_features = input

        bottleneck_output = self.bn_function(prev_features)
        new_features = self.conv2(self.relu2(self.norm2(bottleneck_output)))
        if self.drop_rate > 0:
            new_features = F.dropout(new_features, p=self.drop_rate,
                                     training=self.training)
        return new_features
```

Figura 35

Capa de Transición en la Arquitectura DenseNet

```
class _Transition(nn.Sequential):
    def __init__(self, num_input_features, num_output_features):
        super(_Transition, self).__init__()
        self.add_module('norm', nn.BatchNorm2d(num_input_features))
        self.add_module('relu', nn.ReLU(inplace=True))
        self.add_module('conv', nn.Conv2d(num_input_features, num_output_features,
                                          kernel_size=1, stride=1, bias=False))
        self.add_module('pool', nn.MaxPool2d(kernel_size=3, stride=2))# AvgPool2d stride 2
```

Nota. La modificación hecha aquí se encuentra en el Max Pooling aplicado con un kernel size de 3.

5.4.1 Aplicación de Transfer Learning al modelo

Para el proceso de aplicar transfer learning al modelo se descargó los pesos preentrenados para DenseNet201 desde https://download.pytorch.org/models/densenet201-c1103571.pth.

Asimismo, se usó archivos propios de "torchvision" para hacer uso de clases y funciones internas como Weights, WeightsEnum, _ovewrite_named_param, ImageClassification, _IMAGENET_CATEGORIES.

Los archivos fueron almacenados en Drive para luego ser cargados en el entorno de desarrollo y así usar las clases y funciones que fueran necesarias.

Figura 36

Copia de los archivos de torchvision e importe de las clases y funciones

```
[27] !cp /content/drive/MyDrive/DeepLearning/_api.py /content
     !cp /content/drive/MyDrive/DeepLearning/_utils.py /content
     !cp /content/drive/MyDrive/DeepLearning/_presets.py /content
     !cp /content/drive/MyDrive/DeepLearning/_meta.py /content
     !cp /content/drive/MyDrive/DeepLearning/_internally_replaced_utils.py /content
```

```
[28] !ls

     _api.py                  _internally_replaced_utils.py  sample_data
     covid-chestxray-dataset  _meta.py                       _utils.py
     drive                    _presets.py
```

```
[29] from _api import Weights, WeightsEnum
     from _utils import _ovewrite_named_param
     from _presets import ImageClassification
     from _meta import _IMAGENET_CATEGORIES
```

Figura 37

Aplicación de Transfer Learning al Modelo

```
class DenseNet201_Weights(WeightsEnum):
    IMAGENET1K_V1 = Weights(
        url="https://download.pytorch.org/models/densenet201-c1103571.pth",
        transforms=partial(ImageClassification, crop_size=224),
        meta={
            **_COMMON_META,
            "num_params": 20013928,
            "_metrics": {
                "ImageNet-1K": {
                    "acc@1": 76.896,
                    "acc@5": 93.370,
                }
            },
            "_ops": 4.291,
            "_file_size": 77.373,
        },
    )
    DEFAULT = IMAGENET1K_V1

def _densenet(arch, growth_rate, block_config, num_init_features, weights: Optional[WeightsEnum],
pretrained, progress,
              **kwargs):
    if weights is not None:
      _ovewrite_named_param(kwargs, "num_classes", len(weights.meta["categories"]))

    model = DenseNet(growth_rate, block_config, num_init_features, **kwargs)

    if weights is not None:
      _load_state_dict(model=model, weights=weights, progress=progress)
    model.classifier = nn.Linear(model.classifier.in_features, 2)

    return model
```

Nota: Transfer learning con ImageNet retorna una salida de 1000 categorías sin embargo se hace “Finetunning” para obtener solo 2 salidas en nn.Linear.

Seguidamente se inicializó el modelo usando preentrenamiento, asimismo se envió el modelo hacia el dispositivo (CPU o GPU) que se está usando, se define el optimizador de gradiente descendente que se usará en este caso Adam, igualmente el criterio de medir la perdida se configuró CrossEntropyLoss, por último, se definió un programador de tasa de aprendizaje (learning rate scheduler) en PyTorch En este caso particular, se está usando el programador StepLR, que disminuye la tasa de aprendizaje del optimizador en un factor multiplicativo (gamma) cada vez que se alcanza el número de epochs especificado en step_size. la tasa de aprendizaje se reducirá en un factor de 0.1 cada 7 epochs. Por lo tanto, después del epoch 7, la tasa de aprendizaje se reducirá al 10% de su valor original.

Figura 38

Inicialización del Modelo y Definición de Valores para el Entrenamiento

```
model = densenet201(weights='IMAGENET1K_V1')
model = model.to(device)
optimizer = optim.Adam(model.parameters(), lr=1e-3)
criterion = nn.CrossEntropyLoss()
exp_lr_scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=7, gamma=0.1)
model
```

5.5 Entrenamiento del modelo

Para la fase de entrenamiento se creó una función “train_model” el cual recibe como parámetros el modelo, el criterio, optimizador, scheduler que fueron previamente definidos y como ultimo parámetro el número de épocas que representa la cantidad de veces que utiliza el conjunto de datos para entrenar el modelo.

La secuencia de entrenamiento que se siguió fue la siguiente (ver figura 39):

- Envió de datos inputs y labels hacia el dispositivo disponible (CPU o GPU)
- Asignación de los datos al modelo
- Cálculo de la función de perdida entre las salidas del modelo (outputs) y las etiquetas verdaderas (labels)
- Cálculo de los gradientes
- Aplicación del optimizador
- Actualización del Learning rate

La secuencia para visualizar el modelo fue de la siguiente (ver figura 40):

El código primero cambia el modo del modelo a evaluación (model.eval()) y deshabilita el cálculo de gradientes (torch.no_grad()) para reducir el uso de memoria y acelerar la ejecución del código.

Luego, itera a través del conjunto de datos de validación (dataloaders['val']) y realiza las siguientes operaciones para cada imagen de entrada:

- Mueve la imagen y la etiqueta a la GPU si está disponible (inputs.to(device), labels.to(device)).
- Pasa la imagen a través del modelo para obtener las predicciones (outputs = model(inputs)).
- Utiliza la función torch.max para obtener la clase predicha con la mayor probabilidad (_, preds = torch.max(outputs, 1)).
- Visualiza la imagen junto con la etiqueta verdadera y la predicción del modelo utilizando la función imshow y el objeto class_names que contiene los nombres de las clases del conjunto de datos.

- Detiene el bucle y devuelve el control al programa principal si se han procesado todas las imágenes requeridas.

Figura 39

Código de la función para el entrenamiento del modelo

```
def train_model(model, criterion, optimizer, scheduler, num_epochs=10):
    since = time.time()

    best_model_wts = copy.deepcopy(model.state_dict())
    best_loss = np.inf
    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch+1, num_epochs))
        print('-' * 10)

        # Each epoch has a training and validation phase
        for phase in ['train', 'val']:
            if phase == 'train':
                model.train()  # Set model to training mode
            else:
                model.eval()   # Set model to evaluate mode

            current_loss = 0.0
            current_corrects = 0
            current_kappa = 0
            val_kappa = list()

            for inputs, labels in tqdm.tqdm(dataloaders[phase], desc=phase, leave=False):

                inputs = inputs.to(device)
                labels = labels.to(device)

                # We need to zero the gradients in the Cache.
                optimizer.zero_grad()

                with torch.set_grad_enabled(phase == 'train'):
                    outputs = model(inputs)
                    _, preds = torch.max(outputs, 1)# returns array max values ->indices
                    loss = criterion(outputs, labels)

                    # backward + optimize only if in training phase
                    if phase == 'train':
                        loss.backward() #Computes the gradient of current tensor
                        optimizer.step() #A closure that reevaluates the model and returns the loss

                if phase == 'train':
                    scheduler.step()

                # We want variables to hold the loss statistics
                current_loss += loss.item() * inputs.size(0)
                current_corrects += torch.sum(preds == labels.data)
                val_kappa.append(cohen_kappa_score(preds.cpu().numpy(), labels.data.cpu().numpy()))
            epoch_loss = current_loss / data_sizes[phase]
            epoch_acc = current_corrects.double() / data_sizes[phase]
            if phase == 'val':
                epoch_kappa = np.mean(val_kappa)
                print('{} Loss: {:.4f} | {} Accuracy: {:.4f} | Kappa Score: {:.4f}'.format(
                    phase, epoch_loss, phase, epoch_acc, epoch_kappa))
            else:
                print('{} Loss: {:.4f} | {} Accuracy: {:.4f}'.format(
                    phase, epoch_loss, phase, epoch_acc))

            # EARLY STOPPING
            if phase == 'val' and epoch_loss < best_loss:
                print('Val loss Decreased from {:.4f} to {:.4f} \nSaving Weights... '.format(best_loss,
epoch_loss))
                best_loss = epoch_loss
                best_model_wts = copy.deepcopy(model.state_dict())
        if early_stopper.early_stop(epoch_loss):# applying early stopping
            pass#break

    time_since = time.time() - since
    print('Training complete in {:.0f}m {:.0f}s'.format(
        time_since // 60, time_since % 60))
    print('Best val loss: {:.4f}'.format(best_loss))

    model.load_state_dict(best_model_wts)
    return model
```

Figura 40

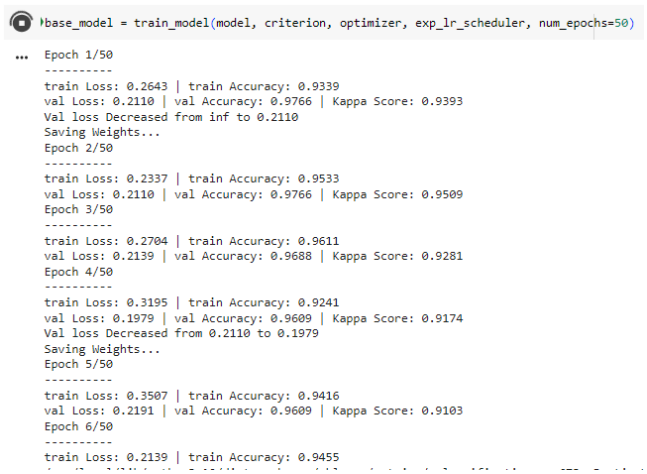
Código de la Visualización del Modelo

```
def visualize_model(model, num_images=6):
    was_training = model.training
    model.eval()
    images_handeled = 0
    ax = plt.figure()

    with torch.no_grad():
        for i, (inputs, labels) in enumerate(dataloaders['val']):
            inputs = inputs.to(device)
            labels = labels.to(device)

            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)

            for j in range(inputs.size()[0]):
                images_handeled += 1
                ax = plt.subplot(num_images//2, 2, images_handeled)
                ax.axis('off')
                print(labels[j].item())
                print(preds[j])
                ax.set_title('Actual: {} predicted:
{}'.format(class_names[labels[j].item()],class_names[preds[j]]))
                imshow(inputs.cpu().data[j], (5,5))

                if images_handeled == num_images:
                    model.train(mode=was_training)
                    return
        model.train(mode=was_training)
```

Una vez definido la función de entrenamiento y visualización del modelo se procedió a ejecutar el proceso de entrenamiento con 50 épocas.

Figura 41

Proceso de Entrenamiento del Modelo

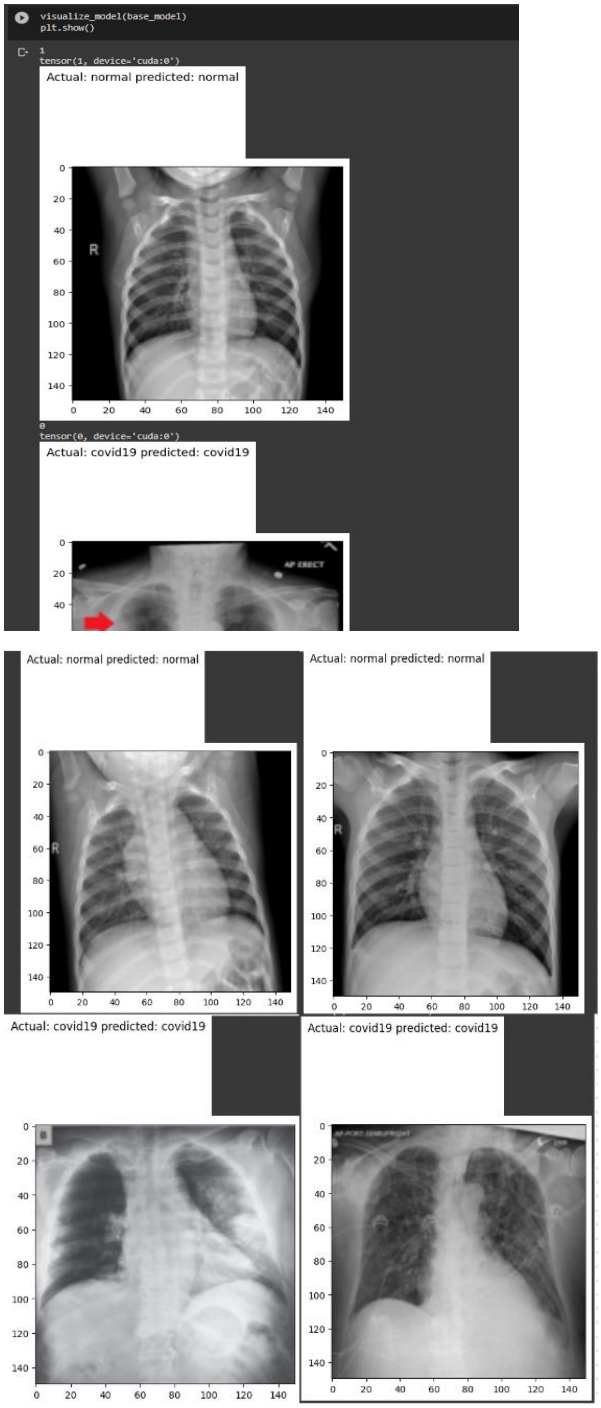


```
base_model = train_model(model, criterion, optimizer, exp_lr_scheduler, num_epochs=50)

Epoch 1/50
----------
train Loss: 0.2643 | train Accuracy: 0.9339
val Loss: 0.2110 | val Accuracy: 0.9766 | Kappa Score: 0.9393
Val loss Decreased from inf to 0.2110
Saving Weights...
Epoch 2/50
----------
train Loss: 0.2337 | train Accuracy: 0.9533
val Loss: 0.2110 | val Accuracy: 0.9766 | Kappa Score: 0.9509
Epoch 3/50
----------
train Loss: 0.2704 | train Accuracy: 0.9611
val Loss: 0.2139 | val Accuracy: 0.9688 | Kappa Score: 0.9281
Epoch 4/50
----------
train Loss: 0.3195 | train Accuracy: 0.9241
val Loss: 0.1979 | val Accuracy: 0.9609 | Kappa Score: 0.9174
Val loss Decreased from 0.2110 to 0.1979
Saving Weights...
Epoch 5/50
----------
train Loss: 0.3507 | train Accuracy: 0.9416
val Loss: 0.2191 | val Accuracy: 0.9609 | Kappa Score: 0.9103
Epoch 6/50
----------
train Loss: 0.2139 | train Accuracy: 0.9455
```

Nota. Muestra las iteraciones que realiza en el momento de entrenamiento

Figura 42

Visualización del Modelo

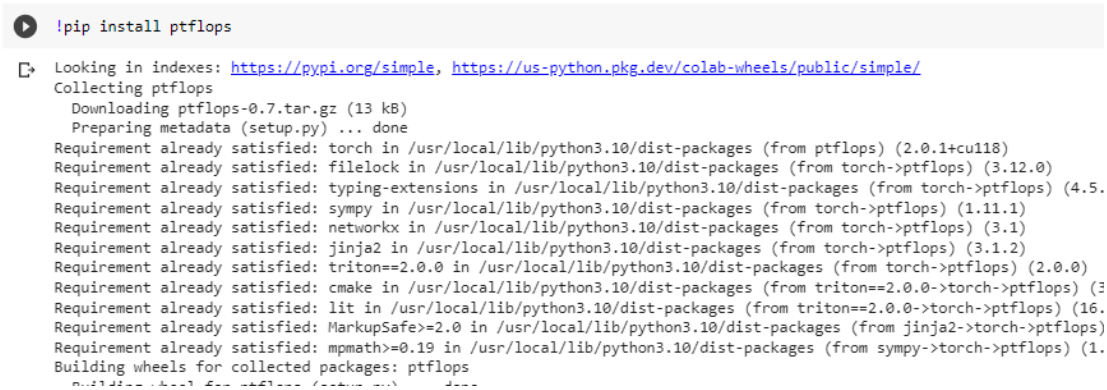


Nota. Resultado de ejecutar la función visualize_model ()

Para el cálculo de la complejidad del modelo se hizo uso de la biblioteca “ptflops”

Figura 43

Instalación de la Biblioteca ptflops



Nota. La biblioteca ptflops calcula la cantidad operaciones que realiza en una red neuronal, asimismo puede calcular el número de parametros, se instala en el entorno de Desarrollo usando el commando pip.

Figura 44

Código para el Cálculo de FLOPs y Número de Parámetros del Modelo

Nota. Se le envía el modelo y las características de la imagen en este caso 3 canales y el ancho y alto de la imagen 150x150 a la función get_model_complexity_info.

Obteniendo un resultado para el modelo propuesto DenseNet201:

- Complejidad computacional de 3.24GFLOPs
- Número de parámetros 18.1M

5.6 Evaluación

Para la etapa de evaluación se hizo uso de un tercer dataset “COVID-19 Radiography Database” se tomaron 100 imágenes de ambas categorías, el cuál de similar manera fue almacenado en Google Drive para luego ser utilizado en el entorno de desarrollo.

Figura 45

Código de Transformación de las Imágenes de Evaluación

```
TEST_DATA_PATH = pathfolder + '/COVID19-DATASET/test/'

test_transforms = transforms.Compose([
                                      transforms.Resize((150,150)),
                                      transforms.ToTensor(),
                                      transforms.Normalize(mean=mean_nums, std=std_nums)
])


test_image = datasets.ImageFolder(TEST_DATA_PATH, transform=test_transforms)

testloader = torch.utils.data.DataLoader(test_image, batch_size=1)
```

Nota: Se realiza la transformación de la imagen cambiando el tamaño de las imágenes a 150x150px luego se convierte a tensor y se normaliza usando la media y la desviación estándar.

Figura 46

Uso del Modelo para la Prueba con el Dataset

```
y_pred_list = []
y_true_list = []
with torch.no_grad():
    for x_batch, y_batch in tqdm.tqdm(testloader, leave=False):
        x_batch, y_batch = x_batch.to(device), y_batch.to(device)
        y_test_pred = base_model(x_batch)
        y_test_pred = torch.log_softmax(y_test_pred, dim=1)
        _, y_pred_tag = torch.max(y_test_pred, dim = 1)
        y_pred_list.append(y_pred_tag.cpu().numpy())
        y_true_list.append(y_batch.cpu().numpy())
```

Nota: En las variables y_pred_list y y_true_list se guardan las etiquetas predichas y las verdaderas respectivamente, se deshabilita el cálculo del gradiente y se usa un ciclo "for" para iterar y hacer la predicción.

5.7 Resultados

Para obtener los resultados se hicieron varias pruebas usando la arquitectura DenseNet con sus variaciones tanto a nivel de profundidad, número de parámetros y complejidad computacional.

Usando Modelo pre-entrenado DenseNet121 durante 50 épocas:

1. El bucle de entrenamiento se ejecutó durante 50 épocas, como se especifica en **num_epochs=50** y esto lo realiza al llamar a la función **train_model()** con los argumentos **model**, **criterion**, **optimizer** y **exp_lr_scheduler**.
2. El bucle de entrenamiento itera a través de cada época y, para cada época, imprime el número de época actual, junto con la pérdida de entrenamiento,

precisión de entrenamiento, pérdida de validación, precisión de validación y Puntuación Kappa.

Figura 47

Entrenamiento con 50 épocas

```
Epoch 47/50
----------
train Loss: 0.1266 | train Accuracy: 0.9591
val Loss: 0.0269 | val Accuracy: 0.9922 | Kappa Score: 0.9844
hello earlystopper class
the counter value is 9
Epoch 48/50
----------
train Loss: 0.1055 | train Accuracy: 0.9630
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:673:
  k = np.sum(w_mat * confusion) / np.sum(w_mat * expected)
val Loss: 0.0263 | val Accuracy: 0.9922 | Kappa Score: nan
hello earlystopper class
the counter value is 10
Epoch 49/50
----------
train Loss: 0.0960 | train Accuracy: 0.9630
val Loss: 0.0230 | val Accuracy: 0.9922 | Kappa Score: 0.9844
hello earlystopper class
the counter value is 11
Epoch 50/50
----------
train Loss: 0.0399 | train Accuracy: 0.9942
val Loss: 0.0251 | val Accuracy: 0.9922 | Kappa Score: 0.9821
hello earlystopper class
the counter value is 12
Training complete in 29m 2s
Best val loss: 0.0203
```

Figura 48

Resultado de DenseNet121 después de 50 épocas

```
[106] print(classification_report(y_true_list, y_pred_list))
0s
                   precision    recall  f1-score   support

                0       0.92      0.99      0.95       100
                1       0.99      0.91      0.95       100

         accuracy                           0.95       200
        macro avg       0.95      0.95      0.95       200
     weighted avg       0.95      0.95      0.95       200
```

Figura 49

Matriz de Confusión para DenseNet121 con 50 épocas

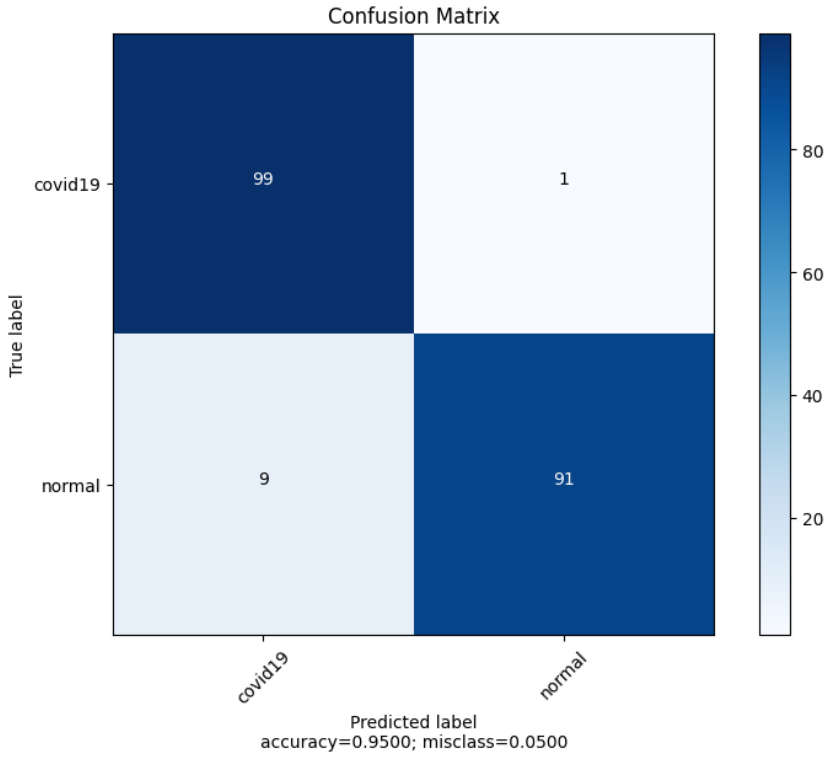


Tabla 2

Complejidad del Modelo DenseNet121

Medida	Cantidad
Complejidad Computacional	0.27 GFLOPs
Número de parámetros	6.96 M

Nota. En número de FLOPs equivale a 270 Millones operaciones de punto flotante en una época de entrenamiento y 6.96 Millones de la suma de los pesos y total de numero de sesgos.

Usando Modelo pre-entrenado DenseNet121 con 100 épocas

Figura 50

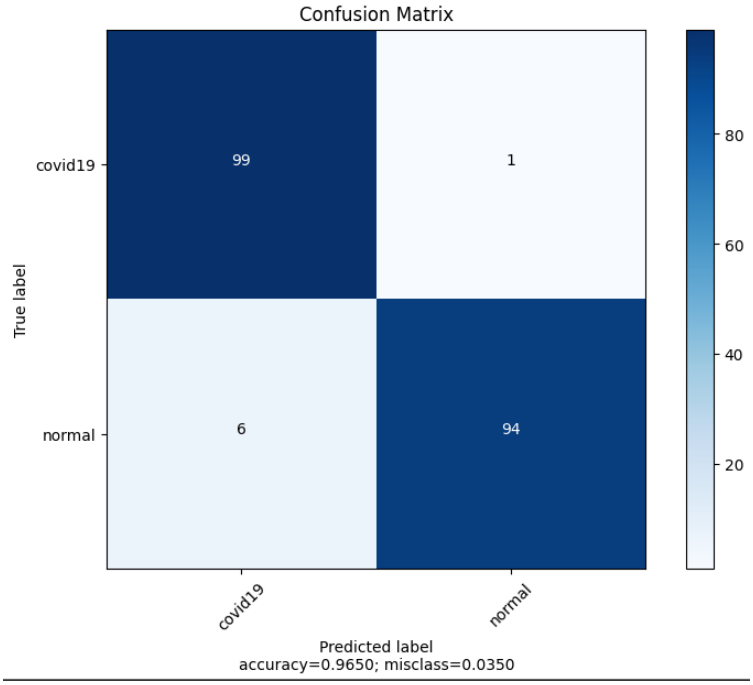
Resultado de DenseNet121 con 100 épocas

```
[45] print(classification_report(y_true_list, y_pred_list))
0s
                   precision    recall  f1-score   support

                0       0.94      0.99      0.97       100
                1       0.99      0.94      0.96       100

         accuracy                           0.96       200
        macro avg       0.97      0.96      0.96       200
     weighted avg       0.97      0.96      0.96       200
```

Nota. Usando la arquitectura DenseNet121 con 100 épocas se llega a una exactitud del 96%.

Figura 51

Matriz de Confusión para DenseNet121



Resultado del entrenamiento con DenseNet169 con 80 épocas aplicando max-pooling a los transition layer.

Figura 52

Reporte de Clasificación de DenseNet169

```
print(classification_report(y_true_list, y_pred_list))
```

	precision	recall	f1-score	support
0	0.90	0.94	0.92	100
1	0.94	0.90	0.92	100
accuracy			0.92	200
macro avg	0.92	0.92	0.92	200
weighted avg	0.92	0.92	0.92	200

Figura 53

Matriz de Confusión para DenseNet169

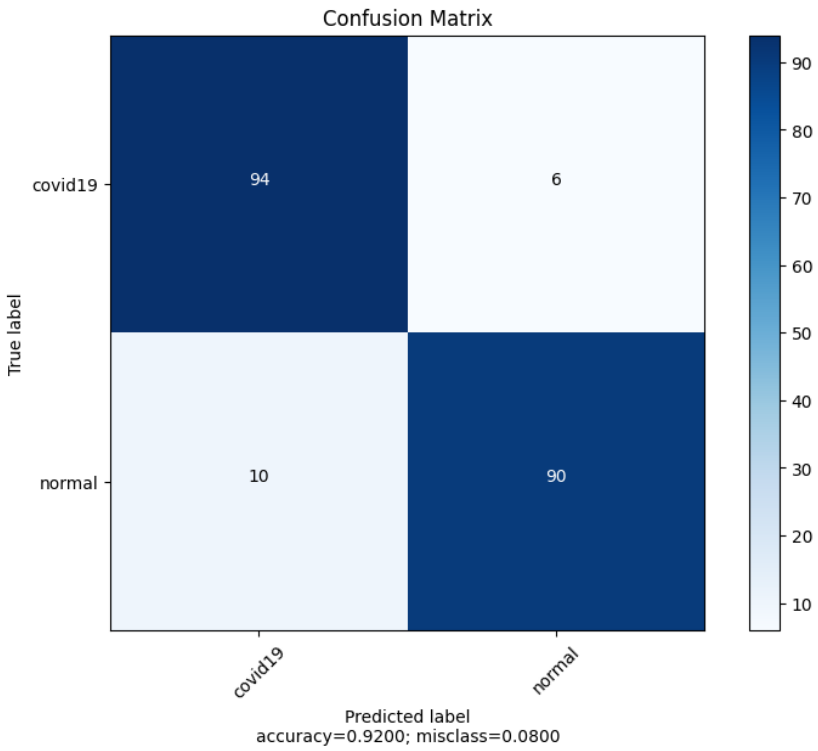


Tabla 3

Complejidad del Modelo DenseNet169

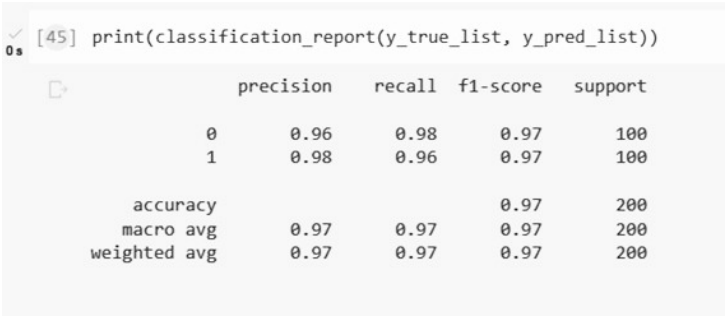
Medida	Cantidad
Complejidad Computacional	2.96 GFLOPs
Número de parámetros	12.49M

Nota. Se observa que la complejidad computacional aumenta a medida que el modelo es mucho más profundo en este caso de 169 capas, tiene 12.49 Millones de parámetros.

Resultado del entrenamiento usando la arquitectura DenseNet201 con 50 épocas:

Figura 54

Reporte de Clasificación para DenseNet201



Nota. El entrenamiento usando la arquitectura DenseNet201 con 50 épocas logra un resultado de 97% de exactitud.

Figura 55

Matriz de Confusión para DenseNet201

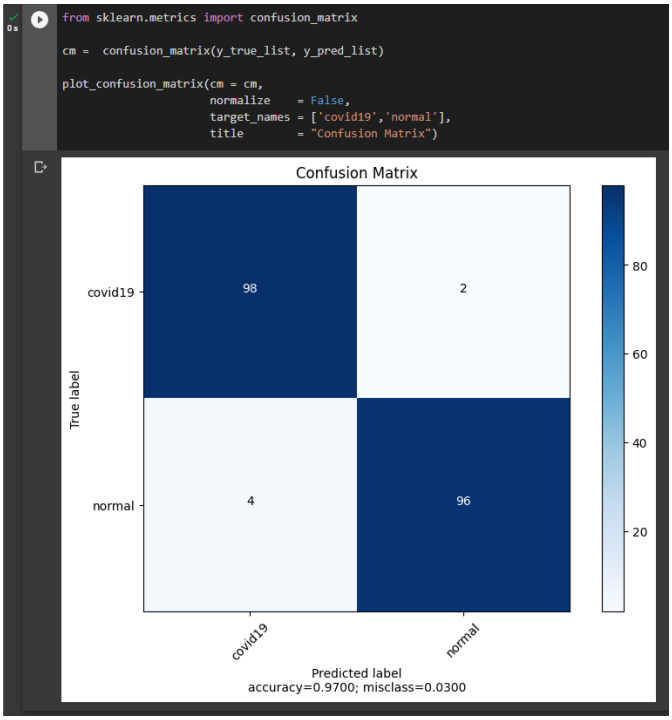


Tabla 4

Complejidad del Modelo DenseNet201

Medida	Cantidad
Complejidad Computacional	3.74 GFLOPs
Número de parámetros	18.1M

Nota. La complejidad computacional y el número de parámetros es mayor a medida que el modelo es mucho más profundo.

Entrenamiento DenseNet201 sin pre-entrenamiento usando grow rate de 48, y cambiando el max pooling con kernel size de 2 en la primera convolución.

Figura 56

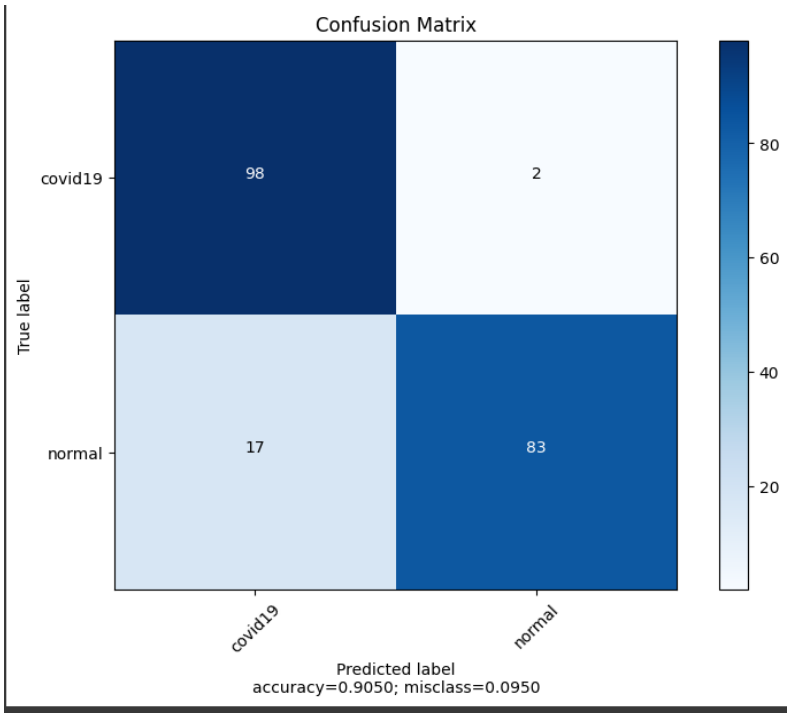
Resultado de DenseNet sin Transfer Learning

```
[97] print(classification_report(y_true_list, y_pred_list))
0s
                    precision    recall  f1-score   support

                 0       0.85      0.98      0.91       100
                 1       0.98      0.83      0.90       100

          accuracy                           0.91       200
         macro avg       0.91      0.91      0.90       200
      weighted avg       0.91      0.91      0.90       200
```

Nota. Al cambiar el grow rate y el kernel size en la arquitectura DenseNet201 se nota un cambio en los resultados tanto en precision, exactitud.

Figura 57

Matriz de confusión para DenseNet201 sin Transfer Learning



Nota. La tasa de falsos negativos llega a ser 17 por tanto la tasa de verdaderos negativos disminuye a 83, lo cual afecta al cálculo de la exactitud que viene a ser 91%.

Tabla 5

Complejidad del modelo DenseNet201 sin Transfer Learning

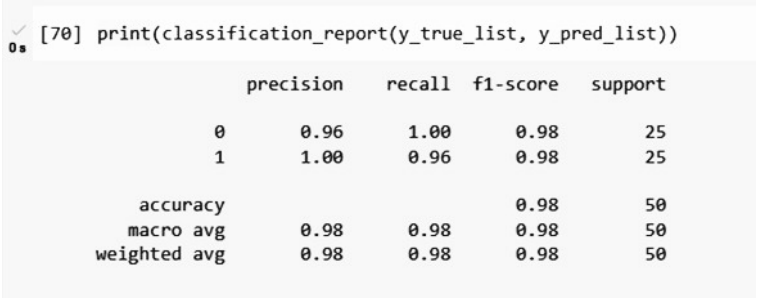
Medida	Cantidad
Complejidad Computacional	8.04 GFLOPs
Número de parámetros	40.31 M

Nota. El número de parámetros aumenta exponencialmente cuando se hace una modificación en el grow rate a 48 y un cambio en el tamaño del kernel.

Entrenamiento de la propuesta DenseNet201 pre-entrenado con max-pooling 2x2 en la primera convolución y max-pooling 3x3 y stride 2 en los “Transition Layers”, con un batch size de 4 en la validación.

Figura 58

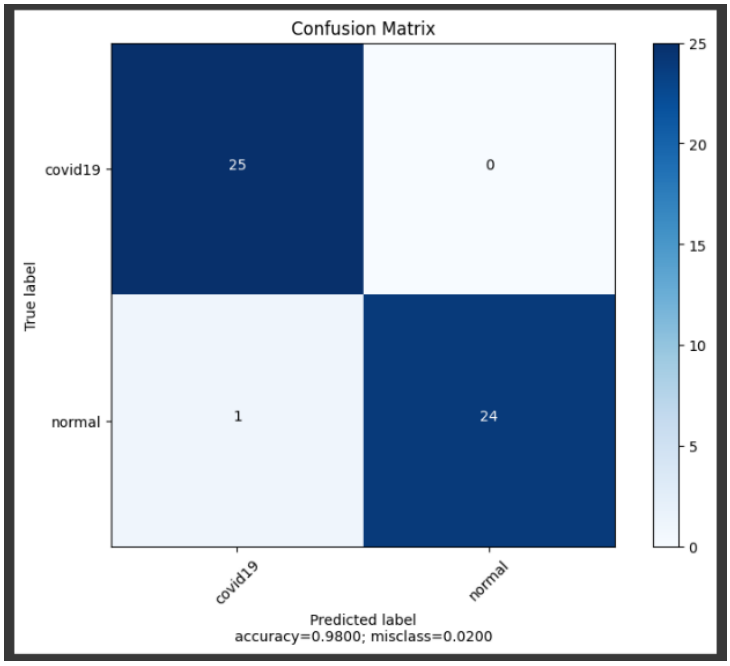
Métricas de Clasificación del Modelo Propuesto



Nota. La exactitud que se logra con el modelo propuesto viene a ser 98%, haciendo uso de un batch size de 4 en la validación, la exactitud puede variar entre 97% y 96% si el batch size es de 8.

Figura 59

Matriz de confusión para el Modelo propuesto



Nota. El total entre la tasa de verdaderos positivos más la tasa de falsos positivos es de 25 debido a que se usa un batch size de 4 en el proceso de validación.

Tabla 6

Complejidad del modelo Propuesto

Medida	Cantidad
Complejidad Computacional	3.24 GFLOPs
Número de parámetros	18.1 M

Nota. Se observa que el número de parámetros es similar al del DenseNet201 tradicional, pero a nivel de complejidad computacional hay una diferencia de 0.5GFLOPs.

Entrenamiento de la propuesta DenseNet201 usando el Dataset propio de 120 imágenes. El procedimiento que se siguió para el entrenamiento fue usar 200 imágenes normales del Dataset Chest-X-Ray y añadir 100 imágenes que vendrían a representar

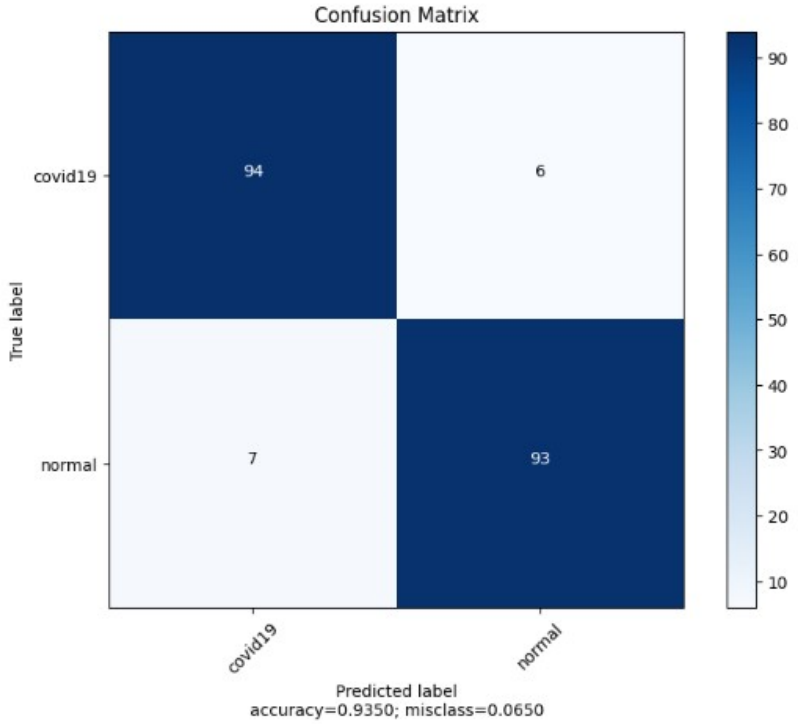
un promedio del 80% del Dataset propio - MSP y el resto para validación, el resultado obtenido fue el siguiente:

Figura 60

Resultado del Entrenamiento Usando el Dataset Propio

```
[60] print(classification_report(y_true_list, y_pred_list))
```

	precision	recall	f1-score	support
0	0.93	0.94	0.94	100
1	0.94	0.93	0.93	100
accuracy			0.94	200
macro avg	0.94	0.94	0.93	200
weighted avg	0.94	0.94	0.93	200



Nota. La exactitud que se obtiene usando el dataset propio es de 94% usando un batch size de 8 para la validación, la complejidad computacional se mantiene igual debido a que se usa la arquitectura propuesta.

Tabla 7

Resumen del resultado de las pruebas

Modelo	Accuracy	Numero de Parametros(M)	GFLOPs
DenseNet121 pre-entrenado 50 epochs	95%	6.96	0.27
DenseNet121 pre-entrenado 100 epochs	96.50%	6.96	0.27
DenseNet169 pre-entrenado 80 epochs	92%	12.49	2.96
DenseNet201 pre-entrenado	97%	18.1	3.74
DenseNet201 grow rate 48	91%	40.31	8.04
(propuesta)DenseNet201 usando Dataset Propio	94%	18.1	3.24
(propuesta)DenseNet201 Transition max-pooling	**98%**	**18.1**	**3.24**

Tabla 8

Comparación con otras Investigaciones

Modelo	Accuracy	Parametros	GFLOPS
Narin et al. ResNet50	96.10%		
Afshar et al. COVID-CAPS	95.70%		
Rahaman et al. VGG19	89.30%	143	
Xu et al. MANet	96,32%	23.51	1.059
Sahinbas & Catak VGG16	80%		3.8
Wang et al. MCFF-Net	94,66%	45.78	5.9
(propuesta)DenseNet201	**98%**	**18.1**	**3.24**

Nota. Algunos campos vacíos se deben a que no se encontraron esos datos en dichas investigaciones, se considera accuracy para la comparación por motivos más prácticos.

CAPITULO VI: DISCUSIÓN

En esta investigación, se propuso un modelo basado en la arquitectura DenseNet201 para la detección de COVID-19 en imágenes radiográficas del torax. A través de la experimentación y evaluación, el modelo propuesto ha demostrado su eficacia, obteniendo un 98% de precisión en la clasificación de imágenes.

Los resultados de las pruebas muestran que la arquitectura DenseNet es efectiva para la clasificación de imágenes médicas en términos de exactitud. Al comparar el modelo propuesto con los estudios anteriores se observa algunas diferencias y mejoras, en algunos casos el modelo propuesto terminó superando en cuando a la precisión y rendimiento. Por ejemplo, Narin et al. (2021) utilizó una arquitectura basada en ResNet50 y obtuvo una precisión del 96.10% lo cual es menor en comparación con el 98% de precisión obtenida en la propuesta. Afshar et al. (2020) utilizó COVID-CAPS y al igual a esta investigación propone un nuevo modelo basado en otra arquitectura la cual también mejora en aspectos como número de parámetros y obteniendo una precisión del 95.7% la cual a su vez también ha sido una de las investigaciones que inspiró la realización de este trabajo, por otro lado, Rehman et al. (2020) obtuvo una efectividad de 98.75% con 20M de parámetros que viene a ser una efectividad un poco más alta y mayor número de parámetros, con ello se observa una mejora en la exactitud de los modelos a medida que se aumenta la complejidad computacional y el número de parámetros.

Además, se observa que el uso de transfer learning con pre-entrenamiento en el conjunto de datos de ImageNet ayuda a mejorar la precisión del modelo en comparación con el modelo sin transfer learning. Por ejemplo, el modelo DenseNet121 pre-entrenado

obtiene una exactitud del 95% después de 50 épocas de entrenamiento, mientras que el modelo DenseNet201 sin pre-entrenamiento solo obtiene una exactitud del 91%.

Así mismo, en comparación con el modelo propuesto por Sahinbas & Catak (2021) que usa VGG16 y obtiene una precisión del 80% con 3.8 GFLOPs el modelo propuesto obtiene unos mejores puntajes tanto en precisión y eficiencia del modelo también con la comparación entre Wang MCFF-Net66-Conv1-GAP que tiene 94,66% de precisión con 45.78M de parámetros y 5.9GFLOPs.

En cuanto a las mejoras realizadas a la arquitectura DenseNet201, se ha demostrado que cambiar el max-pooling de 3x3 a 2x2 en la primera capa de convolución y el cambio de Average Pooling 2x2 a Max pooling 3x3 stride 2 en las capas de transición resultaron en un rendimiento mejorado, además viéndose validado con una dataset propio de esta investigación.

En conclusión, los resultados de las pruebas demuestran que la arquitectura DenseNet es una buena opción para la clasificación de imágenes médicas y que el uso de transfer learning con pre-entrenamiento en ImageNet puede ayudar a mejorar la precisión del modelo. Además, la elección de la arquitectura del modelo debe basarse en un equilibrio entre la precisión y la complejidad computacional, dependiendo de los recursos disponibles y las necesidades específicas de la aplicación.

Aunque este estudio demuestra una alta efectividad con lo propuesto, se puede decir que aún se puede mejorar, validar y madurar este modelo. También es importante recordar que estos modelos de aprendizaje automático no sustituyen a los profesionales, sino que los complementan en su diagnóstico, proporcionándoles una herramienta adicional para tomar decisiones más informadas y rápidas.

CONCLUSIONES

Esta investigación propone un nuevo modelo de clasificación basado en Deep learning usando la arquitectura DenseNet201 para la detección de COVID 19, modificando la arquitectura en la primera fase de convolución, donde se cambia el comportamiento a un Max Pooling2x2 para mantener más información detallada y preservar características importantes, asimismo en los "Transition Layer" se cambió de un Average Pooling 2x2 a un Max Pooling 3x3 para reducir la complejidad computacional del modelo, llegando a tener una efectividad que se ve respaldada por los resultados obtenidos, también se generalizó el modelo haciendo uso de un Dataset propio donde el resultado se encuentra por encima de otras propuestas del estado del arte, fiable y de bajo costo.

- Se evaluó la eficiencia del nuevo modelo de clasificación para la detección de COVID 19, llegando a obtener 18.1M de parámetros y 3.24GFLOPs, lo cual a comparación a un modelo tradicional DenseNet201 llega a tener menos 0.5GFLOPs, además está por encima de otras propuestas, por lo tanto, nuestra propuesta se considerada eficiente.

- Se evaluó la eficacia del nuevo modelo propuesto de clasificación para la detección de COVID 19, realizando las configuraciones necesarias para el entrenamiento y evaluación llegando a obtener un 98% de accuracy, 100% de precisión, 96% de Recall y 98% de F1-score los cuales están por encima de otras propuestas del estado del arte, por lo tanto, nuestra propuesta se considera eficaz.

RECOMENDACIONES

- Se sugiere que el modelo propuesto sea probado en una mayor cantidad de datos para ampliar la validez del modelo y para tener una mayor confianza en la efectividad del modelo.

- Se recomienda entrenar el modelo con un equipo de mayor capacidad computacional en GPU con el cual se pueda llevar a cabo mayor cantidad de épocas y mejorar el rendimiento del modelo, asimismo para realizar cambios más drásticos en la arquitectura.

- Se sugiere que se realice una comparación exhaustiva con otros modelos de clasificación y detección de COVID-19 basados en Deep Learning, utilizando los mismos datasets para tener una evaluación justa y exhaustiva.

- Se sugiere la realización de estudios futuros para evaluar la efectividad del modelo propuesto en la detección de otras enfermedades respiratorias, lo que podría ampliar el uso del modelo y tener una aplicación más amplia en la medicina.

- Se recomienda usar el modelo en otras aplicaciones prácticas de visión por computadora en el cual se pueda medir el rendimiento de este en dichas situaciones, este solamente es un caso aplicado sin embargo se puede usar en otros campos y mejorar el modelo.

REFERENCIAS BIBLIOGRÁFICAS

Afshar, P., Heidarian, S., Naderkhani, F., Oikonomou, A., Plataniotis, K. N., & Mohammadi, A. (2020). *COVID-CAPS: A Capsule Network-based Framework for Identification of COVID-19 cases from X-ray Images*. http://arxiv.org/abs/2004.02696

Apicella, A., Tegolo, D., Valenti, C., Prevete, R., le Dinh, T., Lee, S.-H., Kwon, S.-G., & Kwon, K.-R. (2022). COVID-19 Chest X-ray Classification and Severity Assessment Using Convolutional and Transformer Neural Networks. *Applied Sciences 2022, Vol. 12, Page 4861*, *12*(10), 4861. https://doi.org/10.3390/APP12104861

Barrios, J. I. (2019, July 26). *La matriz de confusión y sus métricas – Inteligencia Artificial –*. https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

Caya Pérez, J. C. (2020). *evaluación de modelos de redes neuronales convolucionales aplicado a radiografías de tórax, para apoyar al proceso de diagnóstico de neumonía asociada al covid-19*. universidad ricardo palma.

Centros para el control y la prevención de enfermedades. (28 de Setiembre de 2022). Obtenido de Centros para el control y la prevención de enfermedades*:* https://espanol.cdc.gov/coronavirus/2019-ncov/symptoms-testing/testing.html#:~:text=Hay%20dos%20tipos%20principales%20de,prueba%20por%20sobre%20la%20otra.

Cohen, J. P., Morrison, P., Dao, L., Roth, K., Duong, T. Q., & Ghassemi, M. (2020). *COVID-19 Image Data Collection: Prospective Predictions Are the Future*. http://arxiv.org/abs/2006.11988

de Moura, J., García, L. R., Lizancos Vidal, P. F., Cruz, M., López, L. A., Lopez, E. C., Novo, J., & Ortega, M. (2020). Deep convolutional approaches for the analysis of Covid-19 using chest X-ray images from portable devices. *IEEE Access*, *8*, 195594–195607. https://doi.org/10.1109/ACCESS.2020.3033762

Edouard Mathieu, H. R.-G.-O. (2020). *OurWorldInData.org*. Obtenido de OurWorldInData.org*: https://ourworldindata.org/coronavirus/country/peru*

DenseNet Explained | Papers With Code. (n.d.). Retrieved January 26, 2023, from https://paperswithcode.com/method/densenet

Devopedia. (2021, April 7). *ImageNet*. https://devopedia.org/imagenet

Donges Niklas. (2022, September 12). *What Is Transfer Learning? A Guide for Deep Learning | Built In*. https://builtin.com/data-science/transfer-learning

El modelo de redes neuronales - Documentación de IBM. (2021, August 17). https://www.ibm.com/docs/es/spss-modeler/saas?topic=networks-neural-model

Galia Puerto. (2019, September 12). *Eficacia, eficiencia y efectividad - Galia Puerto*. https://galiapuerto.es/eficacia-eficiencia-y-efectividad-que-las-diferencia/

GESTIÓN, N. (2022). *La diferencia entre eficiencia y eficacia | nnda nnlt | ECONOMIA | GESTIÓN*. https://gestion.pe/economia/management-empleo/eficiencia-eficacia-diferencias-eficaz-eficiente-significado-conceptos-nnda-nnlt-249921-noticia/

Harikishan NB. (2019, December 10). *Confusion Matrix, Accuracy, Precision, Recall, F1 Score | by Harikrishnan N B | Analytics Vidhya | Medium*. https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd

Hernández Sampieri, R., & Mendoza Torres, C. P. (2018). *Metodología de la investigación : las rutas cuantitativa, cualitativa y mixta*.

ImageNet. (2021, March 11). *ImageNet*. https://www.image-net.org/

Juan Barrios. (2023). *Redes neuronales convolucionales son un tipo de redes neuronales*. https://www.juanbarrios.com/redes-neurales-convolucionales/

Medinaceli Díaz, K. I., Silva Choque, M. M., Medinaceli Díaz, K. I., & Silva Choque, M. M. (2021). Impacto y regulación de la Inteligencia Artificial en el ámbito sanitario. *Revista IUS*, *15*(48), 77–113. https://doi.org/10.35487/RIUS.V15I48.2021.745

MINSA. (2022, June 22). *Cuáles son las pruebas para saber si tienes COVID-19 - Orientación - Ministerio de Salud - Gobierno del Perú*. https://www.gob.pe/9801-cuales-son-las-pruebas-para-saber-si-tienes-covid-19

Mooney, P. (2018). *Chest X-Ray Images (Pneumonia) | Kaggle*. https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia

Narin, A., Kaya, C., & Pamuk, Z. (2021). Automatic detection of coronavirus disease (COVID-19) using X-ray images and deep convolutional neural networks. *Pattern Analysis and Applications*, *24*(3), 1207–1220. https://doi.org/10.1007/s10044-021-00984-y

NCIRD. (2022, September 28). *Pruebas de detección del COVID-19: información importante | CDC*. https://espanol.cdc.gov/coronavirus/2019-ncov/symptoms-testing/testing.html

¿Qué es una red neuronal? Guía de IA y ML - AWS. (2023). https://aws.amazon.com/es/what-is/neural-network/

RAE. (2022a). *efectividad | Definición | Diccionario de la lengua española | RAE - ASALE*. https://dle.rae.es/efectividad

RAE. (2022b). *eficacia | Definición | Diccionario de la lengua española | RAE - ASALE*. https://dle.rae.es/eficacia

RAE. (2022c). *eficiencia | Definición | Diccionario de la lengua española | RAE - ASALE*. https://dle.rae.es/eficiencia

Rahaman, M. M., Li, C., Yao, Y., Kulwa, F., Rahman, M. A., Wang, Q., Qi, S., Kong, F., Zhu, X., & Zhao, X. (2020). Identification of COVID-19 samples from chest X-Ray images using deep learning: A comparison of transfer learning approaches. *Journal of X-Ray Science and Technology*, *28*(5), 821–839. https://doi.org/10.3233/XST-200715

Rahman, T., Chowdhury, M., & Khandakar, A. (2020). *COVID-19 Radiography Database | Kaggle*. https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database

Rehman, A., Naz, S., Khan, A., Zaib, A., & Razzak, I. (2020). *Improving Coronavirus (COVID-19) Diagnosis using Deep Transfer Learning*. https://doi.org/10.1101/2020.04.11.20054643

RSNA. (2022, November 1). *Rayos X del tórax* (G. Balint, B. Antala, C. Carty, J.-M. A. Mabieme, I. B. Amar, & A. Kaplanova, Eds.). Radiological Society of North America; Uniwersytet Śląski. Wydział Matematyki, Fizyki i Chemii. https://doi.org/10.2/JQUERY.MIN.JS

RTVE. (04 de Junio de 2021). PorRTVE. Obtenido de PorRTVE: https://www.rtve.es/noticias/20210604/paises-muertos-coronavirus-poblacion/2012350.shtml

Sahinbas, K., & Catak, F. O. (2021). Transfer learning-based convolutional neural network for COVID-19 detection with X-ray images. In *Data Science for COVID-19 Volume 1: Computational Perspectives* (pp. 451–466). Elsevier. https://doi.org/10.1016/B978-0-12-824536-1.00003-4

sanofi campos. (2020, July 12). *La aplicación de la Inteligencia Artificial en salud | Campus Sanofi*. https://campus.sanofi.es/es/noticias/2021/inteligencia-artificial-salud

Santos Manuel. (2018, April 7). *FLOPS: qué son las operaciones de coma flotante y por qué importan en tu tarjeta gráfica*. https://hardzone.es/2018/04/07/operaciones-en-coma-flotante-flops/

SAS. (2023). *What is Deep Learning and Why It Matters? | SAS*. https://www.sas.com/en_us/insights/analytics/deep-learning.html

Sooryakiran Pallikulathil. (2021, December 21). *neural networks - Does higher FLOPS mean higher throughput? - Artificial Intelligence Stack Exchange*. https://ai.stackexchange.com/questions/33856/does-higher-flops-mean-higher-throughput

UNICEF Perú. (2021, March). *COVID–19: Impacto de la caída de los ingresos de los hogares en indicadores de niñez y adolescencia | UNICEF*. https://www.unicef.org/peru/informes/covid19-impacto-de-la-caida-de-los-ingresos-de-los-hogares-en-indicadores-de-ninez-y-adolescencia

Vasudev Rakshith. (2019, February 11). *Understanding and Calculating the number of Parameters in Convolution Neural Networks (CNNs) | by Rakshith Vasudev | Towards Data Science*. https://towardsdatascience.com/understanding-and-calculating-the-number-of-parameters-in-convolution-neural-networks-cnns-fc88790d530d

Wang, W., Li, Y., Li, J., Zhang, P., & Wang, X. (2021). Detecting COVID-19 in Chest X-Ray Images via MCFF-Net. *Computational Intelligence and Neuroscience, 2021*. https://doi.org/10.1155/2021/3604900

wikepedia. (2022, October 31). *Coma flotante - Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/Coma_flotante

Xu, Y., Lam, H. K., & Jia, G. (2021). MANet: A two-stage deep learning method for classification of COVID-19 from Chest X-ray images. *Neurocomputing*, *443*, 96–105. https://doi.org/10.1016/j.neucom.2021.03.034

Zhuang, L. (2022). *liuzhuang13/DenseNet: Densely Connected Convolutional Networks, In CVPR 2017 (Best Paper Award).* https://github.com/liuzhuang13/DenseNet

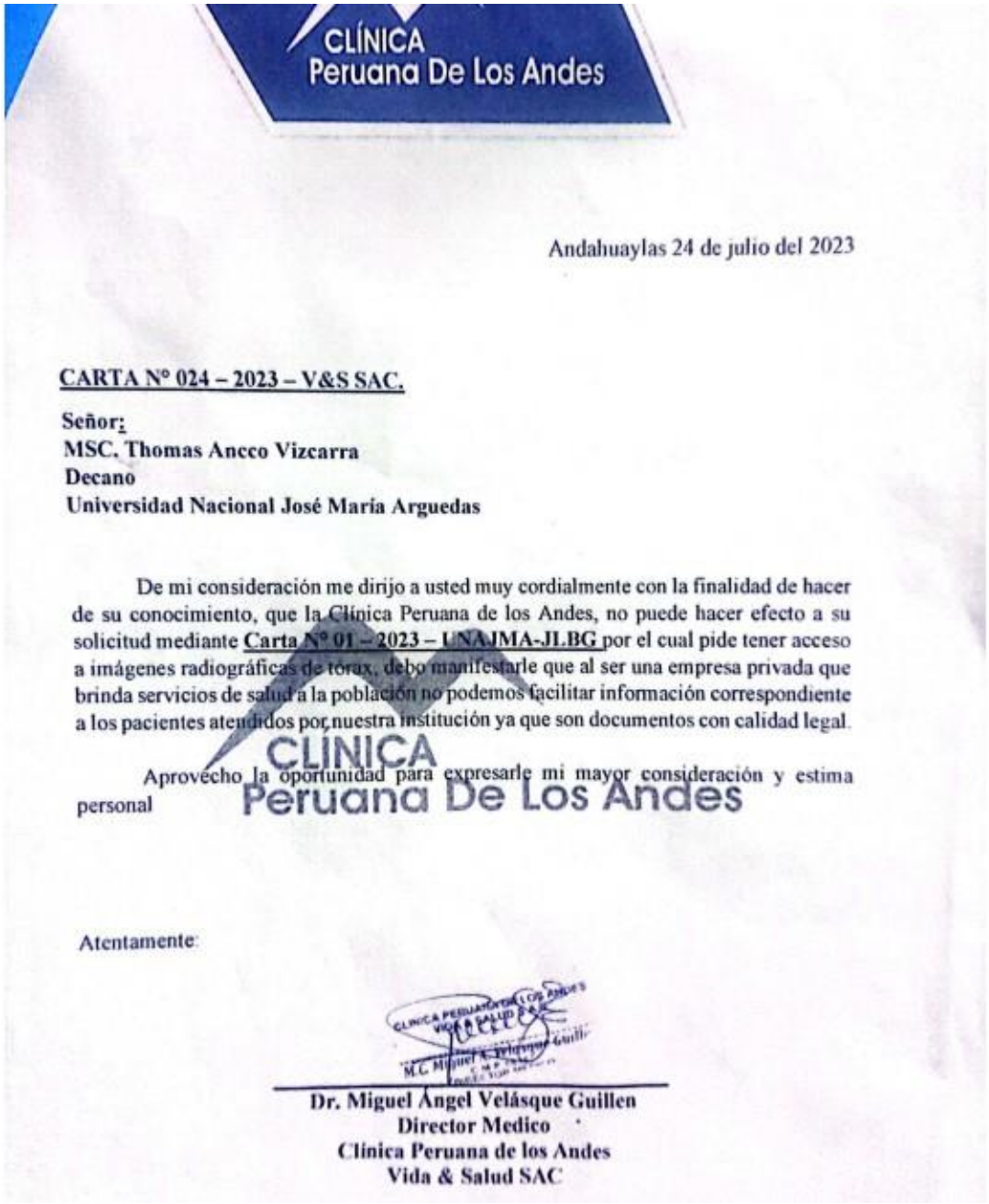
ANEXOS

Anexo 1. Código Fuente del Proyecto

Para visualizar el código fuente y Dataset de la investigación, revisar el siguiente link:

https://github.com/lucho19jose/COVID-Detection-DenseNet201

Anexo 2. Respuesta a la carta para la obtención de imágenes radiográficas en Andahuaylas



CLÍNICA
Peruana De Los Andes

Andahuaylas 24 de julio del 2023

CARTA N° 024 – 2023 – V&S SAC.

Señor:
MSC. Thomas Ancco Vizcarra
Decano
Universidad Nacional José María Arguedas

De mi consideración me dirijo a usted muy cordialmente con la finalidad de hacer de su conocimiento, que la Clínica Peruana de los Andes, no puede hacer efecto a su solicitud mediante **Carta N° 01 – 2023 – UNAJMA-JLBG** por el cual pide tener acceso a imágenes radiográficas de tórax, debo manifestarle que al ser una empresa privada que brinda servicios de salud a la población no podemos facilitar información correspondiente a los pacientes atendidos por nuestra institución ya que son documentos con calidad legal.

Aprovecho la oportunidad para expresarle mi mayor consideración y estima personal

CLÍNICA
Peruana De Los Andes

Atentamente:

Dr. Miguel Ángel Velásque Guillen
Director Medico
Clínica Peruana de los Andes
Vida & Salud SAC