Diagnóstico de neumonía por clasificación de imágenes de rayos X de tórax utilizando el método de redes neuronales convolucionales (CNN)

Francia Lorena Hernández Carmona
Especialización en estadística aplicada

Universidad® Católica de Manizales
VIGILADA MINEDUCACIÓN
Obra de Iglesia de la Congregación
Hermanas de la Caridad Dominicas de La Presentación de la Santísima Virgen

Diagnóstico de neumonía por clasificación de imágenes de rayos x de tórax utilizando el método de redes neuronales convolucionales (CNN)

Trabajo de grado presentado como requisito para optar al título de la especialización en estadística aplicada

Asesor

Rubén Darío Lara Escobar

Autor
Francia Lorena Hernández Carmona

Universidad católica de Manizales
Facultad de educación
Especialización en estadística aplicada
2023

Diagnóstico de neumonía por clasificación de imágenes de rayos x de tórax utilizando el método de redes neuronales convolucionales (CNN).
Diagnosis of pneumonia by classification of chest x-ray images using the method of convolutional neural networks (CNN)

Autores (Hernández Carmona Francia Lorena)[1]
Francia.hernandez1@ucm.edu.co
Asesor (Rubén Darío Lara Escobar)[2]
rlara@ucm.edu.co

Resumen

El presente trabajo es el proyecto final de la especialización de estadística aplicada que tiene como objetivo implementar y validar un algoritmo para detectar posible neumonía en imágenes de rayos X de tórax, por medio del modelo de redes neuronales convolucionales (CNN). Para desarrollar este algoritmo se utilizó Jupyter notebook de Python 3 con las librerías pandas y Numpy para manipulación de la base de datos y tensorflow para modelar el proceso de aprendizaje. Un dataset de 2.542 imágenes divididas en carpetas de entrenamiento, prueba y validación clasificadas en dos categorías (con neumonía, sin neumonía) suministradas de forma libre de pacientes pediátricos entre 1 a 5 años de edad. Los resultados de precisión arrojados por el modelo para el dataset de entrenamiento fueron muy significativos presentando un 98 % de probabilidad de detección de la patología, mientras que en la prueba de validación el modelo presentó un porcentaje por debajo del 80% . Basado en los resultados obtenidos el algoritmo tiene gran potencial en el futuro para mejorar los procesos de diagnóstico por imágen.

*Palabras clave***:** Neumonía, imágenes diagnósticas, redes neuronales convolucionales (**CNN**)

Abstract

This work is the final project of the specialization in applied statistics, which aims to implement and validate an algorithm to detect possible pneumonia in chest X-ray images, through the model of convolutional neural networks (CNN). To develop this algorithm, a Python 3 Jupyter notebook was used with the pandas and Numpy libraries for database manipulation and tensorflow to model the learning process. A dataset of 2,542 images divided into training, test, and validation folders, classified into two categories (with pneumonia, without pneumonia) freely provided by pediatric patients between 1 to 5 years of age. The precision results thrown by the model for the training dataset were significant, presenting a 98% probability of detection of the pathology, while in the validation test the model presented a percentage below 80%. Based on the results obtained, the algorithm has great potential in the future to improve diagnostic imaging processes.

Keywords: Pneumonia, diagnostic imaging, convolutional neural networks (CNN)

Introducción

La inteligencia artificial (IA) se ha convertido en una disciplina clave para la transformación y avance del área de imagenología, se menciona que "ha generado un cambio de paradigma en las formas de procesamiento de la información y ha llegado a superar la capacidad de deducción y observación humana"(Enzo Raschio et al., 2021). La (IA) utiliza algoritmos y modelos estadísticos precisos basados en un conjunto de datos que permite cumplir con los objetivos de "Apoyo al médico radiólogo, optimización de la imagen, reconocimiento de estructuras, segmentación de lesiones y transcripción de informes"(Aguirre et al., 2021).

Es evidente que el área de imágenes diagnósticas ha tenido un desarrollo tecnológico significativo en los últimos años, donde se destaca la utilización de este tipo de aplicación computacional, pero aún se presentan dificultades para analizar profundamente múltiples detalles que contiene la imagen médica debido a que "En numerosas ocasiones tiene una gran influencia subjetiva, porque generalmente se basa en la extracción de una determinada información o elementos simples sobre fondos complejos"(Martínez et al., 2016). Sumado al tiempo de respuesta por el alto volumen de estudios, información del paciente, almacenamiento y gestión realizados por día con diferentes equipos y técnicas.

Razón por la que surge el objetivo de implementar y validar un algoritmo que permita dar una posible solución al análisis cuantitativo de la imágen para un apoyo futuro al equipo médico de imágenes diagnósticas.

El proyecto aplicativo contiene tanto la sección teórica como la implementada por software. En la parte teórica se hará una pequeña introducción de inteligencia artificial y posteriormente se explicará brevemente la formación de una imagen médica, así como el modelo que se utilizará para la clasificación del dataset de imágenes para detección de posible neumonía.

En la implementación por software se mostrará el modelo de redes neuronales convolucionales utilizado para realizar clasificación de imágenes de tórax pediátricas para detección de infección por neumonía o no, basado en entrenamiento y validación de un volumen significativo de imágenes.

Contexto

De acuerdo con la OMS (2022) "La neumonía es la principal causa individual de mortalidad infantil en todo el mundo" (p.1). Se presenta "En niños de 1 a 59 meses de edad y anualmente representa más de 800 muertes en todo el mundo"(Rees et al., 2020). Si bien la vacunación es un mecanismo de impacto para reducir la mortalidad por problemas asociados al aparato respiratorio, existe una variedad de agentes virales que pueden desembocar una infección más compleja como las bacterias predominantes Streptococcus pneumoniae (neumococo) "Con 20 serotipos que pueden causar enfermedad grave, pero cuya frecuencia varía según el grupo de edad y la ubicación geográfica "(ministerio procteccion social colombia, 2014).

Uno de los métodos utilizados para el diagnóstico de neumonía es por medio de la radiografía de tórax y la calidad de su detección depende de múltiples factores como el posicionamiento del paciente, la proyección y los parámetros de radiación utilizados por el tecnólogo, en ocasiones su efectividad de diagnóstico se puede ver comprometida debido "A las técnicas para identificar la etiología que carecen de adecuada especificidad y sensibilidad"(ministerio procteccion social colombia, 2014). Para diferenciar una infección viral de una bacteriana. Por esta razón la medicina ha incorporado a sus procesos el uso de la inteligencia artificial (IA) en el área de las

imágenes médicas como apoyo a los especialistas radiólogos en la interpretación del diagnóstico al paciente.

Planteamiento del problema

Diferenciar un diagnóstico de neumonía con otro tipo de patologías que presentan síntomas relacionados no es una tarea fácil, requiere de habilidades y preparación propias del especialista. En su rutina diaria el radiólogo analiza cada imagen de forma cualitativa, proceso que carece de un factor numérico y estandarizado, que se limita a la interpretación, objetividad y experiencia de cada radiólogo. Situación que puede influir negativamente en la evaluación y diagnóstico del paciente. Para disminuir posibles errores en el diagnóstico es indispensable utilizar una herramienta computacional como lo es la inteligencia artificial (IA), que utiliza una visión artificial basada en procesos numéricos para filtrar y extraer características relevantes de las imágenes, para su posterior clasificación. Herramienta muy útil para el acompañamiento cuantitativo de interpretación, evaluación y toma de decisiones médicas.

Antecedentes

Las enfermedades respiratorias crónicas se encuentran entre las enfermedades no transmisibles más comunes en todo el mundo, en gran parte debido a la ubicuidad de exposiciones nocivas ambientales ocupacionales y conductuales (Labaki & Han, 2020). Los niños son especialmente vulnerables a este tipo de patologías, y afecta la vida de más de un millón de personas en todo el mundo, es la causa predominante de mortalidad y morbilidad (Zar & Ferkol, 2014). La neumonía es una de las afecciones respiratorias más comunes que puede complicarse sino es tratada a tiempo, se define como " Infección aguda del tracto respiratorio inferior, que produce dificultad respiratoria y con evidencia radiológica de infiltrado pulmonar agudo" (Andrés Martín et al., 2012).

De acuerdo con (Alzate et al., 2020), "las infecciones neumocócicas causan variedad de síndromes clínicos, incluidos neumonía, meningitis y bacteriemia, siendo el 10,94% muerte por neumonía durante el año 2005 al 2016 en Colombia".

Cuando se presenta alguna sospecha por neumonía los pacientes son remitidos a un procedimiento de rayos X de tórax debido a su facilidad y baja dosis de radiación como apoyo para el análisis del especialista.

Este método por imagen surgió en 1895 cuando se descubrió los rayos X por el físico alemán Wilhelm Conrad Röentgen, hallazgo que representó el comienzo de una nueva era en el diagnóstico médico por imágen, creando una de las ramas más utilizadas en la actualidad como es la radiografía, y se basa en la interacción de los rayos X con el paciente, por medio de un detector que recibe la radiación emitida por el equipo para formar una imagen que permite visualizar la estructura interna del cuerpo y diagnosticar posibles patologías sin realizar procedimientos invasivos.

A finales del siglo XIX las imágenes se formaban analógicamente por medio de películas elaboradas con cristales de plata, pero debido a que el material con que se elaboraban las láminas tenía un efecto negativo para el medio ambiente por su alta toxicidad fue reemplazado paulatinamente por una imagen digital que se obtiene mediante " la captura directa de la imágen para convertir los rayos X directamente a señales eléctricas "(Raudales Díaz, 2014).

La innovación tecnológica ha permitido que más personas accedan al servicio diagnóstico por medio de imágenes" En la medicina moderna la radiología convencional, la tomografía computarizada son procedimientos realizados frecuentemente para diagnosticar múltiples enfermedades "(Raudales Díaz, 2014).

En la actualidad el área de las imágenes médicas ha integrado a su procedimiento diagnóstico el uso de la inteligencia artificial: "está formada por una serie de algoritmos lógicos suficientemente entrenados a partir de los cuales las máquinas son capaces de tomar decisiones para casos concretos a partir de normas generales "(Ávila et al, 2020, p.1).

Herramienta que tiene como fundamento base, la estadística aplicada y que implementa algunos algoritmos de clasificación como el método de redes neuronales convolucionales (CNN).

Justificación

Diagnosticar adecuadamente al paciente representa grandes desafíos para las entidades de salud debido al gran volumen de estudios que se generan diariamente, interfiriendo en el proceso de análisis y tiempo de respuesta al paciente, esto sumado a la dificultad que se presenta al identificar claramente algunas enfermades como es el caso de la Neumonía que puede ser mal diagnosticada y asociarla con otras afecciones respiratorias.

El interés de este trabajo aplicativo es entender y explorar las capacidades de cómputo basado en modelos estadísticos para el servicio de la comunidad médica, así como la contribución de la ingeniería para potencializar el diagnóstico por adquisición de imagen, herramientas de procesamiento y desarrollo de nuevos dispositivos y mecanismos para la evolución continúa dirigida al bienestar del paciente.

Implementando el modelo de redes neuronales convolucionales (CNN) para clasificación de imágenes de neumonía, se puede lograr la obtención de información oculta, compleja y con visión cuantificada de una imagen radiológica como apoyo en el análisis del especialista, proceso que al ser integrado a la línea de trabajo médica reflejaría un sistema más optimo, eficiente y de mejor calidad para los pacientes.

Objetivo general

Implementar y entrenar un modelo de redes neuronales convolucionales (CNN) para clasificación de imágenes de rayos X de tórax con posible diagnóstico de neumonía.

Objetivos específicos

- Realizar un algoritmo en Jupyter notebook de Python 3 que permita clasificar imágenes de rayos X de tórax de pacientes diagnosticados con neumonía y pacientes sanos.
- Validar el modelo de redes neuronales convolucionales (CNN) utilizado para la clasificación de las imágenes de rayos X de tórax.

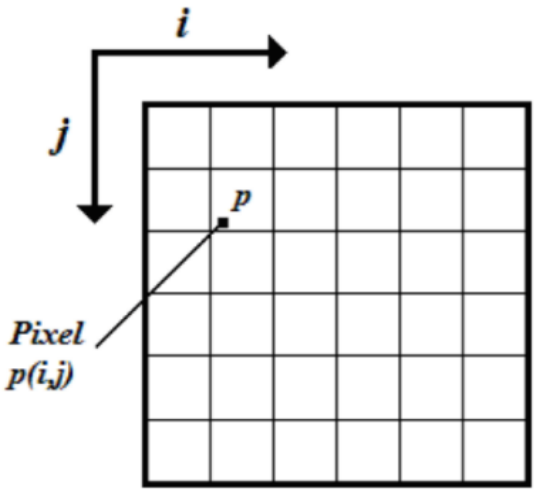
Marco de referencia

Imágen médica digital

Una imagen digital define como " Una representación bidimensional de una imagen utilizando bits (unos y ceros)" (Martínez et al., 2016). Se puede representar como una matriz de mxn donde n es el número de pixeles de ancho y m el número de pixeles de largo, siendo el pixel la unidad básica de toda imagen digital.

En el caso de una imagen radiográfica con equipo de rayos X convencional se puede obtener un *slice* o corte bidimensional en (2D) donde sus coordenadas espaciales, se denominan pixel, p (i, j), y es representado por un valor numérico asociado a la intensidad en la escala de grises. Para una imagen por tomografía computarizada (CT) se obtiene un *slice* en (3D) que se denomina vóxel siendo ésta la unidad cúbica que compone un objeto tridimensional y que aporta el dato de profundidad en la imagen.

Figura 1
Representación de una imagen digital



Nota: Landrove M.M(2017)

Fuente:https://www.researchgate.net/figure/Figura-216-Representacion-de-una-imagen-digital_fig8_317182752

Inteligencia artificial

La inteligencia artificial (IA) está formada por “Una serie de algoritmos lógicos suficientemente entrenados a partir de los cuales las máquinas son capaces de tomar decisiones para casos concretos a partir de normas generales”(Ávila-Tomás et al., 2021).

La incorporación de la inteligencia artificial (IA) en los procesos diagnósticos han tomado fuerza para mejorar la atención médica de los pacientes, “Las imágenes radiológicas, las preparaciones de anatomía patológica y los registros médicos electrónicos de los pacientes se están evaluando con aprendizaje automático”(Ávila-Tomás et al., 2021). Actualmente los expertos en aplicaciones de IA exploran nuevos métodos y modelos tanto en la fase asistencia como de diagnóstico y terapia del paciente.

Algoritmos de aprendizaje automático profundo

Redes Neuronales artificiales

Se define como sistemas inspirados en el comportamiento del sistema nervioso de los seres vivos, tratando de emular principalmente el comportamiento del cerebro. (Iván et al., 2013)**.** Las redes neuronales artificiales presentan características similares al cerebro “Aprenden de la experiencia, generalizan de ejemplos previos a ejemplos nuevos y abstraen las características principales de una serie de datos” (Olabe, n.d.).

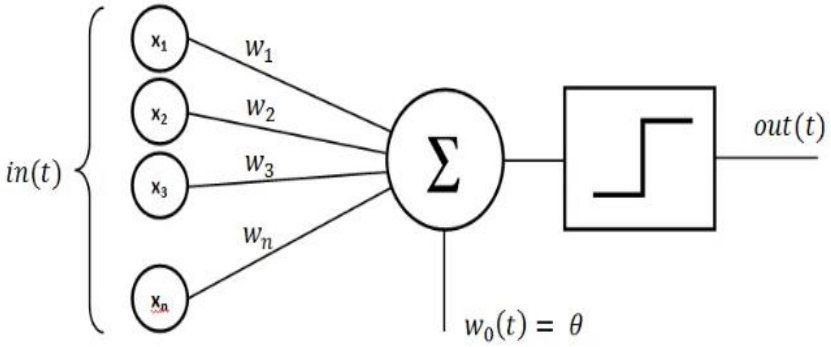
Perceptrón

"El perceptrón está diseñado para ilustrar algunas de las propiedades fundamentales de sistemas inteligentes"(Rosenblatt, 1958).

Es una unidad de red neuronal que realiza cálculos para detectar tendencias y patrones de una serie de datos de entrada. En la **figura 2** se muestra n entradas $(x_1, x_2 \dots x_n)$y cada una de ellas tiene asociada un peso $(w_1, w_2 \dots w_n)$que representa la importancia de cada una de las entradas con respecto a la salida, el peso otorgado depende del estudio ejecutado por el investigador.

Figura 2.

Estructura del perceptrón



La fórmula del perceptrón

Fuente: https://datascientest.com/es/perceptron-que-es-y-para-que-sirve

Función del perceptrón

El perceptrón se traduce a una función matemática. Es así como la entrada (x) se multiplica por los pesos (w) y nos arroja como salida un valor; si el valor es positivo y supera el umbral establecido la neurona se activa, en el caso contrario, si la salida toma un valor negativo y por debajo del umbral la neurona no se activa.

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad w = [w_1 \ w_2 \ w_3]$$

Vector de entrada Vector de pesos

Se realiza un producto escalar entre el vector de entrada y el vector de pesos (vectores $w \,.\, x$) luego de la suma de las multiplicaciones se suma el factor b que indica el umbral y controla el procesamiento de la unidad. Este proceso lo llamamos valor de activación.

$$\boldsymbol{z = (x_1 * w_1) + (x_2 * w_2) + (x_3 * w_3) + b}$$

$$\sum_{i=1}^{m} (w_1 x_2) + b$$

Ecuación (1) Valor de activación

Después de obtener el valor de activación, se realiza la función de activación φ

$$\boldsymbol{Y = \varphi(w.x)}$$

Ecuación (2) Función de activación

Se aplicó esta función debido a que era necesario que la neurona aprendiera por si sola y tomara valores diferentes a 0 y 1 este nuevo tipo de neurona es llamada neurona sigmoide.

Cuando se obtiene el valor de activación ecuación (1) se reemplaza en la función sigmoide. Es aquí donde se introduce la función sigmoide expresada de la siguiente manera:

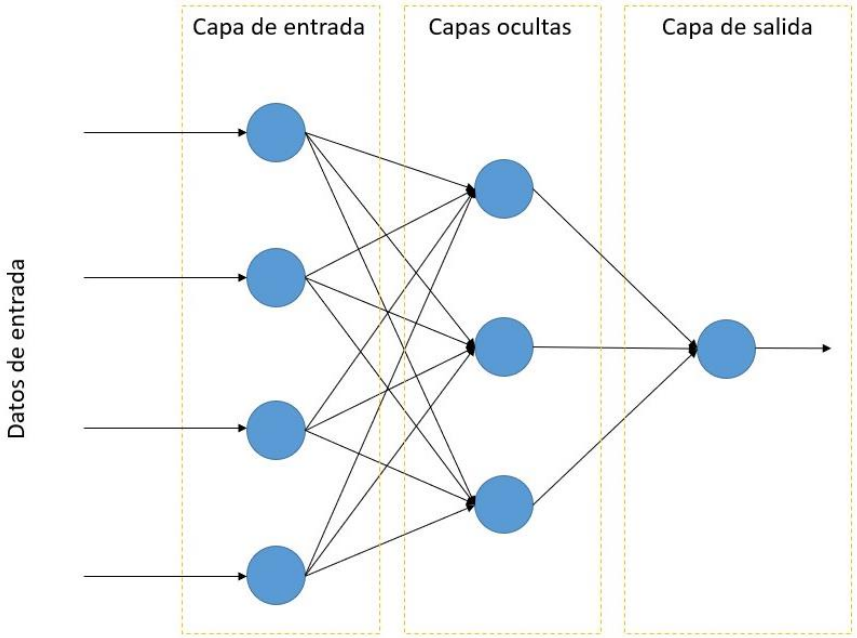
$$\boldsymbol{y = \varphi(z) = \frac{1}{1+e^{-z}}}$$

Ecuación (3) función sigmoide

Multiplayer perceptrón

El multiplayer perceptrón contiene más de una neurona, cuanta más cantidad de perceptrones más capas ocultas se obtienen. Las redes neuronales convolucionales organizan sus respectivas capas de forma jerárquica siendo las primeras especializadas en reconocer líneas y curvas, y las más profundas formas más intrincadas.

Figura 3.
Estructura multiplayer perceptrón



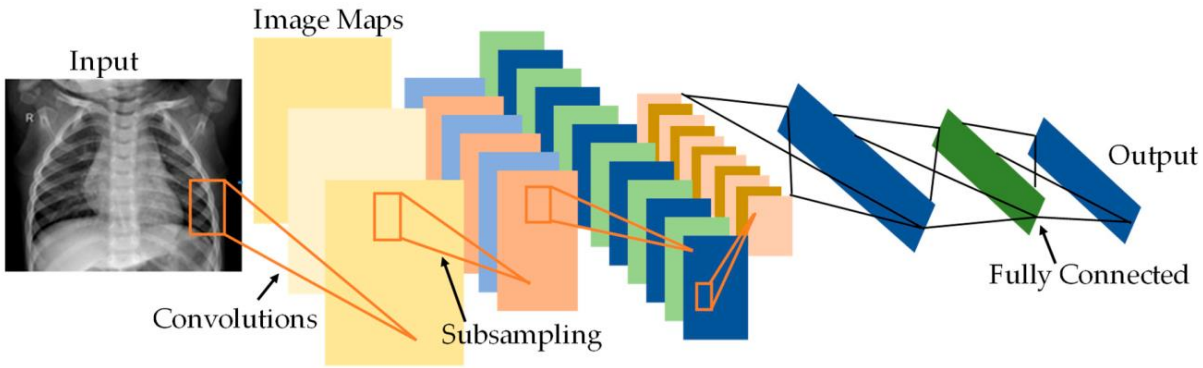
Fuente:https://interactivechaos.com/es/manual/tutorial-de-deep-learning/entrenamiento-de-una-red-neuronal

Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN) son muy utilizadas en el campo de la medicina para clasificación de imágenes, su aprendizaje se da de forma supervisada. “Las capas convolucionales en la red junto con los filtros ayudan a extraer el espacio y características temporales en una imagen”(Rahman et al., 2020) estas capas ocultas actúan analógicamente como un ojo humano para identificar patrones de una serie de datos.

El aprendizaje autónomo de la neurona debe tener un entrenamiento con gran volumen de imágenes para que su detección de características cuente con un nivel de confianza permitido, además de utilizar una técnica de pesos compartidos que optimiza el tiempo de cálculo.

Figura 4.
Arquitectura red neuronal convolucional



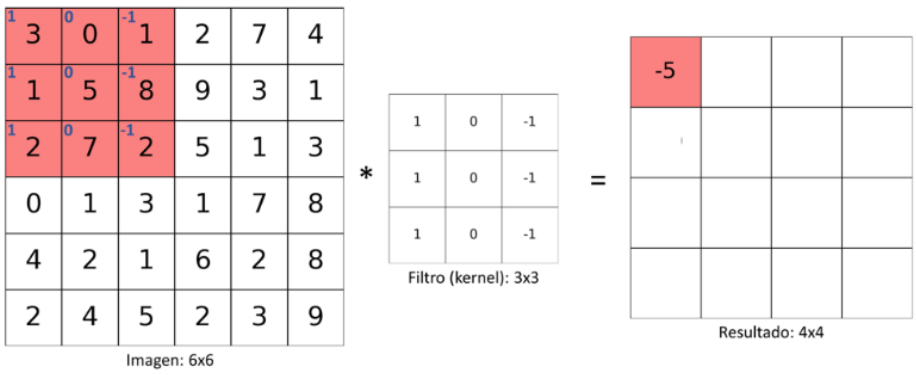
Nota: Imagen que representa la arquitectura de la red neuronal convolucional tomada de Transfer Learning with Deep Convolutional Neural Network (CNN) for Pneumonia Detection Using(p.3) por T. Rahman(Rahman et al., 2020) **2020,** Applied sciences

Las redes neuronales convolucionales (CNN) realizan una serie de cálculos y filtros internos que en la figura 4 se presenta como una segmentación de 3 bloques de construcción.

Convoluciones

La capa de convolución consiste en tomar grupos de pixeles cercanos de la imagen de entrada realizando un producto escalar que forma una pequeña matriz de filtro espacial o máscara llamada kernel y recorre toda la imagen de izquierda a derecha generando una nueva matriz de salida. La señal de entrada se traduce al número de pixeles que tiene la imagen, por ejemplo, si tenemos una imagen en escala de grises que sólo cuenta con un canal de salida y tiene $64X64$ pixeles usaría 4.096 neuronas, en caso de fuese a color se necesitan tres canales (Rojo, verde, azul) 64x64x3 el número de pixeles por los tres canales 12.288 neuronas. Cuando el kernel se desliza sobre los grupos de pixeles va formando una nueva imagen filtrada que sería la primera convolución y muestra un mapa de características de la imagen.

Figura 5.
Filtrado imágen 6 x 6



Nota: Sotaquirá M (marzo 30, 2019)

Fuente: https://www.codificandobits.com/blog/convolucion-redes-convolucionales/

En la figura 5 se observa una imagen de 6x6 con un kernel de 3x3 que recorre de arriba abajo y de izquierda a derecha toda la imagen, el proceso matemático que realiza es el siguiente:

Se multiplica uno a uno los valores de la imagen y el kernel, luego se procede a sumar

$$3.1 + 0.0 + (-1).1 + 1.1 + 0.5 + (-1).8 + 1.2 + 0.7 + (-1).2$$
$$= 3 + 0 - 1 + 1 + 0 - 8 + 2 + 0 - 2 = -5$$

Transformación de unidad lineal rectificada (RELU)

El rectificador es una función de activación que se aplica posterior a la primera capa de convolución, exactamente en la transformación de no linealidad al mapa de características. Está definida como:

$$f(x) = \max(0, x)$$

Donde X es la entrada de la neurona.

Esta función es análoga a la rectificación de media onda en electrónica y activa los pesos de una neurona es muy utilizada para obtener un proceso de menor error que los utilizados cuando se aplica la función logística además es más rápida.

Capa de agrupación (Subsampling)

La capa de agrupación es importante realizarla antes de aplicar una nueva convolución porque se encarga de ajustar la dimensionalidad, esto quiere decir que minimiza el número de parámetros en la entrada y la cantidad de neuronas que se produce por cada convolución, con el propósito de mejorar la eficiencia y el riesgo de sobreajuste situación que se presenta cuando "El modelo se ajusta demasiado bien a los datos de entrenamiento pero falla cuando se utiliza en un nuevo conjunto de datos" (Casal et al., 2021). Para evitar que el procesamiento sea muy robusto se aplica una reducción de las imágenes filtradas obteniendo las características más relevantes, esto permite que la convolución sea más profunda.

Subsampling con Max-pooling

El pooling reduce la dimensionalidad de los mapas de características "Transforma la representación de características conjuntas en información valiosa al mantener información útil y eliminando información irrelevante" (Gholamalinezhad & Khosravi, 2020) , un ejemplo de ello es que si tomamos 32 imágenes obtenidas de la primera convolución, con un número de pixeles de 28x28 y se recorre pixel por pixel se obtiene 25.088 neuronas, pero si se aplica un ajuste de los parámetros en la segunda convolución con max pooling de tamaño 2x2 (2 pixeles de alto por 2 pixeles de ancho) la imagen se reduce a la mitad dando como resultado 32 imágenes de 14x14 pixeles y 6.272 neuronas, si se aplica nuevamente una tercera convolución la salida que se obtiene será de la mitad, imágenes de tamaño 7X7 pixeles y 3.136 neuronas, como se observa se

reduce considerablemente el número de neuronas, resaltando así la información más relevante de características. La arquitectura de la red neuronal convolucional es jerárquica si bien la primera convolución Tiene la capacidad de detectar líneas o curvas, entre más capas de convoluciones se ejecuten los mapas de características podrán reconocer patrones más complejos.

Conexión con red neuronal tradicional

Luego de realizar cada convolución se procede a tomar la última capa de reducción de parámetros con subsampling, la cual tiene forma tridimensional (alto, ancho, mapas) para pasar por un proceso de aplanamiento donde deja de ser tridimensional y pasa a ser una capa de neuronas tradicionales, esta capa se trabaja con una función llamada softmax o función exponencial normalizada, que se emplea para comprimir un vector de K-dimensiones con valores reales en el rango de [0, 1]. La función softmax pasa de probabilidad entre 0 y 1 a neuronas de salida. En este caso si la salida es [0,6 0,4] nos indica que la primera opción tiene una probabilidad de 60% mientras la segunda es del 40 % y la cantidad de neuronas presentes en la última capa depende de lo que se esté clasificando.

Metodología

Base de datos

Para llevar a cabo la implementación del modelo de detección de imágenes fue necesario encontrar un dataset de imágenes suministrada de forma abierta por el centro médico de mujeres y niños de Guangzhou. Las imágenes son estudios de rayos X de tórax (anterior-posterior) realizadas a pacientes pediátricos de un rango de edades de 1 a 5 años. Radiografías que vienen en formato JPEG y que están clasificadas en tres carpetas que se dividen en prueba, entrenamiento y validación. Se trabaja con este dataset de imágenes porque su utilización es libre con fines investigativos y académicos, debido al gran desafío que se presenta para obtener grandes cantidades de imágenes por cuestiones de protección al paciente y confidencialidad. Para desarrollar el ejercicio práctico, utilizamos un total de 2542 imágenes divididas en dos categorías: paciente normal y paciente con neumonía distribuidas de la siguiente manera:

Entrenamiento (Train)

Para entrenamiento se utilizó el 80% de las radiografías con una cantidad de 1.041 imágenes de neumonía y 970 normales.

Prueba (Test)

Para la prueba se utilizó el 20 % de las radiografías con una cantidad de 313 imágenes de neumonía y 203 normales.

Figura 6. Radiografía de tórax con neumonía

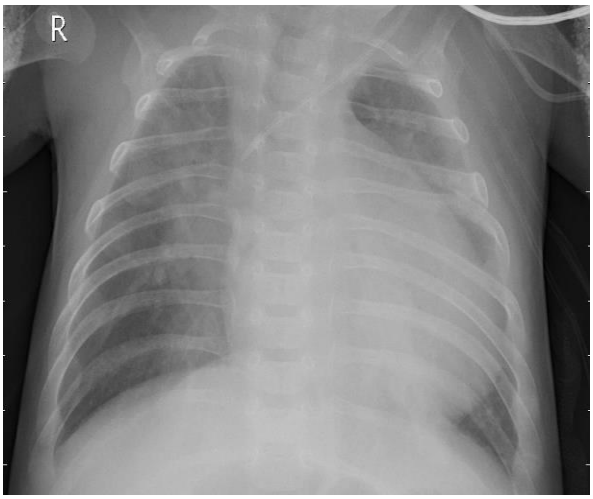
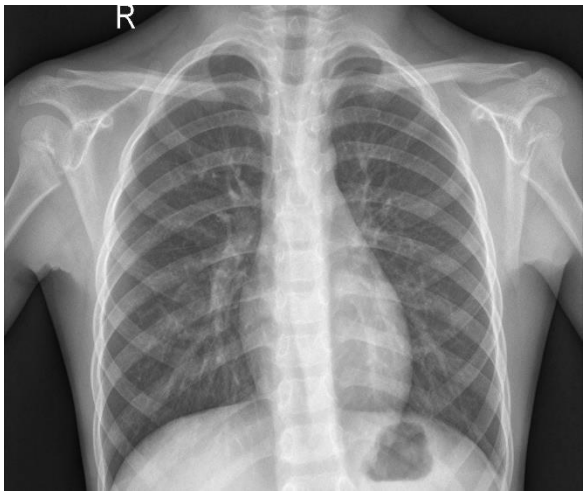


Figura 7. Radiografía de tórax normal



Fuente:https://www.kaggle.com/code/madz2000/pneumonia-detection-using-cnn-92-6-accuracy/data

Descripción de Algoritmo para detección de imágenes con posible neumonía utilizando el modelo (CNN) con herramienta Jupyter Notebook de Python 3

Python

Es un lenguaje de programación de alto nivel que utiliza código libre, es ampliamente utilizado en ciencia de datos, desarrollo de software y machine learning. Cuenta con librerías muy potentes para el análisis de datos y por esta razón se ha decidido realizar todo el código utilizando jupyter Notebook de Python 3 por su facilidad de manejo y soporte de la comunidad de programadores.

Para el análisis fue necesario instalar algunas librerías de gran importancia para trabajar con base de datos:

Librerías principales utilizadas en nuestro algoritmo

Imbalaced- learn (imlearn)

pip install imblearn

Es una librería muy útil para resolver el desequilibrio de datos en problemas de clasificación. Si se presenta un desbalance en la cantidad de imágenes de entrenamiento por categoría imlearn nos ayuda a evitar que el algoritmo arroje información errada de la categoría minorista.

Pandas (pd)

import pandas as pd

Es una librería muy importante para realizar análisis de datos, gracias a esta potente herramienta se pueden cargar, preparar, manipular, modelar y analizar una base de datos.

NumPy

import numpy as np

Del acrónimo Python numérico, es una librería ideal para realizar funciones matemáticas y numéricas, trabaja con procesamiento de matrices multidimensionales que mejora la eficiencia del análisis.

Tensorflow

import tensorflow as tf

Es una librería numérica rápida muy útil para modelar procesos de aprendizaje profundo.

Matplotlib

import matplotlib. pyplot as plt

Es una librería ideal para la visualización de datos en dos dimensiones.

Seaborn

import seaborn as sns

Es una librería basada en matplotlib que también permite crear elegantes gráficos para realizar análisis de datos.

Scikit-learn

Es una biblioteca para implementar aprendizaje automático y permite desarrollar análisis predictivo.

Descripción del algoritmo implementado

La base de datos es descargada y contiene la cantidad de imágenes por categoría en cada una de las tres carpetas procedemos a construir el algoritmo en Jupyter notebook instalando las librerías nombradas anteriormente, esenciales para poder manipular y analizar nuestro dataset de imágenes. Previo al análisis de las imágenes fue necesario implementar otro tipo de librería para realizar una transformación de dimensiones y color.

Figura 8.
Librerias

```
# Cargamos la libreria
# garantiza que sentencias futuras se ejecuten en versiones anteriores a la 2.1
from __future__ import absolute_import, division, print_function, unicode_literals

# libreria pandas para leer nuestra base de datos
import pandas as pd

# permite construir y entrenar redes neuronales
import tensorflow as tf
from tensorflow.keras import datasets, layers, models

# Realiza gráficos
import matplotlib.pyplot as plt
import numpy as np
import os
from tqdm import tqdm #barra de progreso`
import cv2
from glob import glob

#para redimencionar
import sklearn
import skimage
from skimage.transform import resize

import random
#Usaremos datetime para nombrar archivos
import datetime

from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
# Transformacion imagen de color a uno en escala de grises
from skimage.color import rgb2gray

print(tf.__version__)
```

Fuente: Código implementado en jupyter notebook

Posterior a la instalación de las librerías se procede a realizar la ubicación de las imágenes, las cuales son cargadas y almacenadas en dos variables llamadas train_dir y test_dir, luego se recorre cada radiografía realizando un proceso de etiquetado, asignando con el número 0 (Normal) y número 1 (neumonía). Como la cantidad de iteraciones es considerable se utiliza un pequeño módulo llamado **tqdm** que permite visualizar una barra de progreso que se despliega en la pantalla. La imagen es transformada a escala de grises, además se modifica sus dimensiones quedando de (150x150x3).

Figura 9.
Ubicación del directorio de imágenes

```
# ubicación directorio de nuestro dataset
train_dir = "C:/Users/Lore/Desktop/TRABAJO FINAL ESPECIALIZACION/radiografias/train/"
test_dir = "C:/Users/Lore/Desktop/TRABAJO FINAL ESPECIALIZACION/radiografias/test/"

LOAD_FROM_IMAGES = False

def get_data(folder):
    X = [] # Carpetas
    y = [] # Target
       # recorrer las carpetas
    for folderName in os.listdir(folder):
        # Si el nombre de la carpeta es normal entonces :
        if not folderName.startswith('.'):
            if folderName in ['NORMAL']:
                label = 0
                 # si el normbre de la carpeta es neumonia entonces
            elif folderName in ['NEUMONIA']:
                label = 1
            else:
                label = 2     # si no es ninguno de los casos entonces

                 # recorrer imagen por imagen tqdm(for muy largos permite ver una barra de progreso)
            for image_filename in tqdm(os.listdir(folder + folderName)):
                img_file = cv2.imread(folder + folderName + '/' + image_filename)
                if img_file is not None:
                    # Transformar la imagen-se reescala la imagen y se cambia a escala de grises solo se trabaja con una capa
                    img_file = skimage.transform.resize(img_file, (150, 150, 3),mode='constant',anti_aliasing=True)
                    img_file = rgb2gray(img_file)
                    #img_file = scipy.misc.imresize(arr=img_file, size=(150, 150, 3))
                    # se transforma la imagen en un array
                    img_arr = np.asarray(img_file)
                    # agrega la imagen a X
                    X.append(img_arr)
                    # agrega etiqueta a la imágen
                    y.append(label)
```

Fuente: Código implementado en jupyter notebook

Se cargan las imágenes y se graban los arrays para realizar el entrenamiento, esto con el fin de no volver hacer el procedimiento de cargar imágenes aleatorias cada vez que corramos el código. y convertida en un array, la cual es almacenada en una variable X y Y previamente etiquetada.

Figura 10.
Carga de imágenes al array

```
    X = np.asarray(X)
    y = np.asarray(y)
    return X,y


if LOAD_FROM_IMAGES:
    #cargamos las imágenes a los arrays
    Xtrain, ytrain = get_data(train_dir)
    Xtest, ytest= get_data(test_dir)

    #grabamos los arrays en archivos
    np.save('xtrain.npy', Xtrain)
    np.save('ytrain.npy', ytrain)
    np.save('xtest.npy', Xtest)
    np.save('ytest.npy', ytest)
else:
    #cargamos los arrays anteriormente grabados
    Xtrain = np.load('xtrain.npy')
    ytrain = np.load('ytrain.npy')
    Xtest = np.load('xtest.npy')
    ytest = np.load('ytest.npy')
```

Fuente: Código implementado en jupyter notebook

Se realiza un histograma de una imagen puntual, en este caso de la radiografía 4 de la carpeta Xtrain para identificar las intensidades de forma gráfica, siendo el eje X la intensidad de pixeles y el eje Y el número de pixeles.

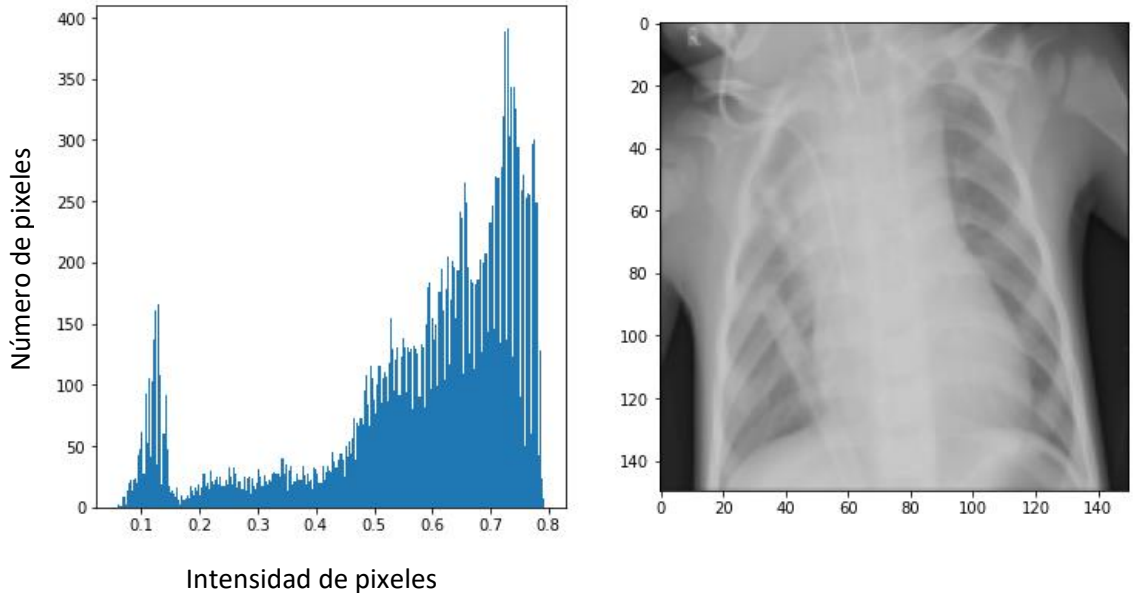
Figura 11.
Histograma

```
def plotHistogram(a):
    plt.figure(figsize=(12,6))
    plt.subplot(1, 2, 1)
    plt.hist(a.ravel(), bins=255)
    plt.subplot(1, 2, 2)
    plt.imshow(a, cmap='gray', vmin=0, vmax=1)
    plt.show()

plotHistogram(Xtrain[4])
```

Fuente: Código implementado en jupyter notebook

Figura 12
Histograma Distribución de pixeles Radiografía 4 de la carpeta de entrenamiento



Nota: Imagen e Histograma extraído del código implementado

Posteriormente utilizamos la librería seaborn para visualizar una gráfica de barras con la cantidad de imágenes por categoría. En la gráfica 12 podemos observar que está balanceada la cantidad de radiografías con neumonía (1) y radiografías de pacientes sanos (0). El eje X nos indica la etiqueta que asignamos a cada categoría y en el eje Y la cantidad de radiografías.

Figura 13.
Gráfica de barras con librería Seaborn

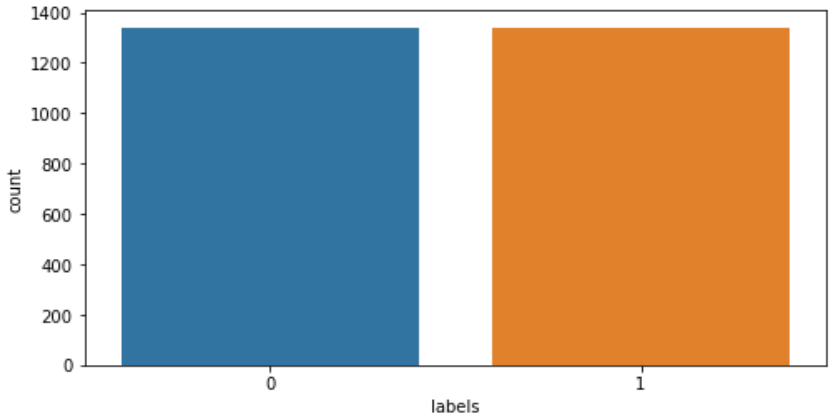
```
# Se importa la libreria seaborn Para graficar
import seaborn as sns

plt.figure(figsize=(8,4))
map_characters = {0: 'No Neumonía', 1: 'Si Neumonía'}
dict_characters=map_characters

df = pd.DataFrame()
df["labels"]=ytrain
lab = df['labels']
dist = lab.value_counts()
sns.countplot(lab)
print(dict_characters)
```

Fuente: Código implementado en jupyter notebook

Figura 14.
Gráfica de barras de la cantidad de radiografías por categoría



Nota: Gráfica de barras extraída del código implementado

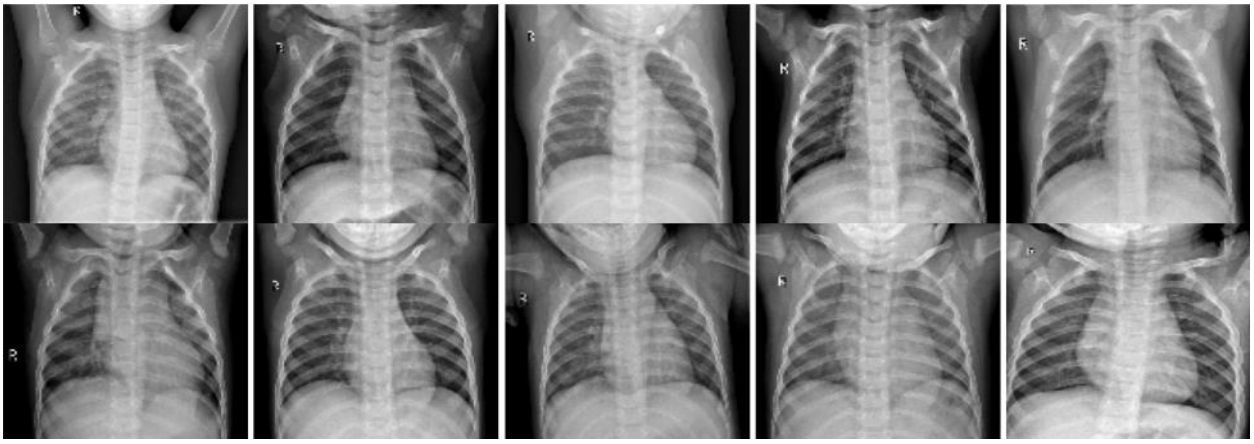
Cargamos múltiples imágenes aleatorias, extraídas de la carpeta de entrenamiento de pacientes sanos (No neumonía) y pacientes con posible patología de neumonía (si neumonía) con el fin de poder visualizar algunas imágenes e identificar los dos tipos de radiografía.

Figura 15
Identificación de radiografías sin neumonía

```
print("No Neumonía")
multipleImages = glob('radiografias/train/NORMAL/**')
i_ = 0
plt.rcParams['figure.figsize'] = (20.0, 20.0)
plt.subplots_adjust(wspace=0, hspace=0)
for l in multipleImages[:25]:
    im = cv2.imread(l)
    im = cv2.resize(im, (128, 128))
    plt.subplot(5, 5, i_+1)
    plt.imshow(cv2.cvtColor(im, cv2.COLOR_BGR2RGB)); plt.axis('off')
    i_ += 1
```

Fuente: Código implementado en jupyter notebook

Figura 16.
Radiografías aleatorias de entrenamiento sin neumonía



Nota: Radiografías extraídas del código implementado

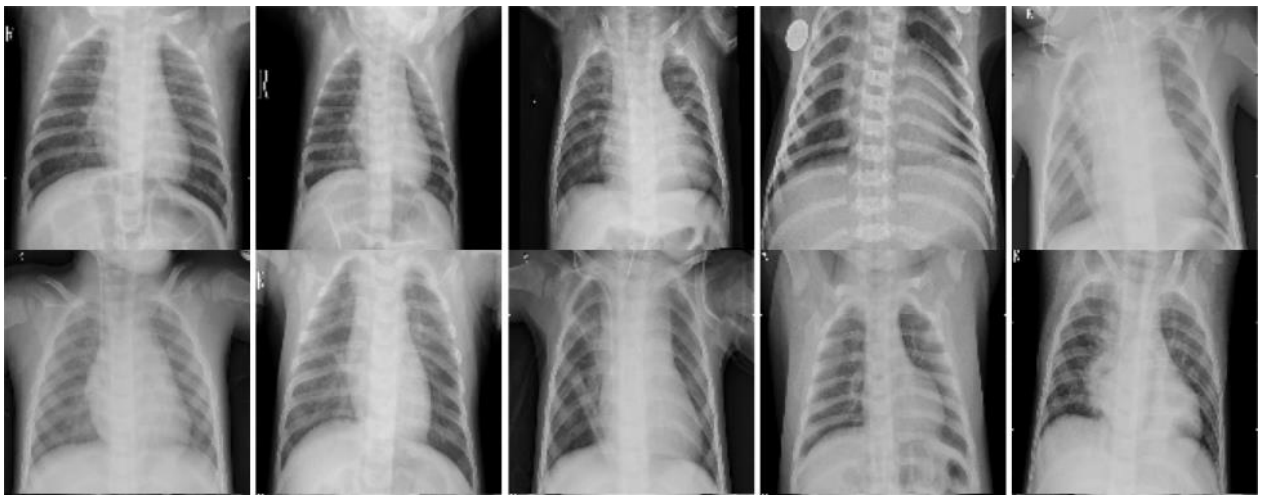
Figura 17
Identificación de radiografías con neumonía

```
# Se imprime imagenes de la carpeta de entrenamiento con Neumonia

print("Si neumonía")
multipleImages = glob('radiografias/train/NEUMONIA/**')
i_ = 0
plt.rcParams['figure.figsize'] = (20.0, 20.0)
plt.subplots_adjust(wspace=0, hspace=0)
for l in multipleImages[:25]:
    im = cv2.imread(l)
    im = cv2.resize(im, (128, 128))
    plt.subplot(5, 5, i_+1)
    plt.imshow(cv2.cvtColor(im, cv2.COLOR_BGR2RGB)); plt.axis('off')
    i_ += 1
```

Fuente: Código implementado en jupyter notebook

Figura 11.
Radiografías aleatorias de entrenamiento con neumonía



Nota: Radiografías extraídas del código implementado

Redimensionamos las radiografías para que queden de (150,150,1) y aplicamos el modelo de redes neuronales convolucionales con el siguiente código:

Figura 18.
Modelo de redes convolucionales (CNN)

```
model = models.Sequential()

model.add(layers.Conv2D(64, (3, 3), activation='relu', input_shape=(150, 150, 1)))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))


model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(2, activation='softmax'))

model.summary()
```

Fuente: Código implementado en jupyter notebook

En el código podemos observar que se lleva a cabo todos los pasos explicados anteriormente en la teoría, tres convoluciones con max pooling efectuando la función de activación ReLu y Softmax. Adicional se utiliza summary que nos arroja la siguiente información:

Figura 19.
Resumen del modelo

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 64)      640

 max_pooling2d (MaxPooling2D  (None, 74, 74, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 34, 34, 64)        36928

 max_pooling2d_2 (MaxPooling  (None, 17, 17, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 15, 15, 64)        36928

 flatten (Flatten)           (None, 14400)             0

 dense (Dense)               (None, 64)                921664

 dense_1 (Dense)             (None, 2)                 130

=================================================================
Total params: 1,033,218
Trainable params: 1,033,218
Non-trainable params: 0
_________________________________________________________________
```

Fuente: Código implementado en jupyter notebook

Vemos que a medida que se ejecuta cada capa se reduce a la mitad sus dimensiones. De la primera convolución a la segunda, aplicando max pooling pasamos de 148 pixeles a 72 pixeles, proceso que continua con las siguientes capas como se ve en la información arrojada por el summary. Para entrenar la red neuronal se utilizó el conjunto de datos agrupados en lotes, por lo que realizamos la validación con 20 épocas (Epoch) término que se usa en aprendizaje automático como un paso completo de datos a través de un algoritmo.

Para este procedimiento se realizó el siguiente código:

Figura 19.
Entrenamiento del modelo

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir, histogram_freq=1)

model.fit(XtrainReshaped,
          ytrain,
          epochs=20,
          validation_data = (XtestReshaped,ytest),
          callbacks=[tensorboard_callback])
```

Fuente: Código implementado en jupyter notebook

Resultados

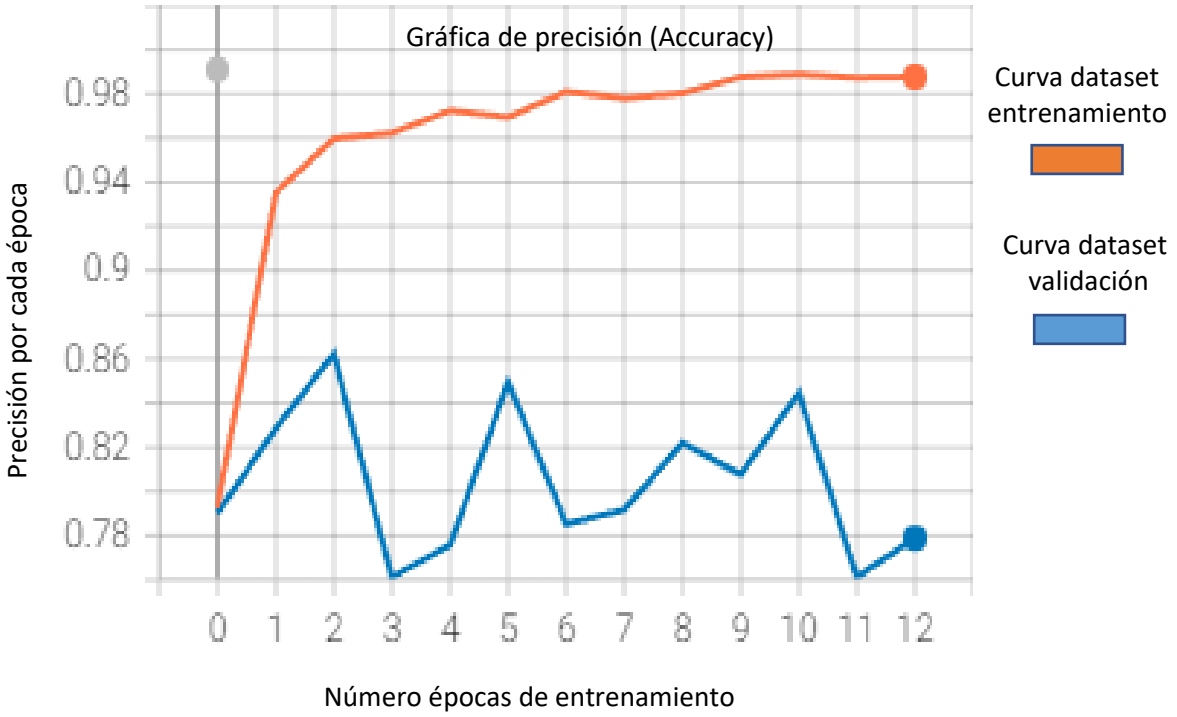
Se observa que al correr el código anterior nos muestra en pantalla la tabla de las veinte épocas que configuramos, arrojando el tiempo estimado y cuatro métricas por cada época (Loss, Accuracy, Val_loss, Val_accuracy)

Figura 20.
Epocas

```
Epoch 1/20
84/84 [==============================] - 59s 688ms/step - loss: 0.4815 - accuracy: 0.7453 - val_loss: 0.5485 - val_accuracy: 0.
7740
Epoch 2/20
84/84 [==============================] - 53s 627ms/step - loss: 0.1822 - accuracy: 0.9273 - val_loss: 0.6373 - val_accuracy: 0.
7724
Epoch 3/20
84/84 [==============================] - 52s 619ms/step - loss: 0.1329 - accuracy: 0.9508 - val_loss: 0.4337 - val_accuracy: 0.
8478
Epoch 4/20
84/84 [==============================] - 52s 625ms/step - loss: 0.1085 - accuracy: 0.9560 - val_loss: 0.9573 - val_accuracy: 0.
7548
Epoch 5/20
84/84 [==============================] - 51s 611ms/step - loss: 0.0781 - accuracy: 0.9705 - val_loss: 0.4451 - val_accuracy: 0.
8558
Epoch 6/20
84/84 [==============================] - 53s 627ms/step - loss: 0.0761 - accuracy: 0.9717 - val_loss: 1.1964 - val_accuracy: 0.
7484
Epoch 7/20
84/84 [==============================] - 52s 621ms/step - loss: 0.0713 - accuracy: 0.9746 - val_loss: 0.5864 - val_accuracy: 0.
8429
Epoch 8/20
84/84 [==============================] - 55s 656ms/step - loss: 0.0480 - accuracy: 0.9810 - val_loss: 0.6081 - val_accuracy: 0.
8413
Epoch 9/20
84/84 [==============================] - 54s 639ms/step - loss: 0.0390 - accuracy: 0.9866 - val_loss: 0.8635 - val_accuracy: 0.
8109
Epoch 10/20
84/84 [==============================] - 55s 653ms/step - loss: 0.0341 - accuracy: 0.9888 - val_loss: 1.1720 - val_accuracy: 0.
7788
```

Fuente: Código implementado en jupyter notebook

Figura 21.
Métricas de precisión

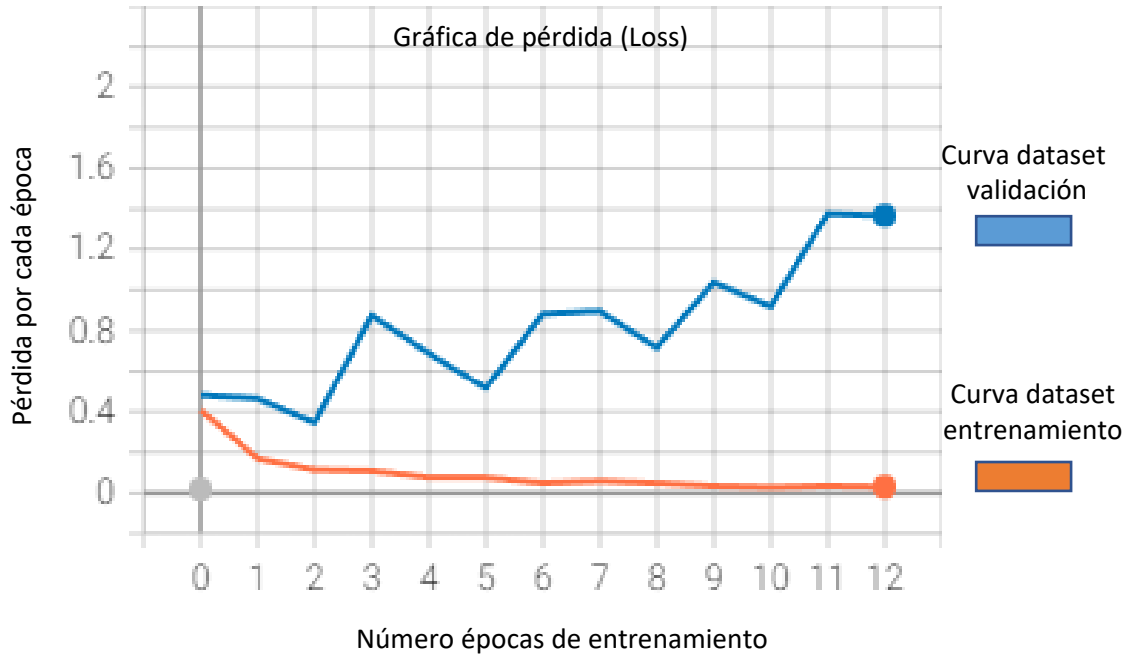


Nota: Grafica tomada del código implementado, visualización en Tensorboard

Se puede evaluar los resultados analizando las métricas de precisión del entrenamiento y la pérdida a medida que avanza por cada época, siendo el eje X el número de épocas o entrenamientos y el eje Y la precisión correspondiente a cada época.

En la figura 21 observamos que el punto más alto de precisión sucede en la época 2 con un 86,22 % de probabilidad de identificar una radiografía con neumonía la época 5, 8 y 10 se encuentran por encima del 80 % de probabilidad.

Figura 22.
Métrica de pérdida



Nota: Grafica tomada del código implementado, visualización en Tensorboard

Como observamos en la figura 22 el set de prueba en la época 2 alcanza en un 34 % su error más bajo, a partir de la época 3 el set de entrenamiento mejora generando un enorme sobreajuste donde el modelo es incapaz de generalizar.

Conclusiones

Se logró implementar y validar un modelo de clasificación de imágenes con posible diagnóstico de neumonía, modelo que al ser ejecutado obtuvo una precisión muy significativa, con más del 90 % de probabilidad en el set de entrenamiento y un 86,22% de precisión en el set de validación, cifra que no es muy confiable cuando se trata de temas relacionados con el área médica, por esta razón se debe trabajar más para mejorar el algoritmo. El desarrollo continuo de este tipo de algoritmo puede llegar a ser una herramienta potente como apoyo en el suministro de información relevante de patrones complejos en forma optimizada al equipo médico de imágenes, pero se debe investigar más al respecto no sólo en la parte técnica o algorítmica sino en la variabilidad de las imágenes de entrada por la perspectiva y experiencia de cada radiólogo.

También es indispensable recopilar mucha más cantidad de imágenes, factor que puede ser una limitante debido a cuestiones de protección y confidencialidad del paciente, además, estas imágenes deben tener altos estándares de calidad y es necesario realizar una buena preparación del paciente a la hora de realizar el estudio, así como el de configurar adecuadamente los protocolos y las técnicas de la adquisición.

Por último, es importante seguir avanzando en la investigación, experimentar con nuevos modelos, técnicas de procesamiento y participación del personal especializado en el tema, con el fin de mejorar el diagnóstico de millones de personas que padecen de esta patología.

Lista de referencias

Aguirre, F., Carballo, L., González, X., & Gigirey, V. (2021). INTELIGENCIA ARTIFICIAL APLICADA A LA ImAGEN mÉDICA. *Rev. Imagenol.*, *XXIV*(2), 48–57.

Alzate, J. P., Calderón, V., Palacios, J., & García, D. (2020). Mortalidad por neumonía y por todas las causas en niños menores de 5 años en Colombia entre el 2005 y 2016. *Infectio*, *25*(2), 108. https://doi.org/10.22354/in.v25i2.928

Andrés Martín, A., Moreno-Pérez, D., Alfayate Miguélez, S., Couceiro Gianzo, J. A., García García, M. L., Korta Murua, J., Martínez León, M. I., Muñoz Almagro, C., Obando Santaella, I., & Pérez Pérez, G. (2012). Etiología y diagnóstico de la neumonía adquirida en la comunidad y sus formas complicadas. *Anales de Pediatria*, *76*(3). https://doi.org/10.1016/j.anpedi.2011.09.011

Ávila-Tomás, J. F., Mayer-Pujadas, M. A., & Quesada-Varela, V. J. (2021). Artificial intelligence and its applications in medicine II: Current importance and practical applications. *Atencion Primaria*, *53*(1), 81–88. https://doi.org/10.1016/j.aprim.2020.04.014

Casal, R. F., Bouzas, J. C., & Oviedo De La Fuente, M. (2021). *Aprendizaje Estadístico*. *1*, 125–153.

Enzo Raschio, A., Cassandra Contreras, R., Felipe Allende, N., & Pablo Maturana, Q. (2021). Artificial intelligence: development of classification and segmentation algorithms in chest radiography. *Revista Chilena de Radiologia*, *27*(1), 8–16. https://doi.org/10.4067/S0717-93082021000100008

Gholamalinezhad, H., & Khosravi, H. (2020). *Pooling Methods in Deep Neural Networks, a Review*. http://arxiv.org/abs/2009.07485

Iván, C., Juan, P., Juan, R., Ferney, H., & Fabio, D. (2013). Implementación de una red neuronal artificial tipo SOM en una FPGA para la resolución de trayectorias tipo laberinto. *2013 2nd International Congress of Engineering Mechatronics and Automation, CIIMA 2013 - Conference Proceedings*. https://doi.org/10.1109/CIIMA.2013.6682790

Labaki, W. W., & Han, M. L. K. (2020). Chronic respiratory diseases: a global view. *The Lancet Respiratory Medicine*, *8*(6), 531–533. https://doi.org/10.1016/S2213-2600(20)30157-0

Martínez, S. M. R., Vidotti, S. A. B. G., & Jorente, M. J. V. (2016). Representación conceptual de imágenes médicas digitales: Integración de contexto y contenido visual. *Revista General de Informacion y Documentacion*, *26*(2), 651–672. https://doi.org/10.5209/rgid.54719

ministerio procteccion social colombia. (2014). *Guia De Manejo De Bronquiolitis Y Neumonia* (Issue 42).

Olabe, B. 2001. (n.d.). *REDES NEURONALES ARTIFICIALES- Xabier Bosogain Olabe.pdf*. http://cvb.ehu.es/open_course_ware/castellano/tecnicas/redes_neuro/contenidos/pdf/libro-del-curso.pdf

Rahman, T., Chowdhury, M. E. H., & Khandakar, A. (2020). applied sciences Transfer Learning with Deep Convolutional Neural Network (CNN) for Pneumonia Detection Using. *MDPI*,

J. App Sci., *3233*, 1–17.
Raudales Díaz, I. (2014). Imágenes diagnósticas: conceptos y generalidades. *Revista Facultad de Ciencias Médicas*, 35–43.
Rees, C. A., Basnet, S., Gentile, A., Gessner, B. D., Kartasasmita, C. B., Lucero, M., Martinez, L., O'Grady, K.-A. F., Ruvinsky, R. O., Turner, C., Campbell, H., Nair, H., Falconer, J., Williams, L. J., Horne, M., Strand, T., Nisar, Y. B., Qazi, S. A., & Neuman, M. I. (2020). An analysis of clinical predictive values for radiographic pneumonia in children. *BMJ Global Health*, *5*(8), e002708. https://doi.org/10.1136/bmjgh-2020-002708
Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. https://doi.org/10.1037/h0042519
Zar, H. J., & Ferkol, T. W. (2014). The global burden of respiratory disease - Impact on child health. *Pediatric Pulmonology*, *49*(5), 430–434. https://doi.org/10.1002/ppul.23030

Anexos

Código implementado en Lenguaje de programación Python

```
# Manejar conjunto de datos desequilibrado
pip install imblearn

# Cargamos la libreria
# garantiza que sentencias futuras se ejecuten en versiones anteriores a
la 2.1
from __future__ import absolute_import, division, print_function,
unicode_literals

# libreria pandas para leer nuestra base de datos
import pandas as pd

# permite construir y entrenar redes neuronales
import tensorflow as tf
from tensorflow.keras import datasets, layers, models

# Realiza gráficos
import matplotlib.pyplot as plt
import numpy as np
import os
from tqdm import tqdm #barra de progreso`
```

```
import cv2
from glob import glob

#para redimensionar
import sklearn
import skimage
from skimage.transform import resize

import random
#Usaremos datetime para nombrar archivos
import datetime

from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
# Transformacion imagen de color a uno en escala de grises
from skimage.color import rgb2gray

print(tf.__version__)

# ubicación directorio de nuestro dataset
train_dir = "C:/Users/Lore/Desktop/TRABAJO FINAL
ESPECIALIZACION/radiografias/train/"
test_dir = "C:/Users/Lore/Desktop/TRABAJO FINAL
ESPECIALIZACION/radiografias/test/"

LOAD_FROM_IMAGES = False

def get_data(folder):
    X = [] # Carpetas
    y = [] # Target
       # recorrer las carpetas
    for folderName in os.listdir(folder):
        # Si el nombre de la carpeta es normal entonces :
        if not folderName.startswith('.'):
            if folderName in ['NORMAL']:
                label = 0
                 # si el normbre de la carpeta es neumonia entonces
            elif folderName in ['NEUMONIA']:
                label = 1
            else:
                label = 2     # si no es ninguno de los casos entonces

                 # recorrer imagen por imagen tqdm(for muy largos permite
ver una barra de progreso)
            for image_filename in tqdm(os.listdir(folder + folderName)):
                img_file = cv2.imread(folder + folderName + '/' +
image_filename)
```

```
69                 if img_file is not None:
70                     # Transformar la imagen-se reescala la imagen y se
71 cambia a escala de grises solo se trabaja con una capa
72                     img_file = skimage.transform.resize(img_file, (150,
73 150, 3),mode='constant',anti_aliasing=True)
74                     img_file = rgb2gray(img_file)
75                     #img_file = scipy.misc.imresize(arr=img_file,
76 size=(150, 150, 3))
77                     # se transforma la imagen en un array
78                     img_arr = np.asarray(img_file)
79                     # agrega la imagen a X
80                     X.append(img_arr)
81                     # agrega etiqueta a la imágen
82                     y.append(label)
83     X = np.asarray(X)
84     y = np.asarray(y)
85     return X,y
86 
87 
88 if LOAD_FROM_IMAGES:
89     #cargamos las imágenes a los arrays
90     Xtrain, ytrain = get_data(train_dir)
91     Xtest, ytest= get_data(test_dir)
92 
93     #grabamos los arrays en archivos
94     np.save('xtrain.npy', Xtrain)
95     np.save('ytrain.npy', ytrain)
96     np.save('xtest.npy', Xtest)
97     np.save('ytest.npy', ytest)
98 else:
99     #cargamos los arrays anteriormente grabados
100     Xtrain = np.load('xtrain.npy')
101     ytrain = np.load('ytrain.npy')
102     Xtest = np.load('xtest.npy')
103     ytest = np.load('ytest.npy')
104 
105 def plotHistogram(a):
106     plt.figure(figsize=(12,6))
107     plt.subplot(1, 2, 1)
108     plt.hist(a.ravel(), bins=255)
109     plt.subplot(1, 2, 2)
110     plt.imshow(a, cmap='gray', vmin=0, vmax=1)
111     plt.show()
112 
113 plotHistogram(Xtrain[4])
114 
115 
```

```
# Cargamos 3 imagenes aleatorias de la carpeta entrenamiento/imagenes
normales

multipleImages = glob('radiografias/train/NORMAL/**')
def plotThreeImages(images):
    r = random.sample(images, 3)
    plt.figure(figsize=(20,20))
    plt.subplot(131)
    plt.imshow(cv2.imread(r[0]))
    plt.subplot(132)
    plt.imshow(cv2.imread(r[1]))
    plt.subplot(133)
    plt.imshow(cv2.imread(r[2]));
plotThreeImages(multipleImages)

print("No Neumonía")
multipleImages = glob('radiografias/train/NORMAL/**')
i_ = 0
plt.rcParams['figure.figsize'] = (20.0, 20.0)
plt.subplots_adjust(wspace=0, hspace=0)
for l in multipleImages[:25]:
    im = cv2.imread(l)
    im = cv2.resize(im, (128, 128))
    plt.subplot(5, 5, i_+1)
    plt.imshow(cv2.cvtColor(im, cv2.COLOR_BGR2RGB)); plt.axis('off')
    i_ += 1

# Se imprime imagenes de la carpeta de entrenamiento con Neumonia

print("Si neumonía")
multipleImages = glob('radiografias/train/NEUMONIA/**')
i_ = 0
plt.rcParams['figure.figsize'] = (20.0, 20.0)
plt.subplots_adjust(wspace=0, hspace=0)
for l in multipleImages[:25]:
    im = cv2.imread(l)
    im = cv2.resize(im, (128, 128))
    plt.subplot(5, 5, i_+1)
    plt.imshow(cv2.cvtColor(im, cv2.COLOR_BGR2RGB)); plt.axis('off')
    i_ += 1

# Se importa la libreria seaborn Para graficar
import seaborn as sns

plt.figure(figsize=(8,4))
map_characters = {0: 'No Neumonía', 1: 'Si Neumonía'}
dict_characters=map_characters
```

```
163
164 df = pd.DataFrame()
165 df["labels"]=ytrain
166 lab = df['labels']
167 dist = lab.value_counts()
168 sns.countplot(lab)
169 print(dict_characters)
170
171 Xtrain.shape
172
173 XtrainReshaped = Xtrain.reshape(len(Xtrain),150,150,1)
174 XtestReshaped = Xtest.reshape(len(Xtest),150,150,1)
175
176 # Aplicamos el modelo de redes neuronales convolucionales
177 model = models.Sequential()
178
179 model.add(layers.Conv2D(64, (3, 3), activation='relu', input_shape=(150,
180 150, 1)))
181 model.add(layers.MaxPooling2D((2, 2)))
182
183 model.add(layers.Conv2D(64, (3, 3), activation='relu'))
184 model.add(layers.MaxPooling2D((2, 2)))
185
186 model.add(layers.Conv2D(64, (3, 3), activation='relu'))
187 model.add(layers.MaxPooling2D((2, 2)))
188
189 model.add(layers.Conv2D(64, (3, 3), activation='relu'))
190
191
192 model.add(layers.Flatten())
193 model.add(layers.Dense(64, activation='relu'))
194 model.add(layers.Dense(2, activation='softmax'))
195
196 model.summary()
197
198 model.compile(optimizer='adam',
199               loss='sparse_categorical_crossentropy',
200               metrics=['accuracy'])
201
202 log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
    tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir,
    histogram_freq=1)

    model.fit(XtrainReshaped,
              ytrain,
              epochs=2,
              validation_data = (XtestReshaped,ytest),
```

```
          callbacks=[tensorboard_callback])

test_loss, test_acc = model.evaluate(XtestReshaped, ytest)
print(test_acc)

test_loss, test_acc = model.evaluate(XtrainReshaped, ytrain)
```

Universidad®
Católica
de Manizales

VIGILADA MINEDUCACIÓN

Obra de Iglesia
de la Congregación

Hermanas de la Caridad
Dominicas de La Presentación
de la Santísima Virgen

Universidad Católica de Manizales
Carrera 23 # 60-63 Av. Santander / Manizales - Colombia
PBX (6)8 93 30 50 - www.ucm.edu.co