Aprendizaje automático aplicado al diagnóstico de la ocurrencia de la Leishmaniasis a través de imágenes de lesiones cutáneas

Camilo Castro Duarte

Nota de Aceptación

Certificamos que el presente Trabajo de Grado Satisface, en alcances y calidad, todos los requisitos que demanda un Trabajo de Grado de Maestría.

DIEGO LUIS LINARES
Director

VALENTINA CORCHUELO GUZMÁN
Jurado

HERNÁN DARIO VARGAS CARDONA
Jurado

Aprobado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana Cali, para optar el título de Magister en Ciencia de Datos.

HERNÁN CAMILO ROCHA NIÑO Ph. D.
Decano Facultad de Ingeniería y Ciencias

JUAN CARLOS MARTÍNEZ ARIAS
Director Posgrados de Ingeniería y Ciencias

Cali – 13/02/2024

Acta de Correcciones al Documento de Trabajo de Grado

Santiago de Cali, 13/02/2024

Autor: Camilo Castro Duarte

Título del Trabajo de Grado: "Aprendizaje automático aplicado al diagnóstico de la ocurrencia de la Leishmaniasis a través de imágenes de lesiones cutáneas"

Director: Diego Luis Linares

Como indica el artículo 2.13 de las Directrices para Trabajo de Grado de Maestría, he verificado que el estudiante indicado arriba ha implementado todas las correcciones que los Jurados del Proyecto de Trabajo de Grado definieron que se efectuaran, como consta en el Acta de Evaluación correspondiente.

Firma del Director del Trabajo de Grado

Santiago de Cali, 13 de febrero de 2024

Ingeniero:
Juan Carlos Martínez Arias
Director Posgrados de Ingeniería
Facultad de Ingeniería y Ciencias
Pontificia Universidad Javeriana - Cali

Con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el Trabajo de Grado y posteriormente optar por el título de Magíster en Ciencia de Datos, nos permitimos presentar a su consideración el proyecto de Trabajo de Grado denominado "Aprendizaje automático aplicado al diagnóstico de la ocurrencia de la Leishmaniasis a través de imágenes de lesiones cutáneas", el cual será realizado por el (la) estudiante Camilo Castro Duarte con código 8972785 perteneciente al énfasis en N/A, bajo la dirección del profesor Diego Luis Linares.

El suscrito director del Trabajo de Grado autoriza para que se proceda a hacer la evaluación de este Proyecto ante el Tribunal que para el efecto se designe, toda vez que ha revisado cuidadosamente el documento y avala que ya se encuentra listo para ser presentado oficialmente.

Atentamente,

Firma
Camilo Castro Duarte

C.C. 1.101.683.867 de Socorro

Firma
Diego Luis Linares

C.C. 16.739.144 de Cali

Maestría en Ciencia de Datos
Facultad de Ingeniería y Ciencias

FICHA RESUMEN
TRABAJO DE GRADO DE MAESTRÍA

TITULO: "Aprendizaje automático aplicado al diagnóstico de la ocurrencia de la Leishmaniasis a través de imágenes de lesiones cutáneas"

1. ÉNFASIS: N/A
2. TIPO DE PROYECTO: Aplicado
3. ÁREA DE TRABAJO: Aprendizaje automático
4. ESTUDIANTE (S): Camilo Castro Duarte
5. CORREO ELECTRÓNICO: castrin65659@javerianacali.edu.co
6. DIRECCIÓN Y TELÉFONO: Cra 14#11-11 / 3174660182
7. DIRECTOR: Diego Luis Linares
8. VINCULACIÓN DEL DIRECTOR (en la universidad): Planta
9. CORREO ELECTRÓNICO DEL DIRECTOR: dlinares@javerianacali.edu.co
10. CO-DIRECTOR(ES) (Si aplica): María Adelaida Gómez
11. GRUPO O EMPRESA QUE LO AVALA (Si aplica): Destino
12. OTROS GRUPOS O EMPRESAS: CIDEIM
13. PALABRAS CLAVE (al menos 5): Leishmaniasis, Machine Learning, predicción, lesiones cutáneas, imágenes.
14. ODS QUE APLICA EL PROYECTO (Agenda 2030): 3. Salud y bienestar
15. FECHA DE INICIO (Desarrollo del proyecto): 1/01/2023
16. RESUMEN (máximo 400 palabras).

Este proyecto buscó aplicar técnicas de aprendizaje automático específicamente Redes Neuronales Convolucionales, para predecir si una lesión de tipo cutánea corresponde o no a la enfermedad de la Leishmaniasis, evaluación que no resulta tan simple mediante observación debido a la similitud con otros tipos de lesiones, para ello, se entrenaron modelos predictivos mediante una base de datos con 885 imágenes de lesiones cutáneas (407 correspondientes a Leishmaniasis y 478 correspondientes a otras lesiones cutáneas), las imágenes de Leishmaniasis fueron suministradas por el CIDEIM (Centro Internacional de Entrenamiento e Investigaciones Médicas). Durante el desarrollo del proyecto se trabajaron diferentes etapas como es el caso de la limpieza y adecuación de la base de datos de imágenes, selección y entrenamiento de los modelos, contraste de resultados y la identificación del modelo más adecuado. Los resultados arrojados por los diferentes experimentos y sus respectivas métricas permitieron establecer cuál es el modelo más adecuado para hacer la predicción, de esta forma es posible hacer un diagnóstico previo de la lesión del paciente sin siquiera estar este de cuerpo presente, también permite ayudar a personal de la salud que no tiene tanta experiencia en este tipo de lesiones a despejar dudas y tomar medidas. De esta forma se le da paso a diferentes posibilidades de aplicación del algoritmo desarrollado dentro del campo médico, como es el caso de integraciones futuras con aplicativos móviles o desarrollos web, esto puede facilitar el análisis de las lesiones de los pacientes de manera remota y de una forma ágil.

Santiago de Cali, 13 de febrero del 2024

Doctora
Gloría Inés Alvarez V.
Directora Maestría en Ciencia de Datos
Facultad de Ingeniería y Ciencias
Pontificia Universidad Javeriana de Cali

Asunto: Presentación para evaluación del proyecto aplicado

Cordial Saludo,

Con el fin de cumplir con los requisitos exigidos por la Universidad para optar por el título de Magíster en Ciencia de Datos, nos permitimos presentar a su consideración el proyecto denominado "Aprendizaje automático aplicado al diagnóstico de la ocurrencia de la Leishmaniasis a través de imágenes de lesiones cutáneas", el cual fue realizado por el estudiante Camilo Castro Duarte con código 8972785 perteneciente a la Maestría en Ciencia de Datos, bajo la dirección de Diego Luis Linares.

El suscrito director del Proyecto Aplicado autoriza para que se proceda a hacer la evaluación de este proyecto, toda vez que ha revisado cuidadosamente el documento y avala que ya se encuentra listo para ser presentado y sustentado oficialmente.

Atentamente,

Camilo Castro Duarte

C.C. 1.101.683.867 de Socorro

Diego Luis Linares

C.C. 16.739.144 de Cali

Documentación anexa:
Resumen del Proyecto Aplicado en formato digital (máximo 1 página).
Una copia digital (PDF) del documento del proyecto aplicado

FICHA RESUMEN
PROYECTO APLICADO – MAESTRÍA EN CIENCIA DE DATOS

TÍTULO:

1. ÁREA DE TRABAJO: Aprendizaje automático
2. TIPO DE PROYECTO (Aplicado, Innovación, Investigación): Aplicado
3. ESTUDIANTE(S): Camilo Castro Duarte
4. CORREO ELECTRÓNICO: castrin65659@javerianacali.edu.co
5. DIRECCIÓN Y TELÉFONO: Cra 14#11-11 / 3174660182
6. DIRECTOR: Diego Luis Linares
7. VINCULACIÓN DEL DIRECTOR: Tiempo completo
8. CORREO ELECTRÓNICO DEL DIRECTOR: dlinares@javerianacali.edu.co
9. CO-DIRECTOR (Si aplica): María Adelaida Gómez
10. GRUPO O EMPRESA QUE LO AVALA (Si aplica): Destino
11. OTROS GRUPOS O EMPRESAS: CIDEIM
12. PALABRAS CLAVE (al menos 5): Leishmaniasis, Machine Learning, predicción, lesiones cutáneas, imágenes.
13. FECHA DE INICIO: 01/01/2023
14. FECHA DE FINALIZACIÓN: 10/11/2023
15. RESUMEN:

Este proyecto buscó aplicar técnicas de aprendizaje automático específicamente Redes Neuronales Convolucionales, para predecir si una lesión de tipo cutánea corresponde o no a la enfermedad de la Leishmaniasis, evaluación que no resulta tan simple mediante observación debido a la similitud con otros tipos de lesiones, para ello, se entrenaron modelos predictivos mediante una base de datos con 885 imágenes de lesiones cutáneas (407 correspondientes a Leishmaniasis y 478 correspondientes a otras lesiones cutáneas), las imágenes de Leishmaniasis fueron suministradas por el CIDEIM (Centro Internacional de Entrenamiento e Investigaciones Médicas). Durante el desarrollo del proyecto se trabajaron diferentes etapas como es el caso de la limpieza y adecuación de la base de datos de imágenes, selección y entrenamiento de los modelos, contraste de resultados y la identificación del modelo más adecuado. Los resultados arrojados por los diferentes experimentos y sus respectivas métricas permitieron establecer cuál es el modelo más adecuado para hacer la predicción, de esta forma es posible hacer un diagnóstico previo de la lesión del paciente sin siquiera estar este de cuerpo presente, también permite ayudar a personal de la salud que no tiene tanta experiencia en este tipo de lesiones a despejar dudas y tomar medidas. De esta forma se le da paso a diferentes posibilidades de aplicación del algoritmo desarrollado dentro del campo médico, como es el caso de integraciones futuras con aplicativos móviles o desarrollos web, esto puede facilitar el análisis de las lesiones de los pacientes de manera remota y de una forma ágil.

APRENDIZAJE AUTOMÁTICO APLICADO AL DIAGNÓSTICO DE LA OCURRENCIA DE LA LEISHMANIASIS A TRAVÉS DE IMÁGENES DE LESIONES CUTÁNEAS

Camilo Castro Duarte

Proyecto Aplicado para optar al título de
Magister en Ciencia de Datos

Director
Diego Luis Linares

Codirectora
María Adelaida Gómez

FACULTAD DE INGENIERÍA Y CIENCIAS
MAESTRÍA EN CIENCIA DE DATOS
SANTIAGO DE CALI, DICIEMBRE 9 DE 2023

TABLA DE CONTENIDO

	Pág.
INTRODUCCIÓN	11
1. DEFINICIÓN DEL PROBLEMA	12
1.1 PLANTEAMIENTO DEL PROBLEMA	12
1.2 FORMULACIÓN DEL PROBLEMA	13
2. OBJETIVOS DEL PROYECTO	14
2.1 OBJETIVO GENERAL	14
2.2 OBJETIVOS ESPECÍFICOS	14
3. MARCO TEÓRICO Y ANTECEDENTES	15
3.1 MARCO TEÓRICO	15
3.1.1 LEISHMANIASIS EN COLOMBIA	15
3.1.2 INTELIGENCIA ARTIFICIAL Y APRENDIZAJE PROFUNDO	16
3.1.2.1 REDES NEURONALES CONVOLUCIONALES	17
3.1.2.2 FUNCIÓN DE ACTIVACIÓN	19
3.1.2.2.1 FUNCIÓN DE ACTIVACIÓN SIGMOID	19
3.1.2.2.2 FUNCIÓN DE ACTIVACIÓN SOFTMAX	19
3.1.2.2.3 FUNCIÓN DE ACTIVACIÓN RELU	20
3.1.2.3 DROPOUT	20
3.1.2.4 OPTIMIZADORES	21
3.1.2.4.1 OPTIMIZADOR ADADELTA	21
3.1.2.4.2 OPTIMIZADOR ADAM	22
3.1.3 MODELOS PRE ENTRENADOS	22
3.1.3.1 VGG-16	22
3.1.3.2 VGG-19	23

3.1.3.3 RESNET-152 24

3.1.4 METRICAS PARA EVALUAR EL DESEMPEÑO DE LOS MODELOS 25

3.1.4.1 MATRIZ DE CONFUSIÓN 25

3.1.4.2 EXACTITUD 25

3.1.4.3 SENSIBILIDAD 26

3.1.4.4 PRESISIÓN 26

3.1.4.5 PUNTAJE F1 26

3.2 ANTECEDENTES / TRABAJOS RELACIONADOS 27

4. PREPARACIÓN DE LOS DATOS 29

4.1 CONSECUCIÓN Y CLASIFICACIÓN DE LAS IMÁGENES 29

4.2 LIMPIEZA DEL DATASET 31

4.3 DISTRIBUCIÓN DEL DATASET PARA ENTRENAMIENTO,
VALIDACIÓN Y TESTEO 32

4.4 AUMENTO DE DATOS 33

5. CONSTRUCCIÓN DE LOS MODELOS Y OPTIMIZACIÓN 35

5.1 SELECCIÓN DE LAS TÉCNICAS DE APRENDIZAJE AUTOMÁTICO ADECUADAS 35

5.2 AJUSTE DE HIPERPARÁMETROS 37

5.3 ENTRENAMIENTO DEL MODELO TOMANDO TRAIN+VALIDATION 39

5.4 TRASNFER LEARNING VGG-16 40

5.5 TRASNFER LEARNING VGG-19 41

5.6 TRASNFER LEARNING RESNET-152 41

5.7 UTILIZACIÓN DE FINE TUNING 42

5.7.1 FINE TUNING A VGG-16 43

5.7.2 FINE TUNING A VGG-19 43

6. ANÁLISIS DE RESULTADOS 45

7. DESPLIEGUE DEL MODELO 47

8. CONCLUSIONES Y TRABAJOS FUTUROS 48

8.1. CONCLUSIONES 48

8.2. TRABAJOS FUTUROS 49

9. REFERENCIAS BIBLIOGRÁFICAS (Estilo IEEE) 50

LISTA DE FIGURAS

Pág.

Figura 1. Red neuronal 17

Figura 2: Red neuronal convolucional 17

Figura 3. Aplicación de un proceso de convolución a una imagen y el correspondiente mapa de características 18

Figura 4. Función de activación Sigmoid 19

Figura 5. Función de activación Relu 19

Figura 6. Red con todas las neuronas encendidas (izquierda) y red después de aplicar Dropout (derecha) 21

Figura 7. Arquitectura VGG-16 23

Figura 8. Arquitectura VGG-19 23

Figura 9. Arquitectura ResNet-152 24

Figura 10. Imagen de Leishmaniasis sin editar 31

Figura 11. Imagen de Leishmaniasis editada 32

Figura 12. Muestra del set de datos de Leishmaniasis para Train 33

Figura 13. Diagrama de bloques de la arquitectura seleccionada 36

Figura 14. Visualización por capas de la arquitectura seleccionada 36

Figura 15. Gráfica de la función de pérdida (superior) y del rendimiento (inferior) 38

Figura 16. Interfaz gráfica 47

LISTA DE TABLAS

Pág.

Tabla 1. Matriz de confusión 25

Tabla 2. Etiquetado de importancia para el proyecto 29

Tabla 3. Lesiones cutáneas causadas por infecciones 30

Tabla 4. Lesiones cutáneas causadas por inflamaciones reactivas 30

Tabla 5. Lesiones cutáneas causadas por tumores 30

Tabla 6. Distribución de los datos para entrenamiento, validación y testeo 32

Tabla 7. Mejores hiperparámetros encontrados 37

Tabla 8. Parámetros de la red 38

Tabla 9. Distribución de los datos para entrenamiento y testeo 39

Tabla 10. Matriz de confusión del modelo entrenado con los mejores hiperparámetros 39

Tabla 11. Métricas del modelo entrenado con los mejores hiperparámetros 40

Tabla 12. Matriz de confusión aplicando Transfer Learning VGG-16 40

Tabla 13. Métricas aplicando Transfer Learning VGG-16 40

Tabla 14. Matriz de confusión aplicando Transfer Learning VGG-19 41

Tabla 15. Métricas aplicando Transfer Learning VGG-19 41

Tabla 16. Matriz de confusión aplicando Transfer Learning ResNet-152 42

Tabla 17. Métricas aplicando Transfer Learning ResNet-152 42

Tabla 18. Matriz de confusión aplicando Fine Tuning VGG-16 43

Tabla 19. Métricas aplicando Fine Tuning VGG-16 43

Tabla 20. Matriz de confusión aplicando Fine Tuning VGG-19 44

Tabla 21. Métricas aplicando Fine Tuning VGG-19 44

Tabla 22. Métricas del modelo entrenado con los mejores hiperparámetros 45

Tabla 23. Métricas aplicando Transfer Learning VGG-19 46

Tabla 24. Métricas aplicando Transfer Learning ResNet-152 46

Tabla 25. Métricas aplicando Fine Tuning VGG-16 46

LISTA DE ANEXOS

	Pág.
Anexo 1. Script implementado	52
Anexo 2. Script interfaz gráfica	63

INTRODUCCIÓN

Las lesiones cutáneas producto de la enfermedad de la Leishmaniasis pueden asemejarse a las producidas por otras enfermedades como es el caso de lepra, cáncer de piel, tuberculosis, micosis cutáneas, así como picaduras de insectos infectados, por lo que mediante observación no resulta tan fácil identificar qué tipo de enfermedad es. Poder predecir con antelación y con una buena precisión a qué se está enfrentando el paciente es de vital importancia para aplicar un tratamiento adecuado que repercuta de manera positiva en la salud de este.

En el presente proyecto se aplicaron Redes Neuronales Convolucionales y se entrenaron modelos con una base de datos de imágenes de lesiones cutáneas de la Leishmaniasis y de otros tipos de lesiones cutáneas, las imágenes de lesiones correspondientes a Leishmaniasis fueron aportada por el CIDEIM (Centro Internacional de Entrenamiento e Investigaciones Médicas), los resultados de dichos modelos fueron contrastados mediante evaluación de métricas para poder establecer cual es más preciso y adecuado para entregar la mejor solución, de este modo mediante una fotografía se pudo predecir si la lesión cutánea de determinado paciente corresponde o no a Leishmaniasis.

1. DEFINICIÓN DEL PROBLEMA

1.1. PLANTEAMIENTO DEL PROBLEMA

La Leishmaniasis es una enfermedad parasitaria transmitida por un mosquito, que se presenta principalmente en regiones tropicales, las manifestaciones clínicas de la enfermedad van desde úlceras cutáneas que cicatrizan espontáneamente, hasta formas fatales en las cuales se presenta inflamación grave del hígado y del bazo. Después de la picadura del mosquito, transcurren entre una y doce semanas para que se desarrolle una pápula eritematosa, que crece y se ulcera, generando una costra de exudado seco. La mayoría de los pacientes desarrollan una o dos de estas lesiones, en cara, manos o piernas, con un tamaño de cada lesión entre 0,5 y 3 centímetros de diámetro, aunque existe una gran variabilidad en la presentación de ellas. Este tipo de Leishmaniasis es más común en América Central y del Sur [1].

Colombia es un país que por su ubicación cuenta con unas condiciones geográficas y climáticas bastante adecuado para la proliferación del mosquito transmisor de la Leishmaniasis, las regiones con mayor afectación son aquellas en cercanías al océano pacifico y que además cuentan con abundancia selvática como es el caso del departamento del Chocó, Cauca, Valle del Cauca y Nariño [1].

En la actualidad existen diferentes tipos de enfermedades que también pueden llegar a generar lesiones cutáneas similares a las producidas por la Leishmaniasis, como es el caso de lepra, cáncer de piel, tuberculosis, micosis cutáneas, así como picaduras de insectos infectados, por lo que a través de observación no resulta tan fácil saber de qué enfermedad se trata y se puede prestar para confusiones, teniendo en cuenta lo anterior, el presente proyecto está enfocado en la aplicación de técnicas de Aprendizaje automático (Redes Neuronales Convolucionales) sobre una base de datos conformada por imágenes de lesiones cutáneas de Leishmaniasis (imágenes aportadas por el CIDEIM) y de otros tipos de lesiones cutáneas similares a la Leishmaniasis (imágenes extraídas de internet mediante “Google imágenes” para conformar la respectiva base de datos a través de diferentes fuentes), por medio de las cuales se pudo entrenar varios modelos con los que es posible realizar predicciones para determinar si las úlceras que presenta un paciente determinado corresponden o no a Leishmaniasis y de este modo saber qué tipo de medicación aplicar.

1.2. FORMULACIÓN DEL PROBLEMA

La problemática planteada implicó responder los siguientes interrogantes:

¿Cómo diagnosticar si una lesión cutánea de algún paciente corresponde o no a Leishmaniasis mediante técnicas de aprendizaje automático aplicadas al reconocimiento de imágenes?
¿Cómo procesar las imágenes suministradas por el CIDEIM?
¿Cómo aplicar las técnicas de aprendizaje automático adecuadas?
¿Cómo refinar los modelos de aprendizaje automático seleccionados?
¿Cómo evaluar los resultados de los modelos de aprendizaje automático aplicados?

2. OBJETIVOS DEL PROYECTO

2.1 OBJETIVO GENERAL

Aplicar técnicas de aprendizaje automático para diagnosticar la ocurrencia de la Leishmaniasis a través de imágenes de lesiones cutáneas

2.2 OBJETIVOS ESPECÍFICOS

- Preprocesar las imágenes correspondientes a la base de datos suministrada por el CIDEIM
- Seleccionar las técnicas de aprendizaje automático adecuadas para el procesamiento de imágenes
- Refinar los modelos de aprendizaje automático seleccionados para diagnosticar la ocurrencia de la enfermedad
- Evaluar los resultados obtenidos por los modelos de aprendizaje automático aplicados

3. MARCO TEÓRICO Y ANTECEDENTES

3.1. MARCO TEÓRICO

A continuación, se presentan los temas que están relacionados con el desarrollo del presente proyecto, inicialmente se detalla la enfermedad y las principales características de su afectación cutánea, posteriormente se presentan conceptos fundamentales sobre aprendizaje profundo, redes neuronales convolucionales y modelos pre entrenados.

3.1.1 LEISHMANIASIS EN COLOMBIA

Las diversas formas clínicas de la Leishmaniasis constituyen un serio problema de salud pública en el mundo. La Organización Mundial de la Salud (OMS), en la matriz de énfasis estratégico del programa de investigación en enfermedades tropicales (Tropical Disease Research), ha clasificado la Leishmaniasis en la categoría I como una enfermedad emergente y sin control. En América Latina, incluida Colombia, se presentan casos de las tres formas clínicas principales de la enfermedad. En Colombia el aumento de los casos notificados en los últimos cinco años refleja un incremento de la transmisión de la enfermedad, que se puede atribuir a diversos factores, entre otros, al aumento de las actividades humanas en ambientes silvestres en donde existe la transmisión enzoótica, la adaptación de los vectores al peridomicilio y domicilio, y la urbanización de la parasitosis en la forma cutánea y visceral [1].

La Leishmaniasis humana presenta diversas manifestaciones clínicas, y se describen la Leishmaniasis cutánea (LC), la Leishmaniasis mucosa o mucocutánea (LM) y la Leishmaniasis visceral (LV). La LV es causada por Leishmania donovani en el viejo mundo y por Leishmania infantum tanto en el viejo como en el nuevo mundo; es la más severa de las formas, en la que la ausencia de tratamiento puede causar la muerte de las personas infectadas, especialmente a los menores de cinco años. Las formas agudas de LV tienen inicio súbito de escalofríos y fiebre alta que a veces se hace periódica; al transcurrir el tiempo, se presenta hepatomegalia y/o esplecnomegalia, anemia y pérdida de peso, en especial en infantes con problemas de desnutrición. La esplenomegalia puede llegar hasta la fosa iliaca izquierda, aunque el bazo se palpa blando y poco doloroso. También hay manifestaciones hemorrágicas. La muerte sobreviene cuando se presentan infecciones como neumonía bacteriana o septicemia, tuberculosis o disentería. El periodo de incubación puede variar desde los 10 días a un año, siendo la media de 2 a 9 meses [1].
Existe un número considerable de especies de Leishmania que causan Leishmaniasis cutánea (LC) o Leishmaniasis mucosa (LM), las cuales no son fatales, pero causan una elevada morbilidad en la población localizada en las áreas endémicas. Las formas clínicas varían desde lesiones cerradas como pápulas, nódulos y placas que pueden ser de aspecto verrugoso, hasta las formas ulceradas.

En Colombia, la presentación más frecuente es la úlcera indolora con compromiso linfangítico y adenopatía regional. Las lesiones se localizan por lo general en cara, brazos y piernas; se inician con la presencia de pequeñas pápulas rojas, que se convierten gradualmente en pequeños nódulos firmes que se ulceran gradualmente. Algunos pacientes jamás se ulceran y presentan lesiones nodulares. Cuando se presentan úlceras, éstas son usualmente redondeadas, con un fondo limpio de aspecto granular y bordes hiperémicos y engrosados, que se tornan indoloros. Si no hay infección bacteriana ni micótica asociada, el paciente no presenta eritema ni descamación de la piel [1].

3.1.2 INTELIGENCIA ARTIFICIAL Y APRENDIZAJE PROFUNDO

Podríamos definir la Inteligencia Artificial (IA) como aquellas técnicas y sistemas informáticos capaces de realizar tareas que normalmente requieren de inteligencia humana, como la percepción visual, la toma de decisiones o el reconocimiento del habla [2]. Al comienzo de su desarrollo, estaba basada en técnicas que no usaban el aprendizaje, sino que seguían un conjunto de reglas establecidas para una determinada tarea, como por ejemplo jugar al ajedrez. Sin embargo, este enfoque se quedaba corto para tareas más complejas y surgió el subcampo que hoy conocemos como aprendizaje automático (o Machine Learning), en el que un sistema es capaz de determinar por sí solo las reglas necesarias para llevar a cabo un problema sin estar específicamente programado para esa tarea. Dentro del Aprendizaje Computacional podemos distinguir entre aprendizaje supervisado, no supervisado y aprendizaje por refuerzo. En el supervisado se le suministran al sistema los datos y las salidas deseadas a dichos datos, como por ejemplo las imágenes y la categoría a la que pertenecen; en el no supervisado el sistema extrae por sí solos las categorías existentes en el problema a partir de las diferencias que encuentra entre los datos; y en el aprendizaje por refuerzo esto es aprendido con base a la retroalimentación que recibe del entorno. Además, dentro del Aprendizaje Computacional se distingue otro subcampo, conocido como Aprendizaje Profundo (o Deep Learning), una serie de algoritmos basados en redes neuronales artificiales, las cuales son capaces de extraer patrones de datos y aprender a realizar tareas a partir de ellos [2].

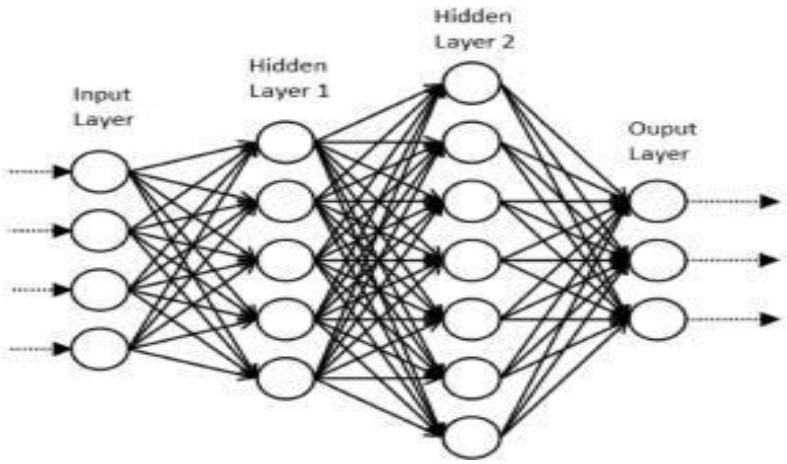

Figura 1. Red neuronal [2]

Una red neuronal artificial es un modelo computacional que se asemeja a las sinapsis neuronales reales y que consiste en un conjunto de neuronas conectadas entre sí. Estas neuronas están organizadas en capas, en donde cada neurona recibe información de neuronas de capas anteriores, la procesa y la propaga a neuronas de la siguiente capa. Cada neurona asigna un peso a cada una de sus entradas, que son las salidas que le llegan de la capa de neuronas anterior [2].

3.1.2.1 REDES NEURONALES CONVOLUCIONALES

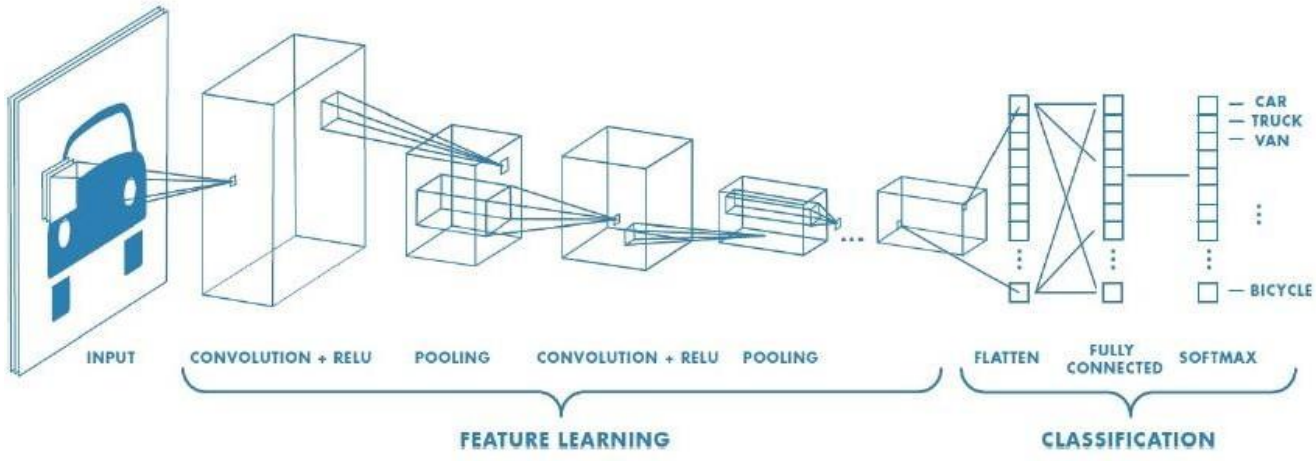

Figura 2: Red neuronal convolucional [3]

Las redes neuronales convolucionales (CNN, del inglés Convolutional Neural Networks), son un tipo de red que se utiliza mayoritariamente para tareas de análisis de imagen y de señal. La razón principal por la que se usan este tipo de redes en dichas tareas es que permiten observar patrones

locales en una imagen, de forma que si lo que se busca es un objeto la red lo detectará independientemente de en qué parte de la imagen se encuentre. En contraste, si se usara una red densa, que son las explicadas anteriormente, solo se podría encontrar patrones generales en la imagen, es decir, cuando el objeto que estamos buscando estuviese en un sitio determinado de la imagen.

Las capas convolucionales son las capas principales que extraen los patrones de interés. Cada capa consta de uno o varios filtros de imagen, de manera que cada uno de los valores matriciales de estos filtros son los pesos a entrenar. De esta forma, cuando estos filtros hacen una convolución a través de toda la imagen, la neurona encargada de analizar un determinado píxel no solo analiza dicho píxel, sino que también recibe información de los píxeles de alrededor, que es lo que facilita encontrar los patrones locales comentados anteriormente. Lo que hace cada uno de estos filtros es resaltar una característica determinada de la imagen, de forma que cada capa cada vez buscará unas características más complejas basadas en las que ha inspeccionado la capa anterior.

Por otro lado, el pooling es un proceso que se aplica entre capas de convolución y que sirve para reducir la resolución de la imagen ayudando así a la red neuronal convolucional con la detección.

La detección de objetos complejos o incluso enfermedades desde una imagen surge de la secuencia de capas usando convolución (combinadas con las de pooling). Si bien en una red neuronal convolucional para detectar coches las primeras capas de convolución no son capaces de detectar ningún coche, sí que son capaces de detectar líneas y curvas. Y unas capas más profundas igual son capaces de detectar una rueda hasta llegar a las últimas capas que son capaces de detectar coches como tal. Es por este suceso que los diagramas de las redes neuronales convolucionales se representan como un embudo [3].

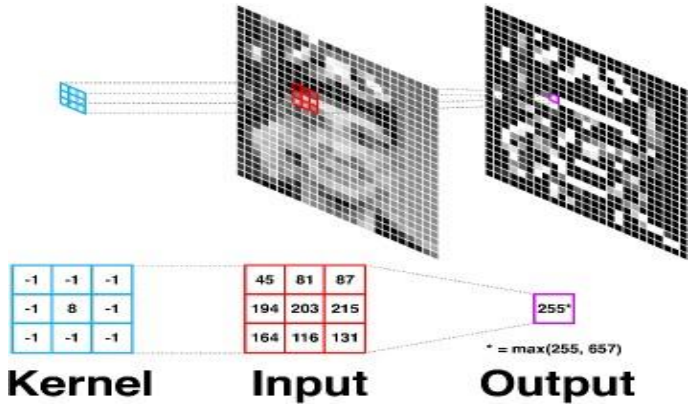

Figura 3. Aplicación de un proceso de convolución a una imagen y el correspondiente mapa de características [3]

3.1.2.2 FUNCIÓN DE ACTIVACIÓN

Una red neuronal está formada de neuronas conectadas entre ellas; a su vez, cada conexión de nuestra red neuronal está asociada a un peso que dictamina la importancia que tendrá esa relación en la neurona al multiplicarse por el valor de entrada.
Cada neurona tiene una función de activación que define la salida de la neurona. La función de activación se usa para introducir la no linealidad en las capacidades de modelado de la red.

3.1.2.2.1 FUNCIÓN DE ACTIVACIÓN SIGMOID

La función sigmoid permite reducir valores extremos o atípicos en datos válidos sin eliminarlos. Una función sigmoidea convierte variables independientes de rango casi infinito en probabilidades simples entre 0 y 1. La mayor parte de su salida estará muy cerca de los extremos de 0 o 1 (resulta importante que los valores estén en este rango en algunos tipos de redes neuronales).

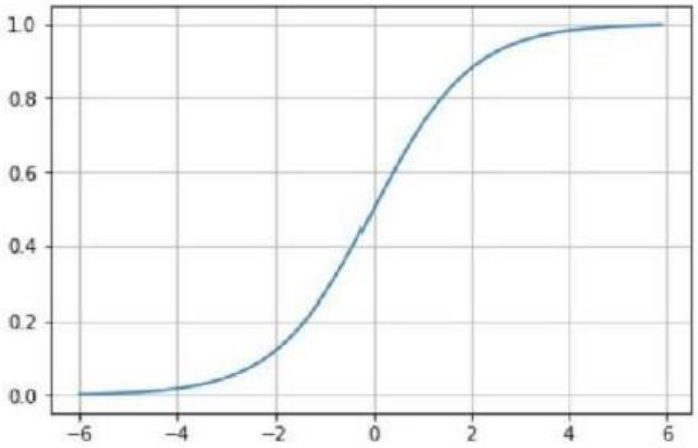

Figura 4. Función de activación Sigmoid [4]

3.1.2.2.2 FUNCIÓN DE ACTIVACIÓN SOFTMAX

La función softmax se basa en calcular “las evidencias” de que una determinada imagen pertenece a una clase en particular y luego se convierten estas evidencias en probabilidades de que pertenezca a cada una de las posibles clases.

Para medir la evidencia de que una determinada imagen pertenece a una clase en particular, una aproximación consiste en realizar una suma ponderada de la evidencia de pertenencia de cada uno de sus píxeles a esa clase.

La función Softmax en lugar de clasificar en binario puede contener múltiples límites de decisión y se encontrará a menudo en la capa de salida de un clasificador.

3.1.2.2.3 FUNCIÓN DE ACTIVACIÓN RELU

La función de activación rectified linear unit (ReLU) es una transformación muy interesante que activa un solo nodo si la entrada está por encima de cierto umbral. El comportamiento por defecto y más habitual es que mientras la entrada tenga un valor por debajo de cero, la salida será cero, pero cuando la entrada se eleva por encima, la salida es una relación lineal con la variable de entrada de la forma $f(x) = x$. La función de activación ReLU ha demostrado funcionar en muchas situaciones diferentes, y actualmente es muy usada [4].

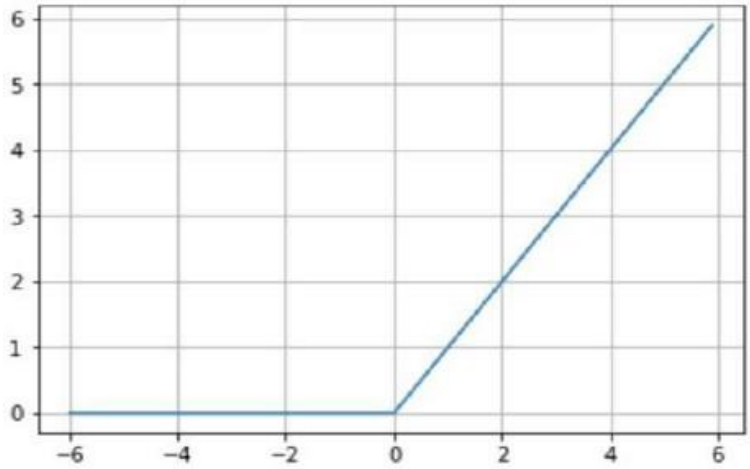

Figura 5. Función de activación Relu [4]

3.1.2.3 DROPOUT

La capa de Dropout (apagado de neuronas) ayuda a prevenir el sobreajuste ya que, durante el entrenamiento, desactiva algunas neuronas en la proporción que se le asigne. Así, se evita que todas las neuronas de una capa se fijen en las mismas características de la capa anterior, de modo que la red se vería forzada a no depender solo de unas pocas características [2].

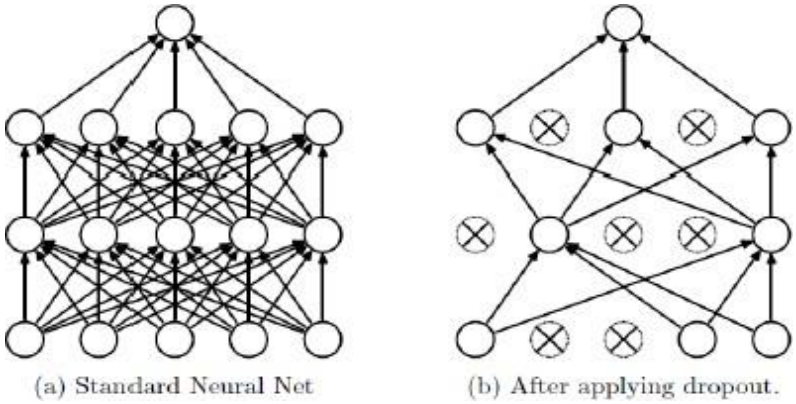

Figura 6. Red con todas las neuronas encendidas (izquierda) y red después de aplicar Dropout (derecha) [2]

3.1.2.4 OPTIMIZADORES

El algoritmo de optimización es el que le permite a la red neuronal calcular los pesos de los parámetros a partir de los datos de entrada y de la función de loss definida.
Keras dispone en estos momentos de diferentes optimizadores que pueden usarse: SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam.

De forma general, podemos ver el proceso de aprendizaje como un problema de optimización global donde los parámetros (los pesos y los sesgos) se deben ajustar de tal manera que la función de loss se minimice. En la mayoría de los casos, estos parámetros no se pueden resolver analíticamente, pero en general se pueden aproximar bien con algoritmos de optimización iterativos u optimizadores, como los mencionados anteriormente [4].

3.1.2.4.1 OPTIMIZADOR ADADELTA

Adadelta es un método de descenso de gradiente estocástico que se basa en la tasa de aprendizaje adaptativo por dimensión para abordar dos inconvenientes:

- El continuo descenso de los ritmos de aprendizaje a lo largo de la formación.
- La necesidad de una tasa de aprendizaje global seleccionada manualmente.

Adadelta es una extensión más sólida de Adagrad que adapta las tasas de aprendizaje en función de una ventana móvil de actualizaciones de gradientes, en lugar de acumular todos los gradientes anteriores. De esta manera, Adadelta continúa aprendiendo incluso cuando se han realizado muchas actualizaciones. En comparación con Adagrad, en la versión original de Adadelta no es necesario establecer una tasa de aprendizaje inicial. En esta versión, se puede establecer la tasa

de aprendizaje inicial, como en la mayoría de los otros optimizadores de Keras [6].

3.1.2.4.2 OPTIMIZADOR ADAM

Adam, o Adaptive Moment Estimation, es un algoritmo de optimización que combina las ventajas de los algoritmos RMSprop y Momentum para mejorar el proceso de aprendizaje de un modelo.

Al igual que Momentum, Adam utiliza una estimación del momento y de la magnitud de los gradientes anteriores para actualizar los parámetros del modelo en cada iteración. Sin embargo, en lugar de utilizar una tasa de aprendizaje constante para todos los parámetros, Adam adapta la tasa de aprendizaje de cada parámetro individualmente en función de su estimación del momento y de la magnitud del gradiente. Esto permite que el modelo se ajuste de manera más eficiente y efectiva a los datos de entrenamiento, lo que puede llevar a una mayor precisión de la predicción en comparación con otros métodos de optimización [7].

3.1.3 MODELOS PRE ENTRENADOS

En la búsqueda de optimizar los resultados obtenidos se planteó la utilización de Transfer Learning mediante la aplicación de modelos pre entrenados, Transfer Learning se refiere al conjunto de métodos que permiten transferir conocimientos adquiridos gracias a la resolución de problemas para resolver otros problemas, el Transfer Learning ha tenido un gran éxito con el crecimiento del Deep Learning. Frecuentemente, los modelos utilizados en este campo necesitan grandes tiempos de cálculo y muchos recursos. Sin embargo, utilizando como punto de partida modelos pre-entrenados, el Transfer Learning permite desarrollar rápidamente modelos eficaces y resolver problemas complejos de Computer Vision o Natural Language Processing.

3.1.3.1 VGG-16

Uno de los modelos más conocidos es VGG-16, el cual es una red neuronal convolucional con 16 capas de profundidad, de las cuales 13 son convolucionales y las demás fully connected, fue propuesta por K. Simonyan y A. Zisserman, de la Universidad de Oxford, y adquirió notoriedad al ganar el desafío de reconocimiento visual a gran escala de ImageNet en 2014, el modelo alcanzó una precisión de 92.7%, fue presentado por primera vez en el artículo de investigación titulado "Very Deep Convolutional Networks for Large-Scale Image Recognition" [13].

VGG-16 fue entrenada con más de un millón de imágenes de la base de datos ImageNet, dicha base de datos es usada en la actualidad para evaluar algoritmos de clasificación, localización y reconocimiento de imágenes, cuenta con más de 14 millones de imágenes etiquetadas en más de 1000 clases [13].

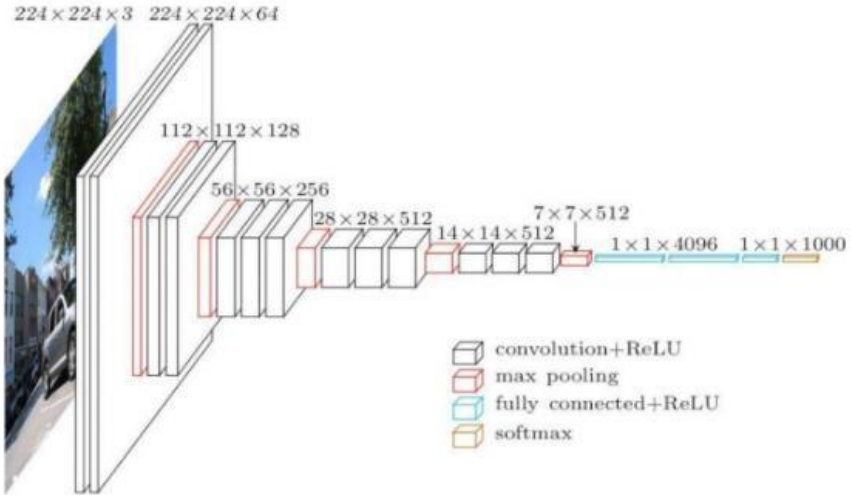

Figura 7. Arquitectura VGG-16 [8]

3.1.3.2 VGG-19

Por otra parte, está VGG-19, que es una red similar a VGG-16 pero en este caso cuenta con 19 capas de profundidad, de las cuales 16 son convolucionales y las demás fully connected, esta red también fue entrenada haciendo uso de la base de datos de imágenes de ImageNet, la red pre entrenada puede clasificar imágenes en 1000 categorías de objetos (por ejemplo, teclado, ratón, lápiz y muchos animales). Como resultado, la red ha aprendido representaciones ricas en características para una amplia gama de imágenes [13].

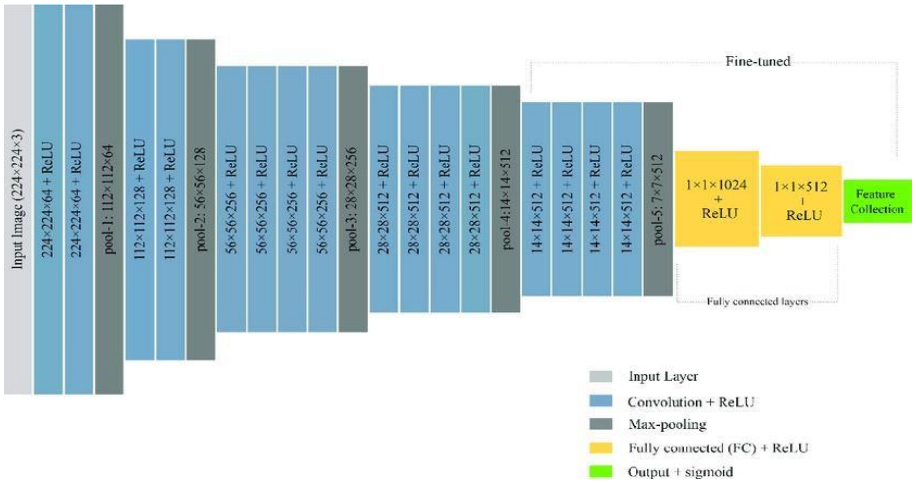

Figura 8. Arquitectura VGG-19 [9]

VGG-19 fue desarrollado por el grupo Visual Geometry Group (VGG) de la Universidad de Oxford. Fue presentado por primera vez en 2014 en el artículo de investigación titulado "Very Deep Convolutional Networks for Large-Scale Image Recognition" por Karen Simonyan y Andrew Zisserman.

3.1.3.3 RESNET-152

El tercer modelo que se planteó utilizar fue ResNet-152, la cual cuenta con 152 capa de profundidad, 150 capas convolucionales y dos fully connected, ResNet es una red residual que se inspira en el hecho biológico de que algunas neuronas se conectan con otras neuronas en capas no necesariamente contiguas, saltando capas intermedias, fue propuesta por Kaiming He, Xiangyu Zhang, Shaoqing Ren y Jian Sun, fue la red ganadora de la competencia ImageNet de 2015 logrando un error de 3.57%, además, permitió por primera vez entrenar redes muy profundas (de más de 100 capas), fue presentado en el artículo titulado "Deep Residual Learning for Image Recognition" y expuesto en la Conferencia sobre Computer Vision and Pattern Recognition (CVPR) en 2016 [14].

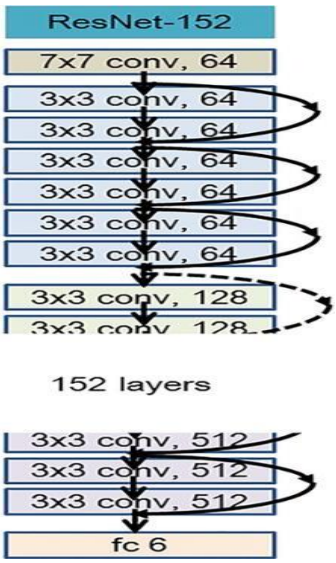

Figura 9. Arquitectura ResNet-152 [10]

Cabe resaltar que de los anteriores modelos pre entrenados solo se utilizó la parte convolucional, la cual se congeló para que no se entrenara en el nuevo proceso de entrenamiento, por otra parte, las demás capas de la red planteada inicialmente de nuestro modelo si se utilizaron para que fueran entrenadas, como es el caso de la capa densa y el clasificador.

3.1.4 METRICAS PARA EVALUAR EL DESEMPEÑO DE LOS MODELOS

3.1.4.1 MATRIZ DE CONFUSIÓN

En el mundo del aprendizaje automático una herramienta para evaluar modelos es la matriz de confusión (*confusion matrix*), una tabla con filas y columnas que contabilizan las predicciones en comparación con los valores reales. Se usa esta tabla para entender mejor que tan bien el modelo se comporta y es muy útil para mostrar de forma explícita cuando una clase es confundida con otra.
Una matriz de confusión para un clasificador binario tiene esta estructura:

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdadero positivo (VP)	Falso negativo (FN)
	Negativos	Falso positivo (FP)	Verdadero negativo (VN)

Tabla 1. Matriz de confusión

En la que:

VP es la cantidad de positivos que fueron clasificados correctamente como positivos por el modelo.

VN es la cantidad de negativos que fueron clasificados correctamente como negativos por el modelo.

FN es la cantidad de positivos que fueron clasificados incorrectamente como negativos.

FP es la cantidad de negativos que fueron clasificados incorrectamente como positivos [4].

3.1.4.2 EXACTITUD

Teniendo en cuenta la anterior matriz de confusión, la exactitud (*accuracy)* se puede calcular sumando los valores de la diagonal principal y dividiendo por el total de casos examinado:

$$Accuracy = (VP + VN) / (VP + FP + VN + FN)$$

Ahora bien, la exactitud puede ser engañosa en la calidad del modelo, esto porque no se distingue entre los errores de tipo falso positivo y falso negativo, como si ambos tuvieran la misma importancia. Por ejemplo, en un modelo que predice si una seta es venenosa. En este caso, el coste de un falso negativo, es decir, una seta venenosa dada por comestible podría ser dramático. En cambio, al revés, un falso positivo tiene un coste muy diferente [4].

3.1.4.3 SENSIBILIDAD

La sensibilidad (o *recall*) nos indica que tan bien el modelo evita los falsos negativos:

$$Recall = VP / (VP + FN)$$

Es decir, el total de observaciones positivas que fueron correctamente identificadas por el algoritmo, también se conoce como Tasa de Verdaderos Positivos (True Positive Rate) [4].

3.1.4.4 PRECISIÓN

Esta métrica representa el número de verdaderos positivos que son realmente positivos en comparación con el número total de valores positivos predichos.

$$Precision = VP / (VP + FP)$$

También se puede considerar como la dispersión del conjunto de valores obtenidos a partir de mediciones repetidas de una magnitud. Cuanto menor es la dispersión mayor la precisión [4].

3.1.4.5 PUNTAJE F1

Esta métrica es la combinación de las métricas de precisión y sensibilidad y sirve como una medida de calidad más general del modelo. La mejor puntuación F1 es igual a 1 y la peor a 0.

$$F1 = 2* (Precision * Recall) / (Precision + Recall)$$

3.2. ANTECEDENTES / TRABAJOS RELACIONADOS

Se realizó una búsqueda de información precisa, relacionada con la Leishmaniasis en Colombia y su afectación en la población, así como libros y artículos referentes a la aplicación de modelos de Machine Learning para el análisis de imágenes de úlceras cutáneas de diferentes enfermedades.

- Una mirada a la epidemiología y al control de la Leishmaniasis Zoonótica en Colombia. En este artículo se estudian las diversas formas de la Leishmaniasis, enfocándose específicamente en Colombia, en donde se evalúa el incremento de la transmisión de la enfermedad por diversos factores, además se puede encontrar abundancia en cuanto a terminología científica que se desprende de la enfermedad [1].

- Clasificación de imágenes de lesiones cutáneas mediante redes neuronales convolucionales y Aprendizaje profundo. Este trabajo de grado contiene información muy precisa con respecto a conceptos relacionados con Deep Learning y las redes neuronales convolucionales, aplicadas específicamente al reconocimiento de las lesiones cutáneas producidas por el melanoma o cáncer de piel [2].

- Deep Learning en la detección de lesiones cutáneas malignas. Este trabajo de grado es similar al descrito anteriormente puesto que está enfocado al reconocimiento de lesiones cutáneas producidas por el cáncer de piel y como mediante la aplicación de estas técnicas se pueden hacer diagnósticos precisos sin el concepto de un dermatólogo, además cuenta con una sección de anexos en donde se describe en detalle la programación del algoritmo implementado en Python, lo cual puede ser de utilidad para el presente proyecto [3].

- Atlas interactivo de Lesihamaniasis en las Américas: aspectos clínicos y diagnósticos diferenciales. En este libro hay información detallada sobre la Leishmaniasis y diversos tipos de lesiones cutáneas de características similares, principalmente aquellas de tipo ulcerado, el contenido es bastante explicativo acompañado de múltiples ilustraciones [5].

- Aplicación móvil para detección de melanoma con redes neuronales artificiales. En este proyecto implementan una red neuronal convolucional para clasificar imágenes de lesiones de melanoma y desarrollan una interfaz gráfica mediante un aplicativo móvil [11].

- Aprendizaje de máquina aplicado a la predicción del éxito del tratamiento de la Leishmaniasis. Este trabajo de grado contiene información de interés relacionada con la enfermedad de la Leishmaniasis y la aplicación de diferentes modelos de aprendizaje automático para predecir el éxito en el tratamiento de la enfermedad [12].

4. PREPARACIÓN DE LOS DATOS

4.1 CONSECUCIÓN Y CLASIFICACIÓN DE LAS IMÁGENES

El CIDEIM (Centro Internacional de Entrenamiento e Investigaciones Médicas) permitió la utilización de un conjunto de 1.386 imágenes distribuidas en 44 carpetas, estas carpetas corresponden a diferentes pacientes y documentan varias etapas de la enfermedad, antes, durante y después del tratamiento. Para el caso de este proyecto resultan de interés aquellas imágenes antes de la aplicación del tratamiento y muy al inicio de este, donde se puede percibir la lesión cutánea en todo su esplendor, para ello el CIDEIM facilitó un diccionario desarrollado por ellos mismos mediante el cual se pudieron identificar y clasificar estas imágenes en específico a través del etiquetado que tenían, el etiquetado de interés es el siguiente:

Acrónimo o texto	Significado	Observaciones
V00 o V0	Visita 0	Visita inicial, primer contacto con el paciente, por lo general de toma de muestra (frotis). No ha ingresado al estudio ni recibido tratamiento.
EdeB	Evaluación de base	Primera visita del estudio, el paciente no ha recibido tratamiento
S0 o S00	Semana 0	Antes del inicio del tratamiento
V1A	Visita 1A	Día 2-3 de tratamiento, ideal 48 horas después de su inicio

Tabla 2. Etiquetado de importancia para el proyecto

De un total de 1.386 imágenes se seleccionaron aquellas con los acrónimos anteriormente señalados, quedando un total de 407 imágenes para a la categoría de interés “Leishmaniasis”.

La segunda categoría que conforma el dataset se identificó como “Otras”, dicha categoría corresponde a otras lesiones cutáneas con características visuales similares a Leshmaniasis, las cuales pueden a simple vista generar confusión al momento de dar un diagnóstico inicial del paciente. Por recomendación del CIDEIM se escogieron aquellas lesiones de tipo ulcerado, tomando como base la siguiente clasificación:

	Enfermedades	Úlcera	Pápula	Placa	Nódulo	Verruga	Tumor	Linfangitis	Cicatriz
Infecciones	Úlcera piógena	X							
	Forúnculos		X		X				
	Sífilis	X		X					
	Microbacterias atípicas	X		X	X	X			
	Tuberculosis cutánea	X	X	X		X			
	Lepra		X						
	Cromomicosis			X	X	X			X
	Esporotricosis	X	X	X	X	X		X	
	Histoplasmosis	X	X	X	X				
	Lobomicosis		X	X	X				
	Paracoccidiodomicosis	X	X	X	X				
	Ectima	X		X					
	Impétigo			X					

Tabla 3. Lesiones cutáneas causadas por infecciones [5]

	Enfermedades	Úlcera	Pápula	Placa	Nódulo	Verruga	Tumor	Linfangitis	Cicatriz
Inflamatorias reactivas	Úlcera venosa	X							
	Úlcera arterial	X							
	Úlcera vascular mixta	X							
	Úlcera diabética	X							
	Lupus discoide		X	X		X			
	Psoriasis			X					
	Necrosis cutánea	X		X					
	Anemia de células falciformes	X							X
	Úlcera traumática	X							
	Picaduras de insecto	X	X	X					X
	Pioderma gangrenoso	X	X	X					X
	Sarcoidosis	X	X	X					
	Úlcera banal	X							
	Granuloma facial		X	X	X				

Tabla 4. Lesiones cutáneas causadas por inflamaciones reactivas [5]

	Enfermedades	Ulcera	Pápula	Placa	Nódulo	Verruga	Tumor	Linfangitis	Cicatriz
Tumorales	Carcinoma basocelular	X	X	X		X	X		X
	Carcinoma escamocelular	X	X	X		X	X		X
	Queratoacantoma	X	X				X		
	Linfoma cutáneo	X	X	X			X		
	Linfocitoma cutis		X	X					

Tabla 5. Lesiones cutáneas causadas por tumores [5]

Para la consecución de estas imágenes se utilizó “Google Imágenes” y se tuvo en cuenta las características específicas de cada lesión visualmente hablando, estas fueron descargadas en formatos tales como JPG, JPEG y PNG, los cuales pueden ser procesados sin problema mediante el lenguaje de programación Python. Para la categoría “Otras” se obtuvieron un total de 478 imágenes.

4.2 LIMPIEZA DEL DATASET

Tanto la categoría “Leshmaniasis” como “Otras”, son fotografías tomadas desde diferentes ángulos y perspectivas, en donde se pueden apreciar diferentes elementos que no son de interés para el análisis, como una mano, un brazo, el torso completo del paciente, las dos piernas, así como objetos que hayan quedado registrados en ese instante. Teniendo en cuenta que de las imágenes lo que resulta de interés son las lesiones propiamente, fue necesario hacer una edición de cada una de las imágenes buscando hacer énfasis en la úlcera cutánea.

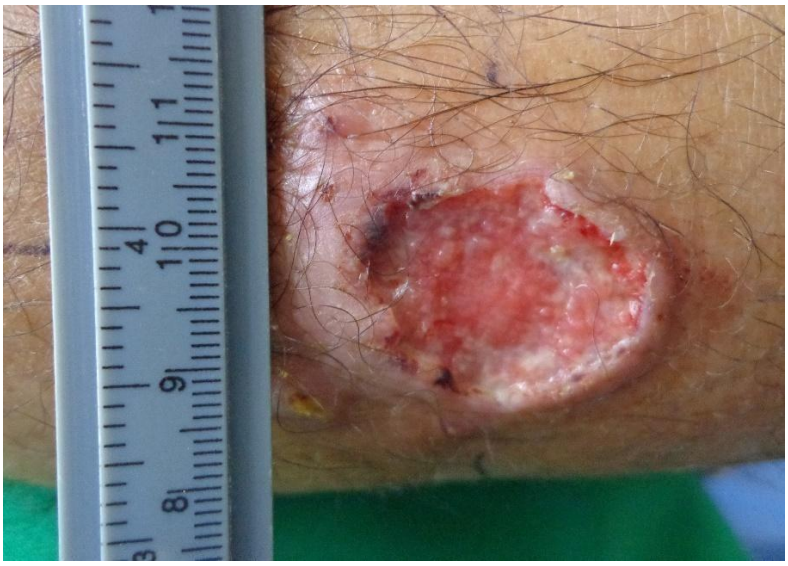

Figura 10. Imagen de Leishmaniasis sin editar

Figura 11. Imagen de Leishmaniasis editada

4.3 DISTRIBUCIÓN DEL DATASET PARA ENTRENAMIENTO, VALIDACIÓN Y TESTEO

Previo a la distribución se decidió no hacer balanceo de datos puesto que la diferencia entre la cantidad de imágenes de Leishmaniasis (407) y Otras (478) no es tan marcada.

La proporción que se seleccionó para este proyecto corresponde a 60% para entrenamiento, 20% para validación y 20% para testeo en las dos categorías planteadas (normalmente el conjunto de datos se suele repartir en esta proporción, lo importante es ser siempre conscientes de que hay que evitar el sobreajuste u "overfitting"), teniendo en cuenta que los datos de entrenamiento son los datos que se usaron para entrenar el modelo y los datos de validación son los datos que se usaron para comprobar si el modelo que se generó a partir de los datos de entrenamiento funciona. Es decir, si las respuestas predichas por el modelo para un caso totalmente nuevo son acertadas o no.

A continuación, se presenta la distribución de las imágenes del Dataset:

Categoría	Total datos 100%	Train 60%	Validación 20%	Testeo 20%
Leishmaniasis	407	245	81	81
Otras	478	286	96	96
	885	**531**	**177**	**177**

Tabla 6. Distribución de los datos para entrenamiento, validación y testeo

A continuación, un ejemplo del conjunto de datos de la clase Leishmaniasis para Train:

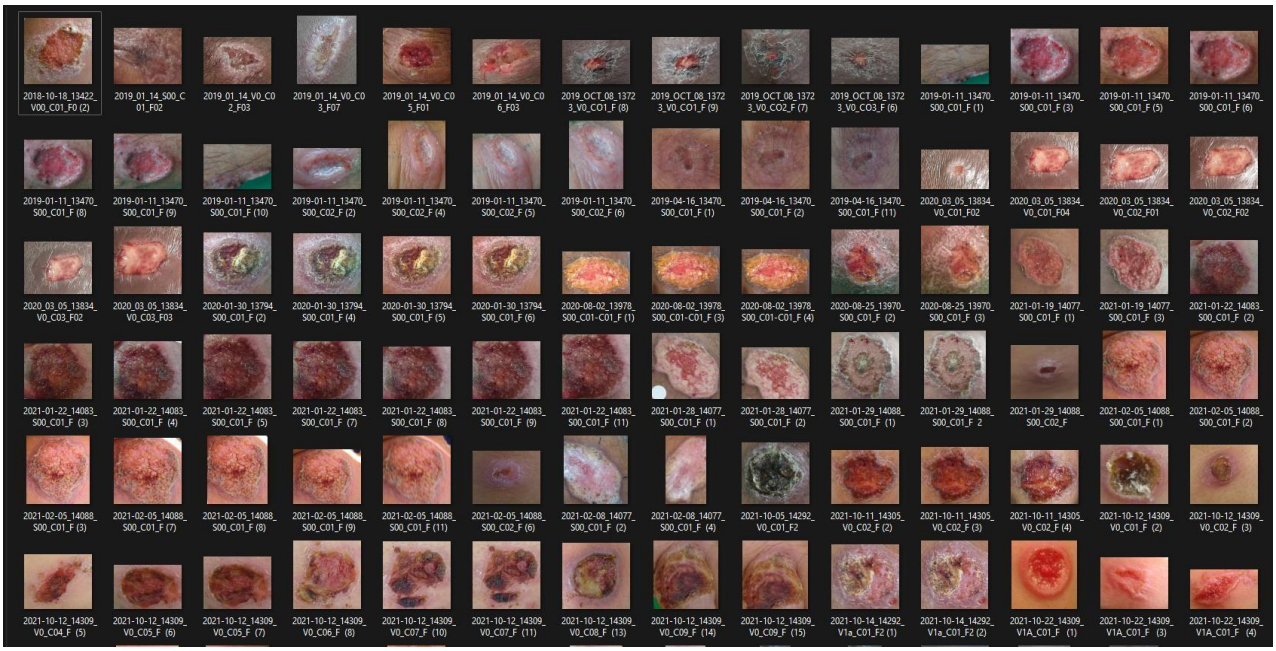

Figura 12. Muestra del set de datos de Leishmaniasis para Train

4.4 AUMENTO DE DATOS

Teniendo en cuenta que no se tiene un conjunto de datos tan grande (885 imágenes), se tomó la decisión de realizar un proceso de aumento de datos mediante programación, esto permitió aplicar diversas transformaciones sobre las entradas originales, obteniendo muestras ligeramente diferentes, lo que permite a la red desenvolverse mejor en la fase de inferencia.

Esta técnica se utiliza mucho en el campo de la visión artificial porque funciona de manera eficiente. Dentro de dicho contexto, una misma imagen de entrada es procesada por la red neuronal tantas veces como épocas se ejecuten en el entrenamiento. Lo que se hace es aplicar transformaciones de forma aleatoria cada vez que se vuelve a introducir la imagen a la red.

Ejemplos de transformaciones:

- Voltear la imagen en horizontal / vertical.
- Rotar la imagen X grados.
- Recortar, añadir relleno, redimensionar.

- Aplicar deformaciones de perspectiva.
- Ajustar brillo, contraste, saturación.
- Introducir ruido, defectos.
- Combinaciones de las anteriores.

Para el presente proyecto se utilizó específicamente el giro horizontal (voltea completamente la imagen), rango de zoom (20%) para hacer acercamiento a las imágenes y rango de corte (20%) para generar inclinaciones, de esta forma se aumentó la información para entrenamiento (se resalta que este aumento de datos fue solo para el conjunto de datos que corresponde a entrenamiento, 531 imágenes) sin necesidad de obtener muestras adicionales, y también sin alargar los tiempos. Lo mejor es que esta técnica permite que el modelo sea capaz de obtener buenos resultados para imágenes tomadas desde distintos ángulos o bajo distintas condiciones de luz. Por tanto, se consigue que la red no se sobreajuste y que generalice mejor.

5. CONSTRUCCIÓN DE LOS MODELOS Y OPTIMIZACIÓN

5.1 SELECCIÓN DE LAS TÉCNICAS DE APRENDIZAJE AUTOMÁTICO ADECUADAS

La técnica de aprendizaje automático seleccionada para el desarrollo de este proyecto es las redes neuronales convolucionales, esta resulta ser la adecuada para proyectos en donde se requiere hacer predicciones mediante imágenes, esto debido a que su aplicación está fundamentada en matrices bidimensionales, son muy efectivas para tareas de visión artificial, como en la clasificación y segmentación de imágenes, entre otras aplicaciones.

Teniendo en cuenta que las redes neuronales convolucionales (CNN) manejan un conjunto de términos bastante variado en sus diferentes etapas (Convolución, Kernel, Función de activación, MaxPooling, Dropout, Flatten, Dense), se estudió en detalle cada uno de estos términos y su funcionalidad dentro del proyecto.

Simultáneamente se desarrolló un Script inicial en Python haciendo uso de la librería de aprendizaje automático TensorFlow, además de Keras (la cual se ejecuta sobre TensorFlow) y es usada para el desarrollo de redes neuronales convolucionales, todo esto mediante el entorno de desarrollo de Google Colab.

Por otra parte, la arquitectura seleccionada para esta red neuronal convolucional es básica en el sentido de que no posee un número tan grande de capas y por ende tiene poca profundidad, la arquitectura que se trabajó es la siguiente:

Dos capas convolucionales con 64 neuronas cada una (el número de neuronas fue obtenido mediante búsqueda de hiperparámetros) con un kernel de 3x3 y función de activación Relu, después de cada capa convolucional hay una capa de MaxPooling con Pool_Size de 2x2, una capa de Dropout de 0.1 (porcentaje obtenido mediante búsqueda de hiperparámetros) en medio de las dos capas convolucionales, una capa Flatten, una capa densa de 128 neuronas (el número de neuronas fue obtenido mediante búsqueda de hiperparámetros) y función de activación relu y una capa de clasificación o salida binaria con 1 neurona y función de activación Sigmoid.

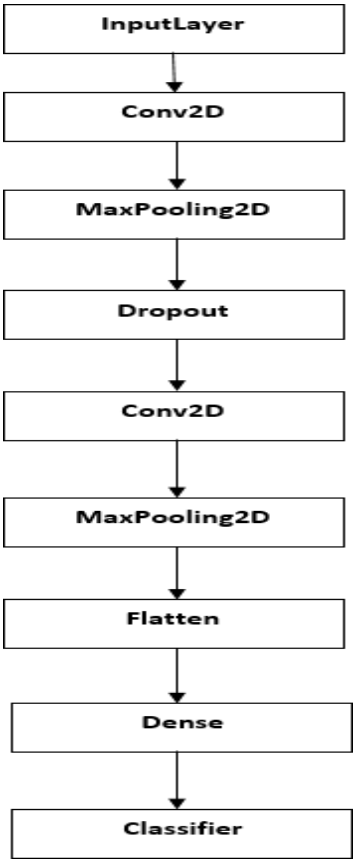

Figura 13. Diagrama de bloques de la arquitectura seleccionada

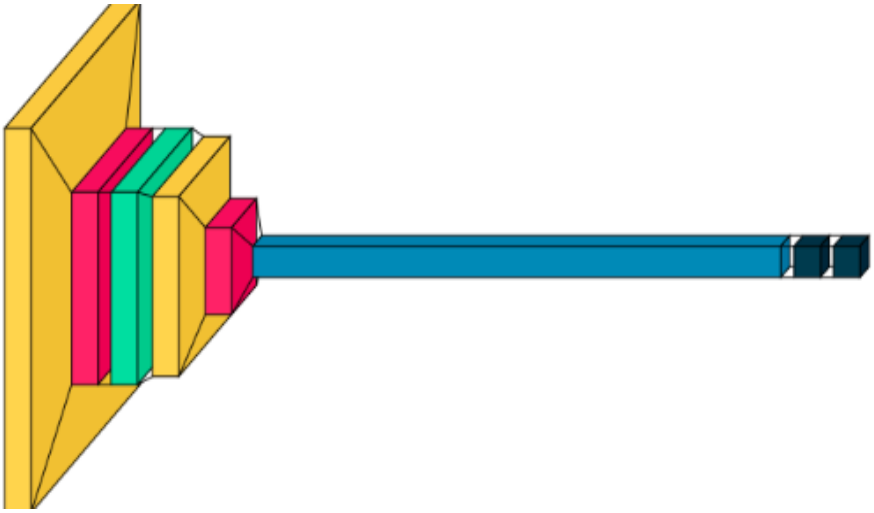

Figura 14. Visualización por capas de la arquitectura seleccionada

De izquierda a derecha, en amarillo las capas convolucionales, en rojo las capas de MaxPooling, en verde el Dropout, en azul la capa Flatten y en azul oscuro la capa densa y el clasificador.

5.2 AJUSTE DE HIPERPARÁMETROS

Se partió de una arquitectura como la ya expuesta, con base en esta se aplicó el ajuste del modelo (model tuning) haciendo uso de una función de búsqueda aleatoria de hiperparametros (RandomSearch), mediante el entrenamiento de varios modelos y experimentando entre las siguientes opciones:

Hiperparámetros a variar:
Neuronas capas convolucionales: 32, 64, 128
Neuronas capa densa: 128, 256, 512.
Optimizador Adam-Learning Rate: 0.1, 0.01, 0.001
Dropout: 0.1, 0.3, 0.5, 0.7

Hiperparámetros fijos:
Función de activación: Relu
Épocas: 50
Tamaño del lote: 32
Max_Trials: 50
Parada anticipada-Patience: 10

Los mejores hiperparámetros obtenidos fueron:

Hiperparámetro	Mejor resultado
Neuronas de las dos capas convolucionales	64
Neuronas de la capa densa	128
Rata de aprendizaje	0.001
Dropout	0.1

Tabla 7. Mejores hiperparámetros encontrados

Por otra parte, los parámetros de la red entrenada fueron:

Capa	Forma a la salida	# de parámetros
Conv2D	62, 62, 64	1.792
MaxPooling2D	31, 31, 64	0
Dropout	31, 31, 64	0
Conv2D	29, 29, 64	36.928
MaxPooling2D	14, 14, 64	0
Flatten	12.544	0
Dense	128	1’605.760
Classifier	1	129
Total de parámetros		**1’644.609**
Total de parámetros entrenados		**1’644.609**

Tabla 8. Parámetros de la red

Teniendo en cuenta los mejores hiperparámetros obtenidos, se evaluó la función de pérdida del modelo y el rendimiento mediante las siguientes gráficas:

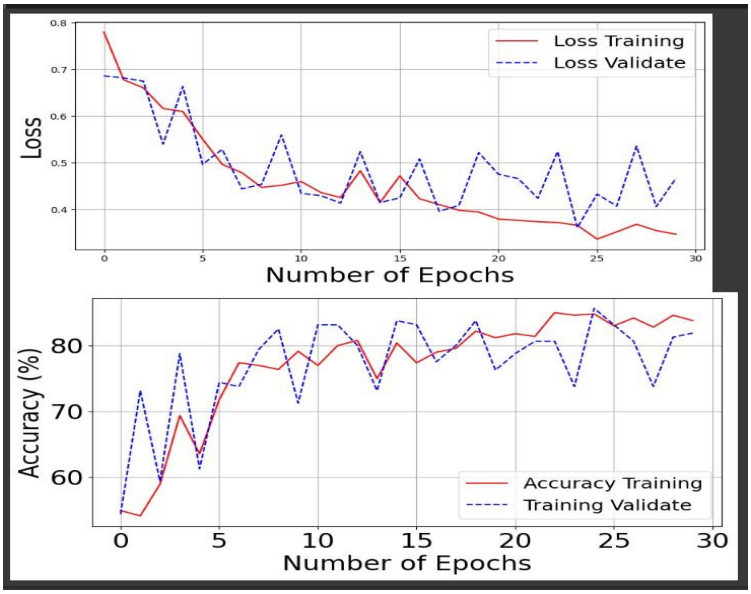

Figura 15. Gráfica de la función de pérdida (superior) y del rendimiento (inferior)

En la anterior gráfica, la función de pérdida permite apreciar un decaimiento adecuado conforme avanzan las épocas, más pronunciado para entrenamiento (rojo) que para validación (azul), la gráfica de validación sigue en gran medida a la gráfica del entrenamiento, excepto en la parte final en donde se separa un poco con tendencia a mantenerse constante (entre 0.4 y 0.5), sin considerarse esto un sobreajuste, la gráfica del rendimiento se comporta de manera incremental en entrenamiento y validación siendo bastante similares, llegando a un techo por debajo de 90%, con un acuraccy de aproximadamente 83%.

5.3 ENTRENAMIENTO DEL MODELO TOMANDO TRAIN+VALIDATION

Tomando los mejores hiperparametros obtenidos, se realizaron cinco (5) entrenamientos desde cero teniendo en cuenta que el Train en este caso fue la suma de Train y Validation (para cada entrenamiento se usaron variaciones aleatorias del set de datos de Train y Test), la distribución es la siguiente:

Categoría	Total datos 100%	Train 80%	Testeo 20%
Leishmaniasis	407	326	81
Otras	478	382	96
	885	**708**	**177**

Tabla 9. Distribución de los datos para entrenamiento y testeo

Los resultados de los cinco entrenamientos fueron promediados, obteniendo las siguientes métricas:

	0	1
0	**0,89**	**0,11**
1	**0,20**	**0,80**

Tabla 10. Matriz de confusión del modelo entrenado con los mejores hiperparámetros

Class	Precision	Recall	F1-Score	Support	Accuracy
0	0,7994	0,8987	0,8457	81	0,8497
1	0,9051	0,8083	0,8533	96	

Tabla 11. Métricas del modelo entrenado con los mejores hiperparámetros

Los anteriores resultados muestran una mejora en el accuracy (84.97%) al tomar como Train a Train + Validation, esto se debe a que el conjunto de imágenes que se toma para entrenar es mucho mayor en este caso y de esta forma se refuerza el entrenamiento del modelo, en general las demás métricas de este modelo fueron un punto de partida para buscar mejorar el rendimiento e incrementar el porcentaje de aciertos al momento de hacer la clasificación, cabe resaltar que las métricas de Precision, Recall y F1-Score fueron calculadas con base en las dos clases, Leishmaniasis (0) y Otras (1).

5.4 TRANSFER LEARNING VGG-16

La primer técnica de Transfer Learning usada para buscar optimizar los resultados del anterior modelo fue VGG-16**,** esta arquitectura ha revolucionado la eficiencia en el entrenamiento de modelos de aprendizaje profundo. Al aprovechar los conocimientos previos adquiridos por VGG-16 en grandes conjuntos de datos, se acelera significativamente el proceso de entrenamiento para tareas específicas. De igual forma se realizaron 5 entrenamientos los cuales fueron promediados, tras aplicar VGG-16 se llegó a los siguientes resultados:

	0	1
0	**0,90**	**0,10**
1	**0,11**	**0,89**

Tabla 12. Matriz de confusión aplicando Transfer Learning VGG-16

Class	Precision	Recall	F1-Score	Support	Accuracy
0	0,8811	0,9012	0,8892	81	0,8972
1	0,9170	0,8937	0,9037	96	

Tabla 13. Métricas aplicando Transfer Learning VGG-16

Los resultados que se obtuvieron aplicando VGG-16 muestran unas métricas (Accuracy, Precision, Recall, F1-Score) mejores a las entregadas por el modelo base (entrenado con Train + Validation), sin embargo, no son los resultados más adecuados puesto que se pretendía obtener un balance general de métricas por encima del 90% en ambas clases.

5.5 TRANSFER LEARNING VGG-19

La segunda técnica de Transfer Learning que se utilizó fue VGG-19, esta permite la transferencia de conocimientos aprendidos en grandes conjuntos de datos a tareas específicas, optimizando el tiempo de entrenamiento. Este enfoque es esencial para problemas donde los conjuntos de datos son limitados, ya que la red preentrenada puede adaptarse eficientemente a nuevas aplicaciones, mejorando la precisión y la eficacia del modelo, los resultados fueron los siguientes:

	0	1
0	**0,94**	**0,06**
1	**0,10**	**0,90**

Tabla 14. Matriz de confusión aplicando Transfer Learning VGG-19

Class	Precision	Recall	F1-Score	Support	Accuracy
0	**0,8937**	**0,9383**	**0,9147**	**81**	**0,9198**
1	**0,9464**	**0,9042**	**0,9241**	**96**	

Tabla 15. Métricas aplicando Transfer Learning VGG-19

Las métricas entregadas por VGG-19 fueron mejores a las que se obtuvo utilizando Transfer Learning con VGG-16, lo cual lo perfiló como uno de los mejores modelos en este punto ya que la mayoría de métricas están por encima del 90% tanto para la clase Leishmaniasis como para la clase Otras.

5.6 TRANSFER LEARNING RESNET-152

Para finalizar, se aplicó una última técnica de Transfer Learning la cual contiene un gran número de capas de profundidad y que podría mejorar los resultados del modelo, los resultados fueron los siguientes:

	0	1
0	**0,56**	**0,44**
1	**0,35**	**0,65**

Tabla 16. Matriz de confusión aplicando Transfer Learning ResNet-152

Class	Precision	Recall	F1-Score	Support	Accuracy
0	**0,6360**	**0,5679**	**0,5424**	**81**	**0,6124**
1	**0,6996**	**0,6500**	**0,6054**	**96**	

Tabla 17. Métricas aplicando Transfer Learning ResNet-152

Tras la obtención de las anteriores métricas, se pudo establecer que la aplicación de Transfer Learning no garantiza que los resultados del nuevo modelo entrenado mejoren, en el caso de ResNet-152 ocurrió lo contrario que en VGG-16 y VGG-19, los resultados empeoraron significativamente para ambas clases como se puede apreciar en las métricas, esto es atribuible a que la profundidad de este modelo preentrenado (150 capas convolucionales) es bastante grande en comparación con la cantidad de información (imágenes) que se manejó y para el numero de clases que es solo de dos.

5.7 UTILIZACIÓN DE FINE TUNING

Adicionalmente al Transfer Learning existe una solución más en la búsqueda de optimizar los resultados obtenidos de la anterior experimentación, Fine Tuning ofrece la opción de descongelar una o varias de las capas convolucionales para que sean entrenadas también en el proceso, opción que puede hacer que los resultados varíen, pero nuevamente como en Transfer Learning existe la posibilidad de que estos mejoren o de que empeoren.

Se procedió a aplicar Fine Tuning únicamente a los modelos que al ser entrenados presentaron mejoras, como fue VGG-16 y VGG-19, teniendo en cuenta que para cada uno de ellos se realizaron 5 entrenamientos y se generó su respectivo promedio.

5.7.1 FINE TUNING A VGG-16

Teniendo en cuenta el planteamiento realizado para Transfer Learning con VGG-16, se procedió a aplicar sobre este el respectivo Fine Tuning, lo cual permite retener los conocimientos generales aprendidos por VGG-16 y afinar las capas finales, esto puede logra una mayor especialización y rendimiento en aplicaciones más específicas, los resultados fueron los siguientes:

	0	1
0	**0,98**	**0,02**
1	**0,04**	**0,96**

Tabla 18. Matriz de confusión aplicando Fine Tuning VGG-16

Class	Precision	Recall	F1-Score	Support	Accuracy
0	**0,9547**	**0,9827**	**0,9684**	**81**	**0,9707**
1	**0,9852**	**0,9604**	**0,9729**	**96**	

Tabla 19. Métricas aplicando Fine Tuning VGG-16

Los anteriores resultados haciendo uso de Fine Tuning muestran una mejora considerable en comparación a VGG-16 con transfer Learning, para este caso se descongeló una sola capa convolucional (la última), la aplicación de la técnica fue positiva para la mejora del rendimiento en este modelo en sus diferentes métricas, con un porcentaje de aciertos para la clase principal (Leishmaniasis) equivalente al 98.27% y de 96.04% para la clase secundaria (Otras), lo cual lo convierte en el mejor modelo en todo el proceso de experimentación que se realizó en este proyecto.

5.7.2 FINE TUNING A VGG-19

Como último experimento, se aplicó Fine Tuning a Transfer Learning VGG-19, esto implica ajustar las capas superiores de la red preentrenada para adaptarse a una tarea específica. Este proceso es crucial para personalizar el modelo a las características únicas del nuevo conjunto de datos, optimizando así su rendimiento, a continuación los resultados:

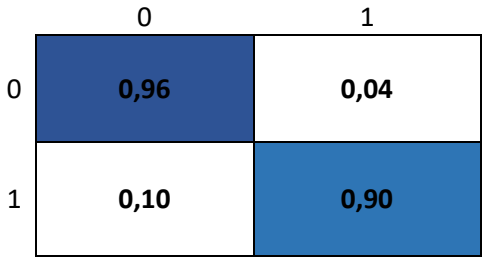

Figura 20. Matriz de confusión aplicando Fine Tuning VGG-19

Class	Precision	Recall	F1-Score	Support	Accuracy
0	0,8964	0,9654	0,9278	81	0,9300
1	0,9697	0,9000	0,9317	96	

Tabla 21. Métricas aplicando Fine Tuning VGG-19

Los resultados muestran que para este caso la aplicación de Fine Tuning también optimizó el modelo, logrando un porcentaje de aciertos para la clase principal (Leishmaniasis) equivalente al 96.54% y de 90.00% para la clase secundaria (Otras), se resalta que de igual forma que con el Fine Tuning de VGG-16 se descongeló sola la última capa convolucional.

6. ANÁLISIS DE RESULTADOS

Previo a la arquitectura que se planteó en el presente documento, se experimentó con un mayor número de capas, lo cual entregó resultados deficientes y que fueron difíciles de optimizar, el sobre ajuste estuvo presente y los resultados de las métricas (Accuracy, Precision, F1-Score, Recall) siempre fueron bajos y difíciles de balancear, además, las clases siempre se comportaron de manera dispar en la matriz de confusión. Estos resultados previos nos mostraron que cuando se quiere realizar una clasificación binaria como en este caso donde solo se tienen dos clases (Leishmaniasis y Otras), la mejor opción es usar una arquitectura básica, de pocas capas, pocas neuronas y poca profundidad, caso contrario si se tiene un gran número de clases probablemente la mejor opción sea usar un número de capas considerable que permita una mayor profundidad del modelo, todo ello sujeto a una correcta validación de sus métricas.

Tras seleccionar la arquitectura expuesta en el principio del anterior capítulo, se implementó una búsqueda aleatoria de hiperparámetros la cual permitió encontrar aquellos que entregaban el mejor desempeño, proceso que implicó un tiempo considerable, ya que de manera automática se realizan múltiples entrenamientos probando diferentes combinaciones de hiperparámetros según las opciones que se hayan especificado hasta entregar los mejores al final, además, los resultados del modelo con los mejores hiperparámetros permitieron apreciar que no hay sobre ajuste como se puede ver en la gráfica de la función de pérdida y en la gráfica del rendimiento, la cual muestra un Accuracy de aproximadamente 83%, lo cual fue un buen punto de partida.

La combinación del conjunto de datos de Train y Validation en uno solo es un proceso que se suele realizar cuando ya se han encontrado los hiperparámetros adecuados y se desea mejorar un poco más el rendimiento del modelo incrementando el número de datos de entrenamiento, efectivamente al usar este nuevo conjunto de datos de Train el Accuracy mejoró subiendo a 84.97%, sin embargo, la Precisión del modelo para la clase principal (Leishmaniasis) estaba cerca de 79.94% lo cual no se puede considerar un valor tan bueno y por otra parte el porcentaje de aciertos de la clase secundaria (Otras) fue apenas del 80.83%, por lo cual, optimizar los resultados de todas las métricas es de vital importancia para garantizar mayor solidez y confiabilidad en las predicciones del modelo para las dos clases.

Class	Precision	Recall	F1-Score	Support	Accuracy
0	0,7994	0,8987	0,8457	81	0,8497
1	0,9051	0,8083	0,8533	96	

Tabla 22. Métricas del modelo entrenado con los mejores hiperparámetros

La técnica del Transfer Learning mediante VGG-16 y VGG-19 fue de gran ayuda para optimizar los resultados anteriormente expuestos, siendo VGG-19 el que entregó las mejores métricas, con un

Accuracy de 91.98%, un porcentaje de aciertos ya más parejo para ambas clases, 93.83% para Leishmaniasis y 90.42% para Otras, pero también se tenía una precisión que seguía estando por debajo de 90% en el caso de Leishmaniasis. Por otra parte, también se pudo apreciar que la utilización de ResNet-152 empeoró los resultados del modelo base en la mayoría de métricas de ambas clases, esto es atribuible a que la profundidad de este modelo preentrenado (150 capas convolucionales) es bastante grande en comparación con la cantidad de información (imágenes) que se manejó y para el numero de clases que es solo de dos, entonces, se pudo concluir que hacer uso de modelos pre entrenados mediante Transfer Learning no garantiza que los resultados del nuevo modelo sean mejores, así que siempre es necesario llevar a cabo la respectiva experimentación para poder tener certeza a través de los resultados numéricos obtenidos.

Class	Precision	Recall	F1-Score	Support	Accuracy
0	0,8937	0,9383	0,9147	81	0,9198
1	0,9464	0,9042	0,9241	96	

Tabla 23. Métricas aplicando Transfer Learning VGG-19

Class	Precision	Recall	F1-Score	Support	Accuracy
0	0,6360	0,5679	0,5424	81	0,6124
1	0,6996	0,6500	0,6054	96	

Tabla 24. Métricas aplicando Transfer Learning ResNet-152

Como recurso final para buscar mejorar un poco más las métricas obtenidas de la etapa anterior, se empleó Fine Tuning, tanto a VGG-16 como a VGG-19, de estos dos se encontró que las mejores métricas fueron para VGG-16 con Fine Tuning a una de las capas convolucionales, con este modelo se logró un buen balance de las 4 métricas (Accuracy, Precision, Recall, F1-Score) estando estas por encima de 90% en las dos clases, el porcentaje de aciertos de la clase principal Leishmaniasis fue de 98.27% y para la clase Otras de 96.04%, el Accuracy fue de 97.07%, Precision de 95.47% y F1-Score de 96.84% en el caso de la clase principal, estos resultados son garantía de confiabilidad por parte del modelo al momento de realizar una predicción para saber si una lesión cutánea corresponde o no a Leishmaniasis. Nuevamente al igual que en el caso de Transfer Learning, hacer uso de Fine Tuning no garantiza que los resultados mejoren, puede que sí o puede que no, por lo que realizar la experimentación es necesario para dar un concepto claro.

Class	Precision	Recall	F1-Score	Support	Accuracy
0	0,9547	0,9827	0,9684	81	0,9707
1	0,9852	0,9604	0,9729	96	

Tabla 25. Métricas aplicando Fine Tuning VGG-16

7. DESPLIEGUE DEL MODELO

Se hizo necesario desarrollar una interfaz gráfica mediante la cual un usuario pueda probar la funcionalidad del modelo y visualizar los resultados de una forma práctica y entendible, esto pensando en que personal de la salud que tenga que lidiar con pacientes con la enfermedad de la Leishmaniasis puedan realizar predicciones mediante fotografías de la lesión, todo ello a través del presente aplicativo, el cual resulta bastante práctico e intuitivo de utilizar.

Esta interfaz gráfica se desarrolló implementando el modelo que entregó los mejores resultados (VGG-16 con Fine Tuning), fue realizada haciendo uso de Python y Streamlit, cabe resaltar que Streamlit es un "framework" de Python de código abierto que permite de manera sencilla e integrada desarrollar aplicaciones web gracias a la interacción con otras librerías para su empleo en campos de la teledetección, ciencia de datos, etc. Streamlit fue desarrollado para facilitar a los científicos de datos y desarrolladores la creación de interfaces de usuario para sus proyectos, sin requerir conocimientos profundos en desarrollo web.

El presente aplicativo fue probado de manera local haciendo uso del navegador web Google Chrome, sin embargo, también puede ser montado en un servidor en línea para que se pueda utilizar en cualquier momento desde cualquier parte del mundo mediante conexión a internet.

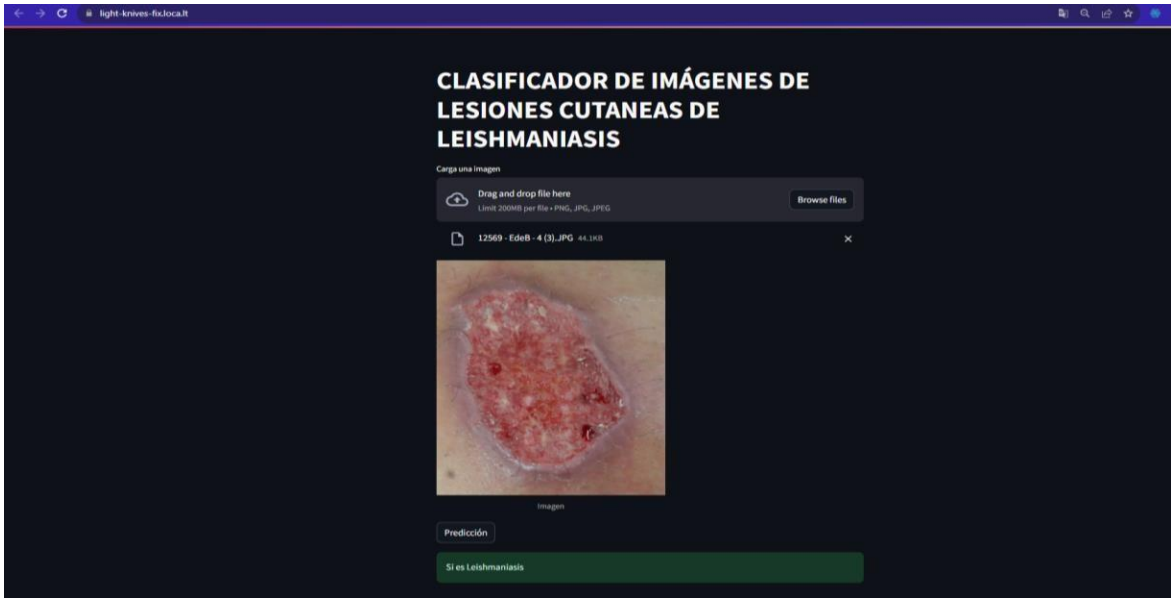

Figura 16. Interfaz gráfica

Mediante esta aplicación es posible cargar una fotografía de la lesión cutánea que se quiera evaluar y oprimiendo un botón se procede a realizar la predicción usando el modelo precargado para de esta forma saber si es o no Leishmaniasis, el procedimiento puede ser repetido tantas veces como se requiera con las imágenes que se disponga validar.

8. CONCLUSIONES Y TRABAJOS FUTUROS

8.1. CONCLUSIONES

- Se logró preprocesar la base de datos de imágenes suministrada por el CIDEIM (1.386 imágenes), imágenes correspondientes a lesiones cutáneas de la Leishmaniasis, proceso que implicó seleccionar y editar dichas imágenes para llegar a un conjunto final de 407 imágenes con la calidad suficiente para una correcta realización de los entrenamientos de los modelos.

- Mediante redes neuronales convolucionales se puede entrenar modelos de alta precisión, por medio de los cuales es posible realizar la clasificación de imágenes de lesiones cutáneas de la Leishmaniasis y de otro tipo de lesiones en específico, ya sea mediante un sistema binario como es el caso de este proyecto, o mediante un sistema categórico de más de dos clases según sea el caso.

- Al aplicar Transfer Learning mediante VGG-16 y VGG-19 para buscar optimizar los resultados obtenidos con el modelo que se entrenó con los mejores hiperparámetros, se logró mejorar dichos resultados de manera significativa en varias de sus métricas.

- Teniendo en cuenta que al aplicar Transfer Learning con VGG-16 y VGG-19 optimizó los resultados del modelo, se procedió a realizar Fine Tuning para buscar mejorarlos un poco más, lo cual generó buenos resultados para el caso de VGG-16 en las diferentes métricas, sin embargo, los resultados de VGG-19 empeoraron.

- Tomando los resultados del presente proyecto, se puede decir que, al usar modelos pre entrenados como base, puede ayudar a optimizar los resultados de una red neuronal convolucional en algunos casos, por lo cual, siempre es necesario realizar la experimentación correspondiente ya que cabe la posibilidad de que los resultados no terminen mejorando, sino que tiendan a empeorar.

- Entre los diferentes modelos obtenidos en la experimentación, se logró llegar a un modelo con un porcentaje de acierto del 100% al momento de hacer predicciones de la clase

principal que es Leishmaniasis y en general con métricas por encima del 90% (accuracy, precision, f1-score y recall).

- Se logró hacer el despliegue del mejor modelo obtenido, programando una interfaz gráfica que permite probar el modelo de una forma ágil e intuitiva para que este pueda ser usado de manera local por aquellas personas que quieran realizar un diagnóstico de la enfermedad de la Leishmaniasis mediante imágenes de lesiones cutáneas.

8.2. TRABAJOS FUTUROS

- Se recomienda realizar el montaje en un servidor de la interfaz web que se desarrolló junto al mejor modelo obtenido, para de este modo poder utilizarla desde cualquier lugar en donde se disponga de una conexión a internet y así poder diagnosticar de manera remota si una lesión cutánea corresponde o no a Lesihmaniasis mediante fotografías de dicha lesión.

- Desarrollar una aplicación móvil que permita usar el modelo implementado junto a la cámara del teléfono celular y de esta forma realizar la predicción de una forma más ágil.

- Realizar un clasificador categórico en donde exista una gama de lesiones cutáneas mucho más grande (no solo Lesihmaniasis y Otras), lo cual podría ayudar a realizar diagnósticos de múltiples enfermedades, de las cuales no se tenga mucho conocimiento por parte del personal de la salud que esté evaluando dichas lesiones en un momento determinado.

9. REFERENCIAS BIBLIOGRÁFICAS

[1] M. Romero Peñuela y J.A. Sánchez Valencia, “Una Mirada a la Epidemiología y al Control de la Leishmaniasis Zoonótica en Colombia”. Biosalud, volumen 6, 2007; págs. 99-105.

[2] J. García Ciudad, “Clasificación de Imágenes de Lesiones Cutáneas Mediante Redes Neuronales Convolucionales y Aprendizaje Profundo”. Universidad de Málaga, E.T.S. Ingeniería Informática, 2020; págs. 7-14.

[3] J. Espasa Rosell, “Deep Learning en la Detección de Lesiones Cutáneas Malignas”. Escuela Técnica Superior de Ingeniería Industrial de Barcelona, 2022; págs. 20-35.

[4] J. Torres, “Deep Learning – Introducción práctica con Keras”. Universidad Politécnica de Cataluña, Campus Nord, módulo C6, 2018; págs. 65-158.

[5] C. Hernández, C. Ovalle, J. Soto, S. Valadas, “Atlas interactivo de Lesihamaniasis en las américas: aspectos clínicos y diagnósticos diferenciales”. Organización Panamericana de la salud, 2020; págs. 223-347.

[6] «Adadelta» https://keras.io/api/optimizers/adadelta/ (accedido 20 de septiembre de 2023).

[7] «Adam, Adaptive Moment Estimation» https://interactivechaos.com/es/manual/tutorial-de-machine-learning/adam (accedido 20 de septiembre de 2023).

[8] «VGG 16 Easiest Explanation» https://medium.com/nerd-for-tech/vgg-16-easiest-explanation-12453b599526 (accedido 02 de octubre de 2023).

[9] «machine learning & knowledge extraction Focal Liver Lesion Detection in Ultrasound Image Using Deep Feature Fusions and Super Resolution» https://www.researchgate.net/figure/Illustration-of-fine-tuned-VGG19-pre-trained-CNN-model_fig1_342815128 (accedido 02 de octubre de 2023).

[10] «Deep neural networks show an equivalent and often superior performance to dermatologists in onychomycosis diagnosis: Automatic construction of onychomycosis datasets by region-based convolutional deep neural network» https://www.researchgate.net/figure/The-representation-of-model-architecture-image-for-ResNet-152-VGG-19-and-two-layered_fig2_322621180 (accedido 02 de octubre de 2023).

[11] P. Acero Díaz, “Aplicación móvil para detección de melanoma con redes neuronales artificiales”. Universidad internacional de la Rioja (UNIR). Ingeniería Informática, 2021; págs. 13-69.

[12] S. Zuluaga Gamboa, “Aprendizaje de maquina aplicado a la predicción del éxito del tratamiento de la Leishmaniasis”. Universidad Pontificia Javeriana Cali. Ingeniería de sistemas y computación, 2022; págs. 3-30.

[13] «VGG: ¿Qué es este modelo?» https://datascientest.com/es/vgg-que-es-este-modelo-daniel-te-lo-cuenta-todo (accedido 02 de octubre de 2023).

[14] «ResNet-152» https://huggingface.co/microsoft/resnet-152 (accedido 02 de octubre de 2023).

ANEXOS

Script implementado

```
# Red Neuronal Convolucional (CNN) para identificar imágenes de lesiones cutaneas de Leishmaniasis

#Desarrollado por: Ing. Camilo Castro Duarte
#Maestría en ciencia de datos
#Universidad Pontificia Javeriana Cali
"""

!pip install keras-tuner --upgrade

"""#Importación de librerias"""

# Commented out IPython magic to ensure Python compatibility.
#Carga de las librerías
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.models import load_model
from tensorflow.keras.applications.imagenet_utils import preprocess_input, decode_predictions
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout, BatchNormalization, Input
from sklearn.metrics import confusion_matrix, f1_score, roc_curve, precision_score, recall_score, accuracy_score, roc_auc_score
from sklearn import metrics
from sklearn.model_selection import train_test_split, RandomizedSearchCV
#from keras.utils.vis_utils import plot_model
from mlxtend.plotting import plot_confusion_matrix
import os
import numpy as np
import matplotlib.pyplot as plt
```

```
from keras_tuner import RandomSearch
# %matplotlib inline

from google.colab import drive
drive.mount('/content/drive')

from keras.preprocessing.image import ImageDataGenerator

tf.__version___

"""# Especificación del directorio"""

batch_size = 32
width_shape = 64
height_shape = 64
directory = '/content/drive/MyDrive/Maestria_pruebas/datasets/Leishmaniasis_2/'

"""#Carga de datos para entrenamiento con aumento de datos"""

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
training_set = train_datagen.flow_from_directory(directory +'train',
                                                 target_size = (width_shape, height_shape),
                                                 batch_size = 32,
                                                 class_mode = 'binary')

"""# Carga de datos para validación"""

valid_datagen = ImageDataGenerator(rescale = 1./255)
valid_set = valid_datagen.flow_from_directory(directory + 'validation',
                                              target_size = (width_shape, height_shape),
                                              batch_size = 32,
                                              class_mode = 'binary')
```

```
"""# Configuración de la Red Neuronal Convolucional para la busqueda de los mejores hiperparámetros"""

def build_model(hp):

  cnn = tf.keras.models.Sequential()

  cnn.add(tf.keras.layers.Conv2D(filters= hp.Choice('filters1', [32, 64, 128]), kernel_size=3, activation='relu', input_shape=[width_shape, height_shape, 3]))

  cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

  cnn.add(tf.keras.layers.Dropout(rate= hp.Choice('rate', [0.1, 0.3, 0.5, 0.7])))

  cnn.add(tf.keras.layers.Conv2D(filters= hp.Choice('filters2', [32, 64, 128]), kernel_size=3, activation='relu'))

  cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

  cnn.add(tf.keras.layers.Flatten())

  cnn.add(tf.keras.layers.Dense(units= hp.Choice('units', [128, 256, 512]), activation='relu'))

  cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))


  optimizer=tf.keras.optimizers.Adam(learning_rate= hp.Choice('learning_rate', [0.1, 0.01, 0.001]))
  cnn.compile(optimizer = optimizer, loss = 'binary_crossentropy', metrics = ['accuracy'])

  return cnn

"""# Entrenamiento y validación para la busqueda de los mejores hiperparámetros"""

tuner = RandomSearch(build_model, objective = 'val_loss', max_trials = 50) #Max trials (numero total de pruebas), por defecto es 10
```

```
checkpoint_cb = tf.keras.callbacks.ModelCheckpoint(directory+"my_keras_model.h5",
                                save_best_only=True)
early_stopping_cb = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10,          #
Parada anticipada, evita el sobre ajuste
                                restore_best_weights=True)

tuner.search(training_set, validation_data = valid_set, epochs = 50,
             callbacks=[checkpoint_cb, early_stopping_cb])

best_model = tuner.get_best_models()[0]

#Verificación de los mejores hiperparametros
print(best_model.layers[0].filters, best_model.layers[1].filters)
print(best_model.layers[0].rate)
print(best_model.layers[0].units)
print(best_model.optimizer.learning_rate)

test_loss, test_acc = best_model.evaluate(valid_set)
print('test_acc:', test_acc)

"""#Entrenamiento desde cero del modelo con los mejores hiperparámetros encontrados, usando
como Train a Train + Validation"""

cnn = tf.keras.models.Sequential()

cnn.add(tf.keras.layers.Conv2D(filters=64,              kernel_size=3,              activation='relu',
input_shape=[width_shape, height_shape, 3]))

cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

cnn.add(tf.keras.layers.Dropout(rate=0.1))

cnn.add(tf.keras.layers.Conv2D(filters=64, kernel_size=3, activation='relu'))

cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

cnn.add(tf.keras.layers.Flatten())
```

```
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))

cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

optimizer=tf.keras.optimizers.Adam(learning_rate= 0.001)
cnn.compile(optimizer = optimizer, loss = 'binary_crossentropy', metrics = ['accuracy'])

tf.keras.utils.plot_model(cnn)

history = cnn.fit(x = training_set, epochs = 50)

#Exportar el modelo en formato h5
cnn.save('leishmaniasis.h5')

!ls

cnn.summary()

#Matriz de confusión para evaluar el modelo entrenado

names = ['leishmaniasis','otras']

test_data_dir = '/content/drive/MyDrive/Maestria_pruebas/datasets/Leishmaniasis_2/test'
test_datagen = ImageDataGenerator(rescale= 1./255)

test_generator = test_datagen.flow_from_directory(
   test_data_dir,
   target_size=(width_shape, height_shape),
   batch_size = batch_size,
   class_mode='binary',
   shuffle=False)


predictions = cnn.predict_generator(generator=test_generator)

y_real = test_generator.classes
```

```
#genero las etiquetas para cada predicción
y_pred =predictions[:,0]>0.5

#Generación de la matriz

matc=metrics.confusion_matrix(y_real, y_pred)

print(metrics.classification_report(y_real,y_pred, digits = 4))

plot_confusion_matrix(conf_mat=matc, figsize=(9,9), colorbar=True, show_absolute=False, show_normed=True)
plt.show()

#Exactitud
exactitud = accuracy_score(y_real, y_pred)
print(exactitud)

#Precision
precision = precision_score(y_real, y_pred, pos_label=0)
print(precision)

#Sensibilidad
sensibilidad = recall_score(y_real, y_pred, pos_label=0)
print(sensibilidad)

#F1
puntaje = f1_score(y_real, y_pred, pos_label=0)
print(puntaje)

!pip install visualkeras

#Visualizamos las capas del modelo entrenado

import visualkeras
visualkeras.layered_view(cnn)
```

```
"""#Transfer Learning y Fine Tuning con VGG16"""

#Cargando el modelo preentrenado
pre_trained_model = VGG16(input_shape=(height_shape, width_shape, 3), include_top = False,
weights = 'imagenet')

#Se verifica la estructura del modelo preentrenado
pre_trained_model.summary()

#Se congelan las capas convolucionales del modelo preentrenado para que no sean entrenadas
nuevamente a exepción de la ultima capa convolucional la cual si se entrena  -  Fine Tuning

pre_trained_model.trainable = True
set_trainable = False

for layer in pre_trained_model.layers:
  if layer.name == 'block5_conv1':
    set_trainable = True
  if set_trainable:
    layer.trainable = True
  else:
    layer.trainable = False

cnn = tf.keras.models.Sequential()

cnn.add(pre_trained_model)

cnn.add(tf.keras.layers.Flatten())

cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))

cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

optimizer=tf.keras.optimizers.Adam(learning_rate= 0.001)
cnn.compile(optimizer = optimizer, loss = 'binary_crossentropy', metrics = ['accuracy'])

history = cnn.fit(x = training_set, epochs = 50)
```

```
#Exportar el modelo en formato h5
cnn.save('leishmaniasis.h5')

!ls

cnn.summary()

#Matriz de confusión para evaluar el modelo entrenado

names = ['leishmaniasis','otras']

test_data_dir = '/content/drive/MyDrive/Maestria_pruebas/datasets/Leishmaniasis_2/test'
test_datagen = ImageDataGenerator(rescale= 1./255)

test_generator = test_datagen.flow_from_directory(
   test_data_dir,
   target_size=(width_shape, height_shape),
   batch_size = batch_size,
   class_mode='binary',
   shuffle=False)


predictions = cnn.predict_generator(generator=test_generator)

y_real = test_generator.classes

#genero las etiquetas para cada predicción
y_pred =predictions[:,0]>0.5

#Generación de la matriz

matc=metrics.confusion_matrix(y_real, y_pred)

print(metrics.classification_report(y_real,y_pred, digits = 4))
```

```
plot_confusion_matrix(conf_mat=matc, figsize=(9,9), colorbar=True, show_absolute=False, show_normed=True)
plt.show()

#Exactitud
exactitud = accuracy_score(y_real, y_pred)
print(exactitud)

#Precision
precision = precision_score(y_real, y_pred, pos_label=0)
print(precision)

#Sensibilidad
sensibilidad = recall_score(y_real, y_pred, pos_label=0)
print(sensibilidad)

#F1
puntaje = f1_score(y_real, y_pred, pos_label=0)
print(puntaje)

!pip install visualkeras

#Visualizamos las capas del modelo entrenado

import visualkeras
visualkeras.layered_view(cnn)

"""#Probando el modelo con imágenes especificas"""

#Carga del modelo para probarlo
longitud, altura = 64, 64

modelo = '/content/drive/MyDrive/Maestria_pruebas/modelo/leishmaniasis.h5'

cnn = load_model(modelo)

names = ['leishmaniasis','otras']
```

```
test_data_dir = '/content/drive/MyDrive/Maestria_pruebas/imagenes2'
test_datagen = ImageDataGenerator(rescale= 1./255)

test_generator = test_datagen.flow_from_directory(
   test_data_dir,
   target_size=(width_shape, height_shape),
   batch_size = batch_size,
   class_mode='binary',
   shuffle=False)


predictions = cnn.predict(test_generator)
predictions

y_pred =predictions[:,0]>0.5
y_pred

y_pred =predictions[:,0]>0.5
y_pred
if y_pred == False:
   print("pred: si es leishmaniasis")
elif y_pred == True:
   print("pred: no es leishmaniasis")

"""#Convirtiendo el modelo a Tensorflow JS"""

#Exportar el modelo en formato h5
cnn.save('leishmaniasis.h5')

#Para convertirlo a tensorflow.js, primero debemos instalar la libreria
!pip install tensorflowjs

#Crear carpeta donde se colocaran los archivos resultantes
!mkdir carpeta_salida

#Libreria necesaria
```

```
!pip install jax==0.4.14

#Realizar la exportacion a la carpeta de salida
!tensorflowjs_converter --input_format keras leishmaniasis.h5 carpeta_salida

#Confirmar que en la carpeta de salida se hayan generado los archivos. Deben aparecer archivos
"bin" y "json"
!ls carpeta_salida
```

Script interfaz gráfica

```
#Interfáz gráfica en Streamlit para probar nuestro modelo entrenado para el reconocimiento de lesiones cutaneas de la Leishmaniasis

#Desarrollado por: Ing. Camilo Castro Duarte
#Maestría en ciencia de datos
#Universidad Pontificia Javeriana Cali

##Instalación de paquetes
"""

!pip install streamlit -q

!pip install streamlit-lottie

!pip install Pillow

#%%writefile /usr/local/lib/python3.10/dist-packages/colab_kernel_launcher.py

import streamlit as st
import requests
from streamlit_lottie import st_lottie
from PIL import Image
from skimage.transform import resize

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.models import load_model
from tensorflow.keras.applications.imagenet_utils import preprocess_input, decode_predictions
import os
import numpy as np
```

```
#La pagina

#st.write("""My first app Hello *world*""")
MODEL_PATH= '/content/leishmaniasis.h5'


#Dimensiones de la imágene de entrada
width_shape = 64
height_shape = 64


#Clases
names = ['leishmaniasis', 'otras']



# Se recibe la imagen y el modelo, devuelve la predicción
def model_prediction(img, model):


   img_resize = resize(img, (width_shape, height_shape))


   x = np.expand_dims(img_resize, axis=0)


   preds = model.predict(x)

   y_pred  =preds[:,0]>0.5

   if y_pred == False:
     y_respuesta= "Si es Leishmaniasis"
    #y_respuesta= x
   elif y_pred == True:
    y_respuesta= "No es Leishmaniasis"
    #y_respuesta= x
```

```
    return y_respuesta


def main():

    model=''

    # Se carga el modelo
    if model=='':
        model = load_model(MODEL_PATH)

    st.title("CLASIFICADOR DE IMÁGENES DE LESIONES CUTANEAS DE LEISHMANIASIS")

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

    # El usuario carga una imagen
    if img_file_buffer is not None:
        image = np.array(Image.open(img_file_buffer))
        st.image(image, caption="Imagen", use_column_width=False)

    # El botón predicción se usa para iniciar el procesamiento
    if st.button("Predicción"):
         predictS = model_prediction(image, model)
         st.success(predictS)


if __name__== '_main_':
    main()

!wget koboldai

"""##Validación de mi IP"""

print("#############")
```

```
print("IP v")
!curl https://ipv4.icanhazip.com/
print("#############")

"""##Corremos la aplicación"""

!streamlit run /usr/local/lib/python3.10/dist-packages/colab_kernel_launcher.py & npx localtunnel --port 8501
```