"Modelo predictivo para determinar el desenlace terapéutico del paciente con leishmaniasis a partir de imágenes de lesiones"

Jhon Alexander Segura Dorado
ID.8975681

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.

Directora
Gloria Inés Álvarez Ph.D

Jurado
María Constanza Pabón

Jurado
Mario Julián Mora

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

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

JUAN CARLOS MARTÍNEZ ARIAS
Director Posgrados de Ingeniería y Ciencias

Cali,15/02/2023

Santiago de Cali, 26 de 02 de 2024

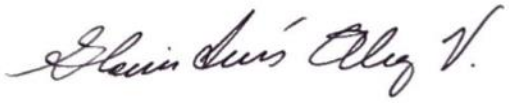
Ingeniero:
Juan Carlos Martínez Arias
Director Posgrados de Ingeniería
Facultad de Ingeniería y Ciencias
Pontificia Universidad Javeriana - Cali

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 **Modelo predictivo para determinar el desenlace terapéutico del paciente con leishmaniasis a partir de imágenes de lesiones** , el cual será realizado por el (la) estudiante Jhon Alexander Segura Dorado con código 8975681, bajo la dirección del profesor Gloria Inés Álvarez.

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
Jhon Alexander Segura Dorado
C.C. 1061809809 de Popayán



Firma
Gloria Inés Alvarez Vargas Ph.D
C.C. 30306105 de Manizales

FICHA RESUMEN
TRABAJO DE GRADO DE MAESTRÍA

TITULO: "Modelo predictivo para determinar el desenlace terapéutico del paciente con leishmaniasis a partir de imágenes de lesiones"

1. ÉNFASIS: N/A
2. TIPO DE PROYECTO: Aplicado
3. ÁREA DE TRABAJO: Machine Learning (Modelamiento y recuperación de datos e información)
4. ESTUDIANTE (S): Jhon Alexander Segura Dorado
5. CORREO ELECTRÓNICO:jhonsegura25@javerianacali.edu.co
6. DIRECCIÓN Y TELÉFONO: Cr 21c N° 16-43 (Popayán), 3106527768
7. DIRECTOR: Gloria Inés Álvarez
8. VINCULACIÓN DEL DIRECTOR (en la universidad): Planta
9. CORREO ELECTRÓNICO DEL DIRECTOR: galvarez@javerianacali.edu.co
10. CO-DIRECTOR(ES) (Si aplica): María Adelaida Gómez
11. GRUPO O EMPRESA QUE LO AVALA (Si aplica): Centro Internacional de Entrenamiento e Investigaciones Médicas (CIDEIM)
12. OTROS GRUPOS O EMPRESAS:
13. PALABRAS CLAVE (al menos 5): Aprendizaje de Máquina, Selección de Características, Leishmaniasis, Predicción, Tamaño de lesiones, Desenlace terapéutico, procesamiento de imágenes.
14. ODS QUE APLICA EL PROYECTO (Agenda 2030): Salud
15. FECHA DE INICIO (Desarrollo del proyecto): 23/01/2022
16. RESUMEN:

El aprendizaje automático ha aportado avances al campo de la medicina, sin embargo, en muchos casos es difícil implementar esta tecnología debido a la baja cantidad de datos que pueden estar disponibles en los estudios médicos en relación con el número de características que se planean analizar. Este estudio exploro ocho modelos de aprendizaje automático para predecir el desenlace terapéutico de los pacientes con *leishmaniasis cutánea* a partir de las imágenes de las lesiones. Este nuevo enfoque permitirá proponer nuevos mecanismos en el manejo de esta enfermedad a partir de una herramienta para predecir el desenlace terapéutico en tiempo real, además de efectuar recomendaciones en el tratamiento de los pacientes. Finalmente, la contribución de este proyecto servirá de base para las futuras investigaciones que el Centro Internacional de Entrenamiento e Investigaciones Médicas pueda llevar a cabo para encontrar un tratamiento eficaz contra la leishmaniasis.

Modelo predictivo para determinar el desenlace terapéutico del paciente con leishmaniasis a partir de imágenes de lesiones

Jhon Alexander Segura Dorado
Código 8975681

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

Directora
Gloria Inés Álvarez

Codirectora
María Adelaida Gómez

FACULTAD DE INGENIERÍA Y CIENCIAS
MAESTRÍA EN CIENCIA DE DATOS
SANTIAGO DE CALI, FEBRERO 15 DE 2024

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

1. INTRODUCCIÓN

La leishmaniasis es una enfermedad infecciosa causada por parásitos del género Leishmania [1]. Esta infección se transmite al ser humano por la picadura de insectos (vectores) del género *Lutzomyia*, conocidos popularmente en Colombia como arenillas o manta blanca. Esta afecta la piel, las mucosas y las vísceras, además puede causar incapacidad o incluso pérdida de la extremidad afectada [2]. El ciclo de transmisión de la infección implica un reservorio animal (exclusivamente mamíferos) portador del parásito, un vector que se alimenta de este reservorio y un humano afectado por la enfermedad [3].

Esta enfermedad está presente en 98 países y se calcula que en todo el mundo hay unos 12 millones de personas infectadas, con unos 0,9 a 1,6 millones de nuevos casos al año [1]. Dependiendo de la especie de Leishmania infectante, se pueden manifestar tres tipos de enfermedad en humanos: Leishmaniasis visceral (LV), leishmaniasis mucosa (LMC) y leishmaniasis cutánea (LC) [4]. En Colombia, el 99% de los casos reportados son de LC y aunque esta presentación clínica rara vez es mortal, se considera un problema de salud pública en Colombia, debido a la disminución de la calidad de vida de quienes la padecen, los altos costos que deben asumir las entidades de salud y la aparición de nuevos brotes [5]. Según datos del Instituto Nacional de Salud (INS) con corte a la semana epidemiológica 11 de 2022, se notificaron 677 casos de leishmaniasis en todo el país, el 674 (99,6%) corresponden a la forma clínica cutánea y 3 (0,4%) a la mucosa [2].

La población más afectados por LC son, militares que constituyen el 32% de los casos, seguido por los agricultores con el 17.8%. Así mismo ha afectado principalmente a hombres (77%) que habitan en zonas rurales (80%), el grupo de edad más afectado es de 20-29 años (29%) [6]. Las lesiones más frecuentes se encuentran en miembros superiores (46%), miembros inferiores (39%), cara (18%) y tronco (16%). Por otro lado, a nivel departamental, Guaviare, Vaupés, Putumayo y Santander tienen incidencias superiores a 50 casos por cada 100.000 habitantes en riesgo [7].

Por esta razón, se han propuesto diferentes enfoques para este problema como el uso de nuevos fármacos [8], evaluación de los genes que confieren resistencia a los medicamentos [1], análisis de las vitaminas y minerales para mejorar los tratamientos [4], estudio retrospectivo colaborativo para describir la efectividad y seguridad de los tratamientos antileishmaniales en niños ≤ 10 y adultos ≥ 60 años [5], Citocinas y redes de señalización que regulan los resultados de la enfermedad en la leishmaniasis [9]. Por otro lado, se ha utilizado el aprendizaje automático para proponer nuevas metodologías en el manejo de esta enfermedad, como el diagnóstico y

pronóstico de los pacientes con leishmaniasis cutánea antroponótica[1] que no responden al tratamiento mediante redes neuronales artificiales [10], aprendizaje automático para el diagnóstico y pronóstico en dermatología [11], Inteligencia artificial en dermatología [12], Rediseño de fármacos contra la leishmaniasis con redes neuronales de grafos hiperbólicos [13], detección y clasificación automática de imágenes microscópicas de parásitos mediante redes neuronales convolucionales profundas [14], esta metodología ha mostrado buenos resultados en estudios sobre la detención de la leishmaniasis mediante procesamiento de imágenes, optimizando el tiempo para hacer un diagnóstico.

En este trabajo, se llevaron a cabo varios modelos predictivos para ser aplicados en la LC mediante la implementación de aprendizaje de máquina. Se utilizaron diversos enfoques para desarrollar los modelos. En primer lugar, se contó con un modelo propio estructurado a partir de la revisión de la literatura. Luego, se aplicó el método grid search al modelo propio para optimizar sus hiperparámetros. Posteriormente, se exploró el transfer learning, primero con la red VGG16 y parámetros, luego con la VGG19 y ResNet-50, ambos también configurados con parámetros. Adicionalmente, para mejorar aún más la eficacia de los modelos, se aplicó el método grid search a cada una de las redes VGG16, VGG19 y ResNet-50 en modelos separados, en los cuales se buscó la combinación óptima de hiperparámetros para cumplir con el objetivo del estudio. Además, los resultados fueron validados mediante un experto, que en este estudio fue la Dr. María Adelaida Gómez[2].

Este documento está estructurado en 7 capítulos, iniciando con esta introducción, seguido de la definición del problema, luego los objetivos, en cuarto capítulo se presenta el marco conceptual el cual consta de dos secciones que son la parte teórica de la leishmaniasis y la parte conceptual del aprendizaje automático utilizado en esta investigación, el quinto capitulo es la preparación de los datos, el sesto es la construcción de los modelos, séptimo análisis de los resultados Por último, se presentan las conclusiones y trabajos futuros, finalizando con la bibliografía.

[1] Transmisión de humano a humano.

[2] La Dra. María Adelaida Gómez es una integrante del CIDEM que ha apoyado continuamente investigaciones realizadas en la Pontificia Universidad Javeriana Cali relacionadas con el tratamiento contra la leishmaniasis.

1. DEFINICIÓN DEL PROBLEMA

1.1. PLANTEAMIENTO DEL PROBLEMA

Las condiciones climáticas, geográficas y ecológicas hacen que la población colombiana sea vulnerable a las enfermedades transmitidas por picadura, una de las cuales es la LC [15]. Esta es una enfermedad infecciosa causada por parásitos del género Leishmania [16]. La infección se transmite al ser humano por la picadura de insectos del género Lutzomyia, conocidos popularmente en Colombia como arenillas. Esta enfermedad afecta a la piel produciendo laceraciones o grandes úlceras en cualquier parte del cuerpo, que pueden causar discapacidad o incluso la pérdida del miembro afectado [17]. La epidemiología de la leishmaniasis depende de las características del parásito y de las especies de flebótomos (arenillas), de las características ecológicas de los lugares de transmisión, de la exposición previa y actual de la población humana al parásito y del comportamiento humano [16]. En el ciclo de transmisión de la infección implica exclusivamente a un mamífero que porta el parásito, un insecto que porta el paracito y un humano que se ve afectado por la enfermedad (ver Figura 2) [18].

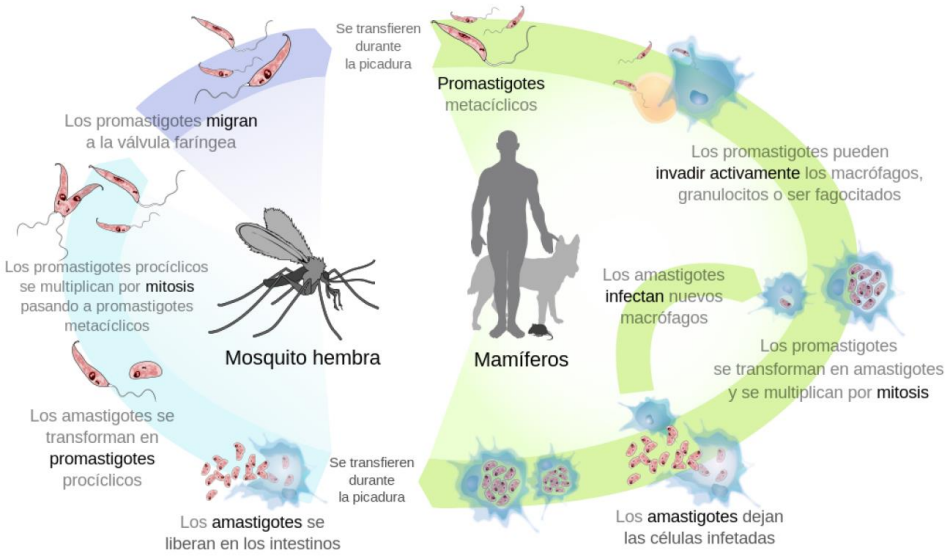


Figura 1. Ciclo de transmisión de la leishmaniasis cutánea
Fuente: Tomado de [18]

La LC es una de las enfermedades infecciosas desatendidas más importantes debido a su fuerte asociación con la pobreza [19]. Según un informe del Instituto Nacional de Salud Colombiano (INS) del año 2022 [20], el número de casos notificados para ese mismo año disminuyó en 72 casos, lo que reflejo una incidencia nacional de 0,61 casos por 100.000 habitantes, frente a la incidencia anterior de 26,2 casos por 100.000 habitantes. Esta brusca diferencia podría deberse a la llegada de la contingencia sanitaria global por COVID-19, que afectó significativamente a la notificación de casos y a su seguimiento. Para el año 2022, el porcentaje total de casos tratados fue del 93,1%. Se observó que el 84.7% de los casos reportados es decir 61 casos corresponden a población rural mientras que el restante 15.3% obedece a población que habita en cabecera municipal. El grupo más afectado se encuentra dentro de los 15 a 44 años, donde se encontró 75% de los casos reportados para el año 2022 [20]. Así mismo, En la Figura 3, se puede observar que el departamento con mayor número de contagios por Leishmaniasis en lo trascurrido del 2022 es Antioquia, con un total de 485 caso reportados, seguido de Santander con 291 casos.

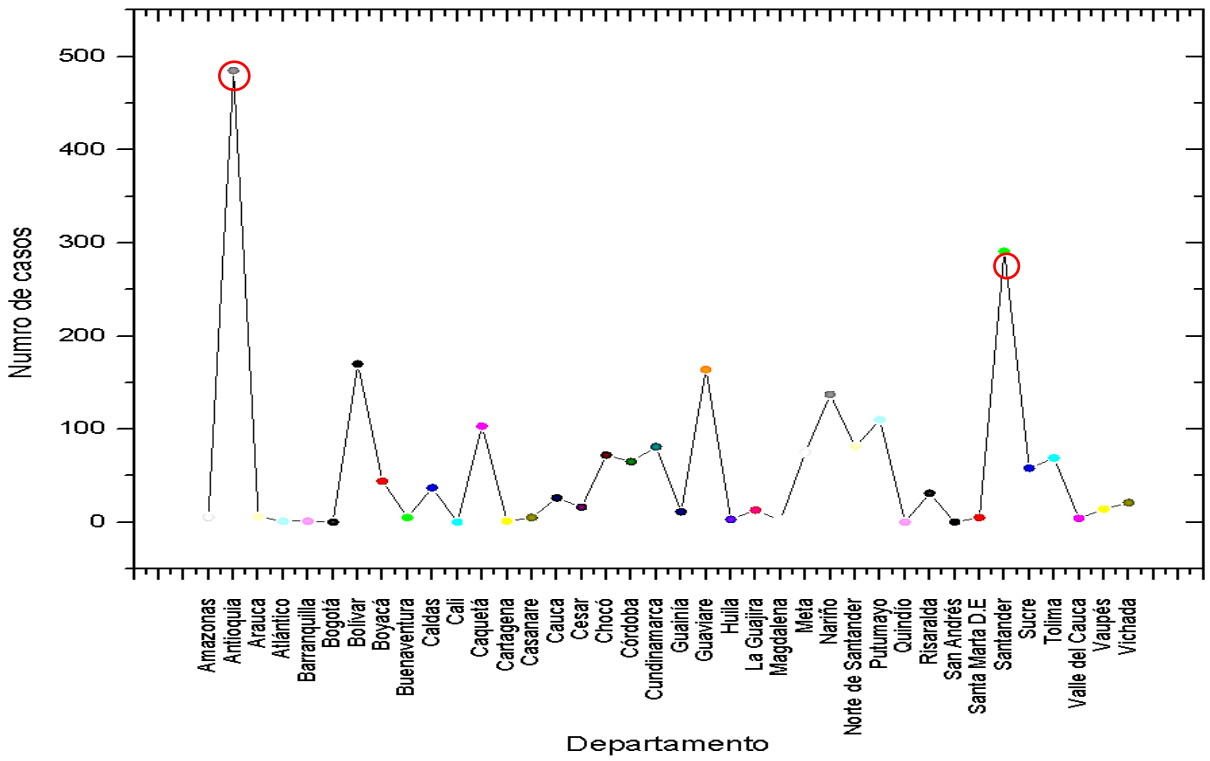


Figura 2. Número de casos reportados por Leishmaniasis en el 2022

Fuente: Autor

Para el tratamiento de la LC se emplea medicamentos el cual consiste en una dosis única diaria de antimonio pentavalente de 20 mg por kilogramo al día durante 20 días [21]. El problema de este tratamiento es su agresividad y sus secuelas, junto con una efectividad de sólo un 70% aproximadamente, lo que genera un balance no muy positivo teniendo en cuenta que un tercio de los pacientes no se curan y acaban más afectados por el mismo tratamiento [21]. Esto produce un gasto de recursos teniendo en cuenta el costo total del tratamiento y una alteración en la salud del paciente que en algunos casos muy específicos los ha llevado al fallecimiento [7]. Un factor que puede contribuir a mejorar el desenlace terapéutico de los pacientes con leishmaniasis es identificar la relación que puede existir entre las características de las lesiones, permitiendo predecir el desenlace terapéutico del paciente. Esta relación se puede hacerse porque, al ser visibles, accesibles y generalmente bien delimitadas, las lesiones son fáciles de comparar, por ello, al inicio del tratamiento deben medirse y registrarse debidamente en la historia de cada paciente. Esto proporcionará pruebas cuantificables para determinar si hay una respuesta en la terapia del paciente [22]. Además, se evitará que los pacientes que no han tenido una respuesta positiva sigan sometiéndose a un tratamiento que sólo les dejaría efectos secundarios, secuelas y los riesgos que conlleva, además de la pérdida de recursos utilizados en el tratamiento.

Por lo tanto, este proyecto propone realizar una predicción del desenlace terapéutico utilizando técnicas de aprendizaje automático, debido al impacto que ha tenido en estudios clínicos, como la investigación realizada por [15], este propuso un modelo predictivo para la ocurrencia de leishmaniasis en Colombia, a partir de variables ambientales y socioeconómicas, los datos demuestran que el análisis de componentes principales con algoritmos de aprendizaje automático logró una precisión global del 75% en el diagnóstico. Así mismo [10] plantearon un enfoque novedoso de diagnóstico y pronóstico para pacientes con leishmaniasis cutánea antroponótica utilizando redes neuronales artificiales, como resultado se demostró que el modelo presentó multicapa podría utilizarse para la detección rápida, el pronóstico preciso y el tratamiento eficaz de pacientes con una precisión del modelo en 80%.

Por esta razón, se desarrolló un modelo basado en redes neuronales convolucionales para determinar el desenlace terapéutico final pacientes, para esto se utilizó una base de datos de imágenes del Centro Internacional de Entrenamiento e Investigaciones Médicas (CIDEIM), donde se encuentras fotografías de la evolución del paciente sometido al tratamiento contra la leishmaniasis, estas están separadas por semanas desde la semana 0 que entra el paciente con las lesiones, hasta la semana 13 donde el paciente está completamente curado, es decir sin presencia de lesiones. Este modelo propuesto podría ser una herramienta valiosa para mejorar la prevención, diagnóstico y tratamiento de la enfermedad en el país, debido a que el modelo podrá

realizar un procesamiento de un gran volumen de imágenes en tiempo real, permitiendo obtener un diagnóstico en un lenguaje que el tomador de decisiones pueda interpretar y ser asertivo en el tratamiento del paciente. Además, puede contribuir a la formulación de políticas publicas para reducir el impacto negativo que tiene esta enfermedad.

1.2. FORMULACIÓN DEL PROBLEMA

En el campo de la salud, la implementación de tecnologías innovadoras emerge como una necesidad apremiante destinada a potenciar la precisión y eficacia de los tratamientos médicos. La leishmaniasis, una enfermedad parasitaria caracterizada por manifestaciones cutáneas distintivas, presenta desafíos específicos en relación con el diagnóstico y la elaboración de estrategias terapéuticas. En vista de la imperante demanda de atención personalizada, surge una interrogante central.

1.2.1. Pregunta de investigación

¿Cómo implementar un modelo predictivo con técnicas de aprendizaje automático para determinar el desenlace terapéutico de los pacientes con leishmaniasis utilizando las imágenes de las lesiones?

1.2.2. Preguntas de sistematización

¿Cómo realizar el procesamiento de imágenes de las lesiones para que sean variables de entrada a los modelos de aprendizaje automático?

¿Cuáles modelos se pueden usar en esta investigación?

¿Cómo entrenar los modelos seleccionados?

¿Como evaluar los resultados de los modelos?

¿Cuál es el mejor modelo para predecir el resultado terapéutico con un índice de asertividad significativo?

2. OBJETIVOS DEL PROYECTO

2.1. OBJETIVO GENERAL

Desarrollar un modelo predictivo con técnicas de aprendizaje automático para determinar el desenlace terapéutico de los pacientes con leishmaniasis utilizando las imágenes de las lesiones

2.2. OBJETIVOS ESPECÍFICOS

- Construir un conjunto de datos a partir de las imágenes de las lesiones y el resultado terapéuticos de los pacientes.
- Seleccionar los modelos para predecir el desenlace terapéutico de los pacientes teniendo en cuenta las imágenes de las lesiones.
- Entrenar los modelos a partir de la base de datos procesada.
- Evaluar los modelos con métricas propias del aprendizaje automático.
- Proponer un modelo computacional que permita predecir el desenlace terapéutico

3. MARCO TEÓRICO Y ANTECEDENTES

3.1. MARCO TEÓRICO

En el presente capítulo está dividido en dos secciones. En la primera parte se presenta los conceptos teóricos de la leishmaniasis, seguido de una caracterización de las redes neuronales convolucionales, Finalmente se presentan los estudios previos con relación a esta investigación.

3.1.1. Leishmaniasis

La leishmaniasis en el ser humano puede ser causada por unas 20 especies de parásitos pertenecientes al género Leishmania, un complejo ciclo de vida el cual involucra a múltiples vectores artrópodos y especies de mamíferos que actúan como reservorios, que requieren sangre de mamíferos para completar su ciclo reproductivo [23]. Diez de estas especies están presentes en Colombia, donde la enfermedad se transmite por la picadura de insectos hembra del género Lutzomyia. Los principales factores de riesgo para adquirir la enfermedad son: la pobreza, el déficit nutricional, el desplazamiento de la población, el déficit de vivienda, la debilidad del sistema inmunitario y la falta de recursos [18]. Existen tres tipos de leishmaniasis que se describen a continuación.

Leishmaniasis cutánea: Tiene diferentes formas clínicas y variedad de lesiones cerradas como pápulas, nódulos y placas con aspecto verrugoso, hasta las formas en úlceras. La enfermedad tiende a ser dolorosa cuando hay sobreinfección bacteriana (ver Figura 3). La LC puede tornarse crónica luego de doce o más semanas sin cierre de la úlcera o transformaciones de esta en placas con costras [21].

Figura 3. Leishmaniasis cutánea
Fuente: Tomado de [24]

Leishmaniasis mucocutánea (mucuosa): Ocurre como resultado de la diseminación linfohematógena del parásito. Afecta las mucosas de las vías aéreas superiores, nariz, faringe, laringe, boca y tráquea. Entre el 3% y 5% de los pacientes con LC, suelen presentar también la mucocutánea. Se producen lesiones en las zonas afectadas, congestión y obstrucción. Además, de producir graves malformaciones por falta de atención oportuna (ver Figura 4) [21].

Figura 4. Leishmaniasis mucocutánea
Fuente: Tomado de [20]

Leishmaniasis visceral: Es una enfermedad del sistema retículo endotelial. Se caracteriza por fiebre, anemia, leucopenia, trombocitopenia y debilidad progresiva. También se pueden presentar diarreas e infecciones respiratorias (ver Figura 5) [21].

Figura 5. Leishmaniasis visceral
Fuente: Tomado de [21]

La leishmaniasis cutánea (LC) es la forma más común de leishmaniasis y se caracteriza por la aparición de úlceras en zonas expuestas del cuerpo o también puede producirse en otras zonas por diseminación hematógena o linfática del parásito. Se calcula que cada año se producen en el mundo entre 600.000 y 1 millón de nuevos casos de leishmaniasis cutánea [25]. En 2017, el 95% de las notificaciones de leishmaniasis cutánea procedieron de seis países: Afganistán, Argelia, Brasil, Colombia, República Islámica de Irán y República Árabe Siria [1].

En Colombia la LC es la más representativa, comprendiendo el 98% a 99% de los casos, caracterizada por lesiones cutáneas de lenta evolución, generalmente causadas por la migración del parásito por vía hematógena o linfática hacia las mucosas nasales, visceral, que es la menos frecuente, pero es la forma clínica que por sí misma puede ser causa directa de muerte, caracterizada por la invasión sistémica del parásito y el compromiso hepático, esplénico y de médula ósea. En Colombia, la población que predominantemente se ve afectada por esta forma clínica es la conformada por los menores de 5 años [20].

La incidencia nacional para la semana epidemiológica del 19 al 25 de junio de 2022 reporto que se presentan 18,6 casos por cada 100.000 habitantes en riesgo. En el análisis departamental, se observa que Guaviare, Vaupés, Putumayo y Santander tienen incidencias superiores a los 50 casos por cada 100.000 habitantes en riesgo [20] (ver Figura 6).

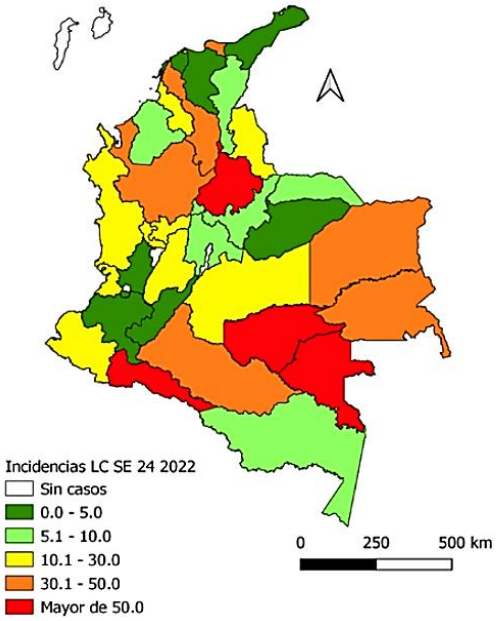


Figura 6. Incidencias leishmaniasis cutánea del 19 al 25 de junio de 2022
Fuente: Tomado de [20]

La LC ha afectado principalmente a hombres (77 %), a personas procedentes de área rural (80 %), el grupo más afectado es el de 20 a 29 años (29 %) y el régimen de afiliación al SGSSS más frecuente es el subsidiado (54 %). Las lesiones se ubican en miembros superiores (46 %), inferiores (39 %), cara (18 %) y tronco (16 %). Se reportó que el 94 % fueron tratados [20].

3.1.2. Ciclo de vida de la infección natural por Leishmaniasis

El ciclo biológico de transmisión de la leishmaniasis se puede dividir en 8 etapas (ver Figura 7), es importante saber que este ciclo es digénico, es decir, que una parte del ciclo se desarrolla en el hospedador, por ejemplo, perros o conejos, y otra parte en insectos del género Phlebotomus (moscas de arena). Los parásitos de Leishmania están dentro del tubo digestivo del insecto, el cual se infecta al picar a un mamífero infectado, donde luego se transforman dentro del sistema digestivo del insecto y al picar a otro animal o ser humano le transmite la enfermedad, este ciclo puede durar de 4 a 20 días [26].

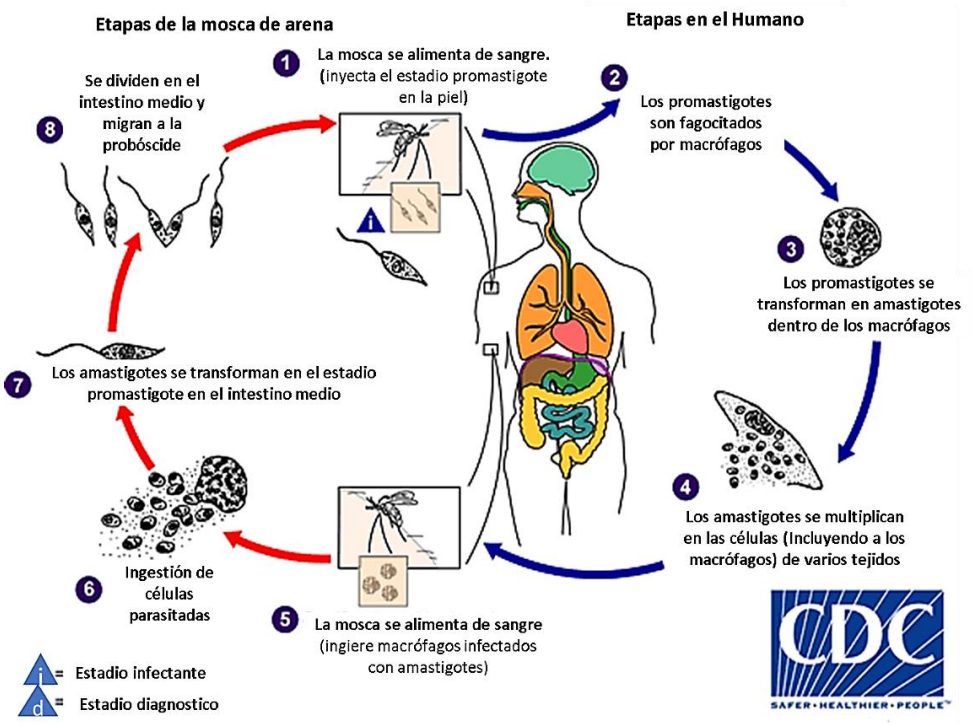


Figura 7. Ciclo biológico de la leishmaniasis
Fuente: Tomado de [26]

La mosca se alimenta de un humano e inyecta el promastigote (Fase 1), los promastigotes en el punto de la infección por el insecto son fagocitados por los macrófagos (Fase 2), los promastigotes se transforman en amastigotes (Fase 3), estos amastigotes se multiplican y afectan diferentes tejidos causando manifestaciones clínicas de la leishmaniasis (Fase 4), los mosquitos pican animales infectados y adquieren el parásito (Fase 5 y Fase 6), en el aparato digestivo del mosquito, el parásito se desarrolla en promastigotes (Fase 7), para completar el ciclo, los promastigotes se multiplican se desarrollan y se trasladan a las partes bucales del vector, donde al picar a otra persona se contagia la enfermedad (Fase 8).

3.1.3. Aprendizaje de máquina

Esta sección del presente capítulo está dedicada a presentar los conceptos y terminología más importante que se utilizó para el desarrollo de la investigación en términos de aprendizaje de maquina y redes neuronales convolucionales.

Preprocesamiento de imágenes: En esta etapa se lleva el dataset a un estado apropiado para un procesamiento posterior. Entre las actividades que aportan a este objetivo se encuentran [27]:
Adquisición datos: Es el proceso de reunir todos los datos necesarios para el aprendizaje automático. Dicho proceso puede ser tedioso, ya que los datos residen en muchas fuentes, como ordenadores portátiles, la nube, aplicaciones y dispositivos. Además, los datos pueden tener formatos y tipos diferentes según la fuente. Por ejemplo, no es fácil utilizar conjuntamente datos de vídeos y datos tabulares. por lo tanto, las imágenes en el estudio deberán hacer adecuadas a un tamaño correcto.

Etiquetado de datos: El etiquetado de datos es el proceso de identificación de los datos en bruto (imágenes, archivos de texto, vídeos, etc.) y la adición de una o más etiquetas significativas e informativas para proporcionar contexto, de modo que un modelo de aprendizaje automático pueda aprender de ellos. Por ejemplo, las etiquetas pueden indicar si el tratamiento que se le suministro al paciente contra la Leishmaniasis fallo o curo. El etiquetado de datos es necesario para diversos casos de uso, como la visión por ordenador, el procesamiento del lenguaje natural y el reconocimiento del habla.

Limpieza de Datos: Completado de los valores ausentes, suavizado de datos ruidosos, identificación o eliminación de valores atípicos y resolución de inconsistencias.

Transformación de Datos: La transformación de datos es el proceso de convertir el formato, el valor o la estructura de los datos en otra forma. Esto implica añadir, replicar y eliminar entradas, así como normalizar su estética.

Reducción de Datos: Reducción del volumen de datos, pero produciendo resultados analíticos iguales o similares.

Discretización de datos: Reemplazado de atributos numéricos por nominales.

Publicación: Una vez finalizados los procesos de organización, limpieza y mejora, sólo queda publicar el conjunto de datos final.

3.1.4. Tipos de Aprendizaje automático

El aprendizaje automático (AM) es el proceso mediante el cual se utilizan modelos matemáticos de datos para ayudar a un ordenador a aprender sin instrucciones directas, este se considera un subconjunto de la inteligencia artificial (IA). Actualmente se utilizan dos tipos principales de algoritmos de aprendizaje automático: el aprendizaje supervisado y el aprendizaje no supervisado [28]. La diferencia entre ellos viene definida por la forma en que cada uno aprende sobre los datos para hacer predicciones:

Aprendizaje automático supervisado: Los algoritmos supervisados de aprendizaje automático son los más utilizados. Con este modelo, un científico de datos actúa como guía y enseña al algoritmo las conclusiones que debe sacar. En el aprendizaje supervisado, el algoritmo se entrena mediante un conjunto de datos que ya está etiquetado y tiene un resultado predefinido. Este aprendizaje es el que se va a utilizar en este trabajo de investigación.

Aprendizaje automático no supervisado: El aprendizaje automático no supervisado utiliza un enfoque más independiente, en el que un ordenador aprende a identificar procesos y patrones complejos sin la orientación estrecha y constante de un ser humano. El aprendizaje automático no supervisado implica un entrenamiento basado en datos que no tienen ni etiquetas ni un resultado específico definido.

3.1.5. Algoritmos de aprendizaje automático

Inteligencia artificial (IA): Es el estudio del diseño de agentes inteligentes que son capaces de tomar la decisión de qué acciones tomar y cuándo tomar estas acciones [29].

Support Vector Classifier (SVC): Support Vector Classifier, es un algoritmo de clasificación que separa las clases a predecir por medio de un hiperplano que intenta atravesar un espacio *n* dimensional producido por *n* características, donde a un lado del hiperplano se encuentran todos los casos positivos, y al otro, todos los casos negativos [27].

Clustering: Es una técnica de aprendizaje no supervisado [29], que consiste en tomar un conjunto de datos y generar diferentes grupos a partir de características que compartan los elementos de dicho conjunto.

Tuning: Conocido también como afinamiento u optimización de hiperparámetros, consiste en buscar objetivamente diferentes valores para los hiperparámetros de un modelo y elegir un subconjunto que dé como resultado un modelo con el mejor rendimiento para el conjunto de datos suministrado [27].

Redes neuronales: Primero es importante mencionar que una neurona es una célula del cerebro cuya función principal es la recolección, procesamiento y emisión de señales eléctricas [30]. La idea es que las redes neuronales artificiales simulen el comportamiento de las neuronas biológicas, es así como las redes neuronales artificiales, están compuestas por unidades a través de conexiones dirigidas. Una conexión de una unidad con otra sirve para propagar la activación, además, cada conexión tiene un peso numérico asociado que determina su fuerza y el signo de conexión. Cada unidad primero calcula una suma ponderada de sus entradas. Luego se aplica una función de activación a la suma para producir la salida [30], la Figura 8 muestra esto en detalle.

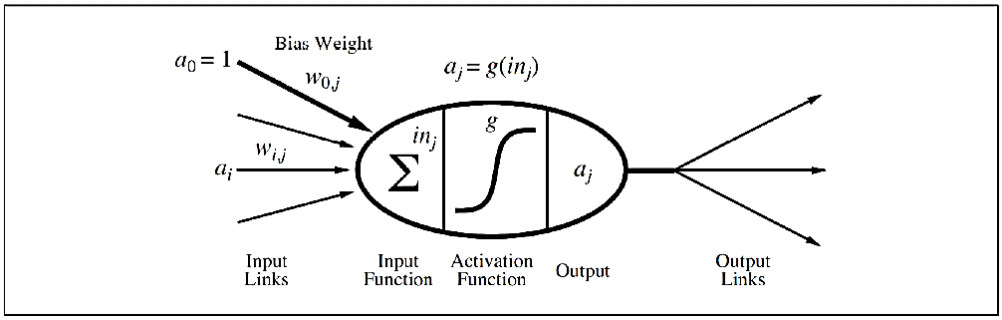


Figura 8. Modelo matemático de una neurona
Fuente: Tomado de [27]

3.1.6. Arquitectura de una red neuronal simple

Una red neuronal básica tiene neuronas artificiales interconectadas en tres capas [30]:

Capa de entrada: La información del mundo exterior entra en la red neuronal artificial desde la capa de entrada. Los nodos de entrada procesan los datos, los analizan o los clasifican y los pasan a la siguiente capa.

Capa oculta: Las capas ocultas toman su entrada de la capa de entrada o de otras capas ocultas. Las redes neuronales artificiales pueden tener una gran cantidad de capas ocultas. Cada capa oculta analiza la salida de la capa anterior, la procesa aún más y la pasa a la siguiente capa.

Capa de salida: La capa de salida proporciona el resultado final de todo el procesamiento de datos que realiza la red neuronal artificial. Puede tener uno o varios nodos. Por ejemplo, si tenemos un problema de clasificación binaria (sí/no), la capa de salida tendrá un nodo de salida que dará como resultado 1 o 0. Sin embargo, si tenemos un problema de clasificación multiclase, la capa de salida puede estar formada por más de un nodo de salida.

3.1.7. Arquitectura de una red neuronal profunda

Las redes neuronales profundas, o redes de aprendizaje profundo, tienen varias capas ocultas con millones de neuronas artificiales conectadas entre sí. Un número, denominado peso, representa las conexiones entre un nodo y otro. El peso es un número positivo si un nodo estimula a otro, o negativo si un nodo suprime a otro. Los nodos con valores de peso más altos tienen mayor influencia en los demás nodos [28].

En teoría, las redes neuronales profundas pueden asignar cualquier tipo de entrada a cualquier tipo de salida (ver Figura 9). Sin embargo, también necesitan mucho más entrenamiento en comparación con otros métodos de aprendizaje de máquina. Necesitan millones de ejemplos de datos de entrenamiento en lugar de los cientos o miles que podría necesitar una red más simple [28].

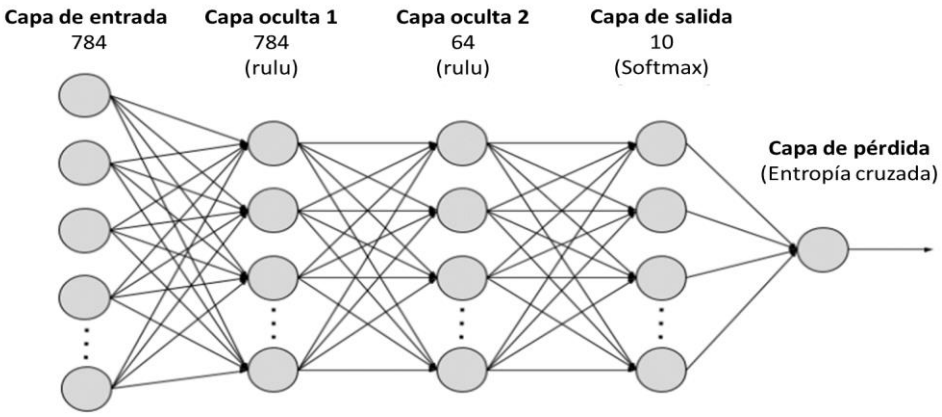


Figura 9. Red neuronal profunda
Fuente: Tomado de [27]

3.1.8. Estructura de una red neuronal convolucionales

Las redes neuronales convolucionales (CNN) son, en términos generales, modelos de redes neuronales artificiales multicapa. De acuerdo con la estructura de la corteza visual animal descrita por Hubel y Weisel, el modelo puede interpretarse visualmente como se muestra en la Figura 10.

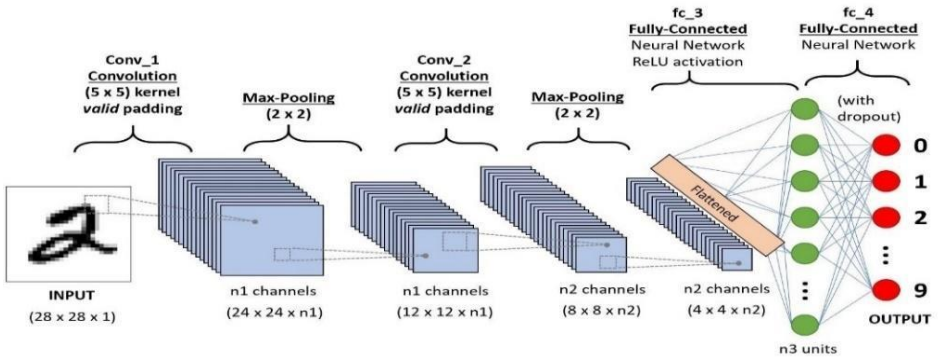


Figura 10. Visualización de una CNN
Fuente: Tomado de [28]

Se puede observar que cada uno de los bloques representa una capa diferente de la red de neuronas convolucional, los cuales tienen altura, anchura y profundidad. De izquierda a derecha podemos observar, las capas de entrada, las capas ocultas y las capas completamente conectadas. Después de la última capa, el modelo genera una clasificación [31].

Las capas completamente conectadas imponen la conectividad local entre las neuronas y las capas adyacentes [32]. Como tal, las entradas de las capas ocultas son un subconjunto de neuronas de la capa que precede a dicha capa oculta. Esto asegura que las neuronas del subconjunto de aprendizaje produzcan la mejor respuesta posible. Además, las unidades comparten el mismo peso y sesgo en el mapa de activación, de modo que todas las neuronas que estén en una capa dada están analizando la misma característica [33].

3.1.9. Curva ROC

La clasificación binaria en problemas de clasificación se fundamenta en la elección entre dos opciones posibles: 0 y 1, "sí" o "no", o "positivo" y "negativo". Este concepto desempeña un papel esencial en la toma de decisiones dentro de los modelos de Machine Learning, la Ciencia de Datos y la Econometría [34]. El concepto utilizado para evaluar la calidad de la clasificación binaria y las predicciones se conoce como Curva Característica Operativa del Receptor, o ROC (Receiver Operating Characteristic) curve en inglés [35].

Los problemas de clasificación de dos clases involucran la asignación de cada instancia (i) a uno de los dos elementos del conjunto {p, n}, que representan las etiquetas de clase positiva (p) y negativa (n). Dependiendo del modelo de clasificación, se pueden obtener dos escenarios posibles. En el primero, se calcula una estimación de la probabilidad de pertenencia de las instancias a una clase, y se pueden aplicar diferentes umbrales para predecir la probabilidad de pertenencia a una clase específica. En el segundo escenario, se asigna una etiqueta discreta que indica la clase predicha para cada instancia [36].

Para diferenciar entre la clase real y la pronosticada, se usan las etiquetas {Y, N}, para las predicciones de clases dadas por el modelo. Para cada instancia (i) y dado un clasificador, hay cuatro posibles resultados:

- Si la instancia es positiva y se clasifica como positiva, se cuenta como un verdadero positivo (true positive).

- Si la instancia es positiva y se clasifica como negativa, se cuenta como un falso negativo (false negative).
- Si la instancia es negativa y se clasifica como negativa, se cuenta como un verdadero negativo (true negative).
- Si la instancia es negativa y se clasifica como positiva, se cuenta como un falso positivo (false positive).

Para un conjunto de instancias (conjunto de prueba) y un clasificador dado, se genera una matriz de confusión de 2*2, llamada también tabla de contingencia, que representa las disposiciones del conjunto de instancias. La Figura 11 es un ejemplo de una matriz de confusión y diferentes ecuaciones para diferentes métricas que se pueden calcular a partir de esta. Los números de la diagonal {p, Y} y {n, N} representan las decisiones correctas del modelo y la otra diagonal son los errores, es decir, la confusión entre las clases [37].

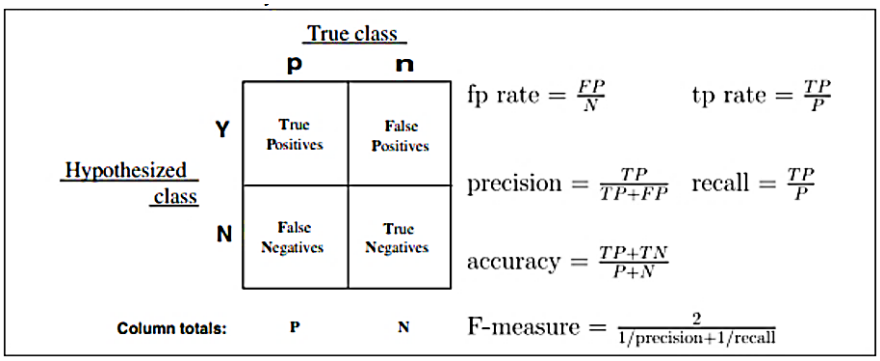


Figura 11. Matriz de confusión y métricas de rendimiento
Fuente: Tomado de [37]

3.1.10. Aumento de datos

En las últimas dos décadas se ha suscitado un gran interés por el aprendizaje profundo (Deep Learning) lo que implica que las CNN se hayan convertido en la herramienta más usada para análisis y clasificación de imágenes. Sin embargo, existen desafíos con este tipo de modelos. Por un lado, está el hecho que son impulsadas por redes muy grandes que implican el entrenamiento de miles de parámetros, y por otra parte está la falta de conjuntos de datos de entrenamiento

confiables, lo que origina problemas de sobreajuste y poca capacidad de generalización. Por lo general, el problema más común es la falta de datos, que en algunos casos pueden ser difíciles de obtener [38].

Una alternativa a este problema es aumentar los datos (data augmentation) realizando transformaciones afines y elásticas tradicionales: crear nuevas imágenes mediante la rotación o el reflejo de la imagen original, acercar y alejar, desplazar, aplicar distorsión y cambiar la paleta de colores, esto hace que el modelo sea más robusto y evita que la red neuronal aprenda patrones irrelevantes, permitiendo un aumento del rendimiento en general.
Para cada técnica de transformación se debe especificar el factor por el cual se aumentaría el tamaño de su conjunto de datos (factor de aumento de datos), suponiendo que no se necesita conocer qué hay más allá de la imagen. Las técnicas más utilizadas de aumento son:

- Voltear (Flip): este parámetro permite voltear la imagen horizontal o verticalmente.

- Rotación (Rotation): permite girar la imagen en un rango de 0 a 180 grados. Esto puede hacer que las dimensiones de la imagen no se conserven después de la rotación.

- Escala (Escale): esta opción permite escalar hacia afuera o hacia adentro. Si se escala hacia afuera, el tamaño final de la imagen será mayor que el tamaño original, caso contrario, si se escala hacia adentro, la nueva imagen se recortará en determinadas secciones, pero con un tamaño igual que la original.

- Recortar (Crop): a diferencia del escalado, parte de la imagen original se recorta y se crea una nueva imagen del tamaño de la imagen original. Este método se conoce como recorte aleatorio (random cropping).

- Translación (Translation): implica mover la imagen a lo largo de las coordenadas X o Y o ambas. Esto permite a la red convolucional buscar casi que en cualquier parte de la imagen un objeto.

- Ruido Gaussiano (Gaussian Noise): su objetivo es distorsionar las características de alta frecuencia, cuyos patrones se repiten en relativamente un número grande de imágenes del conjunto de datos y pueden no ser útiles. El ruido gaussiano, tiene media cero y puede

distorsionar características de alta frecuencia y baja frecuencia, permitiendo a la red mejorar su capacidad de aprendizaje.

3.2. ANTECEDENTES

Existen varios estudios previos relacionados con la LC, estos estudios se pueden dividir en dos grupos. Los que se centran en el procesamiento de imágenes de la leishmaniasis y los estudios que se centran en la generación de métodos predictivos para el resultado terapéutico. Estos últimos son los que más interesan en el presente trabajo.

Para construir la revisión de literatura se consultaron las bases de datos Web of Science, SpringerLink, IEEE Xplore, PubMed, Science Direct, Scientific Electronic Library Online (SciELO), SPIE Digital Library y Wiley Online Library con una estrategia de búsqueda diseñada para obtener resultados relacionados para esta sesión. Para esto se utilizaron diferentes términos de búsqueda: "procesamiento de imágenes", "análisis de imágenes", "imágenes de diagnóstico médico", "leishmaniasis cutánea", "lesión cutánea", "herida cutánea", “clasificación”, “segmentación de heridas”, “predicción”, “medición del área de la herida”, “delineación de los límites de la herida”, “modelado 2D”, “Aprendizaje automático”, “aprendizaje profundo”, “redes neuronales”, “redes neuronales convolucionales”, entre otros.

3.2.1. Aprendizaje automático aplicado al procesamiento de imágenes con LC

N. Steyve y E. Pierre [39] presentaron un enfoque de diagnóstico optimizado para las enfermedades tropicales desatendidas de la piel mediante la identificación automática de lesiones cutáneas en su fase inicial. Las características detectadas se introducen en una máquina de vectores de soporte (SVM) optimizada por un algoritmo de agujero negro (BHO) para clasificarse en tiempo real. El modelo logra una precisión de clasificación global del 96 %, una especificidad del 94 %, un F-SCORE del 89 % y una sensibilidad del 92 % de las imágenes en diferentes condiciones. Estos resultados mejoran la situación sanitaria de la población y posibilitar el desarrollo de un Diagnóstico Asistido por Computador (CAD) eficiente. M. Górriz y D. López-Codina [40] presentan un estudio que utiliza el procesamiento de imágenes para automatizar el proceso de detección basado en un enfoque de aprendizaje profundo. Entrenaron un modelo U-net que segmenta con éxito los parásitos de leismania y los clasifica en promastigotes, amastigotes y parásitos adheridos. M. Zare [41] Presento un estudio que tuvo como objetivo

desarrollar un algoritmo basado en inteligencia artificial para el diagnóstico automático de la leishmaniasis. Se utilizamos el algoritmo de Viola-Jones para desarrollar un sistema de detección de parásitos de LC. El algoritmo incluye tres procedimientos: extracción de características, creación de imágenes integrales y clasificación. Como resultado el modelo obtuvo una precisión del 50% en la detección de macrófagos infectados con el parásito leishmania. Y. Wu, Y. Yang y M. Saitoh [42] Presentan el diseño de una red neuronal artificial feed-forward para la clasificación de imágenes hiperespectrales de úlceras cutáneas en 4 tipos de causas: vasculopatía oclusiva, venosa, leishmaniasis y diabética. Como resultado se obtiene una estructura de red neuronal que logra un porcentaje de acierto superior al 72% en la clasificación de datos. J. Galeano y S. M. Robledo [43] Proponemos utilizar un modelo de reflectancia difusa de tres capas para analizar la leishmaniasis en dos pacientes. Se tomaron imágenes multiespectrales y luego se procesaron utilizando un modelo matemático directo y un enfoque de optimización. Este proceso nos proporcionó valores de variables ópticas que están relacionadas con varios parámetros biológicos, como la fracción volumétrica de melanina, hemoglobina, saturación de oxígeno, diámetro de queratinocitos, colágeno, fibroblastos y macrófagos, así como la fracción volumétrica de colágeno.

3.2.2. Aprendizaje automático aplicado a la predicción frente a la LC

R. J. King y C. R. Davies [44] desarrollaron un modelo predictivo de la ocurrencia de LC dividiendo el territorio nacional de Colombia en 5 zonas biogeográficas para el análisis epidemiológico. Dichos autores crearon un modelo de regresión logística a partir de la asociación entre la altura del municipio, la cobertura del suelo (25 variables categóricas) y la probabilidad de que se haya reportado al menos un caso de la enfermedad. Trabajaron en un análisis de los factores ambientales de riesgo para la LC. Por otro lado, C. Valderrama-Ardila [45] realizaron un análisis que se centró en el brote de 2003 a 2007 en Colombia. En dicha investigación utilizaron un modelo condicional autorregresivo de Poisson para la correlación espacial, las variables predictoras fueron, el uso de la tierra, la elevación y variables climáticas como la temperatura media y la precipitación. La variable respuesta fue el total de casos de LC en los 5 años de estudio, con una función de vinculo logarítmica y el logaritmo de la población como compensación. Los resultados de este estudio mostraron que la mayor incidencia de LC se presentó en temperaturas promedio de 26°C. Este estudió mediante sus hallazgos confirmó el papel del clima y el uso de la tierra en la transmisión de LC. M. Pérez-Flórez y N. Alexander [46] identificaron los factores de riesgo ambientales para la LC en Colombia (región andina 715 municipios rurales y urbanos) utilizando 10 años de vigilancia (2000 - 2009) mediante análisis espaciotemporales en modelos de efectos aleatorios autorregresivos condicional de Poisson, para modelar la dependencia de la incidencia

de LC en el uso de la tierra, el clima, la elevación y la densidad de población. Los resultados de esta investigación identificaron que las selvas tropicales, los bosques, la vegetación secundaria, la temperatura y la precipitación anual están asociadas positivamente con la incidencia de LC. Además, que en general el clima y el uso de la tierra se pueden utilizar para identificar áreas de alto riesgo de LC.

El uso de técnicas de aprendizaje automático para realizar predicciones teniendo en cuenta el desenlace terapéutico de la LC ha sido muy escaso. Sin embargo, se han implementado técnicas de aprendizaje profundo (deep learning) para predecir perfiles epidemiológicos a partir de series de tiempo, haciendo uso de redes neuronales recurrentes (RNN) y redes neuronales convolucionales (CNN). La comparación de los resultados de dichas técnicas con respecto a modelos autorregresivos de series de tiempo mostró un mejor desempeño [25]. Y. Wu, Y. Yang, H. Nishiura, and M. Saitoh [31] desarrollarón un marco de aprendizaje profundo, por primera vez, para predecir perfiles epidemiológicos en la perspectiva de series temporales. Adoptando redes neuronales recurrentes (RNN) para capturar la correlación a largo plazo de los datos y redes neuronales convolucionales (CNN) para fusionar información de datos de diferentes fuentes. Por otro lado, aplicaron una estructura residual para evitar problemas de sobreajuste en el proceso de formación. N. Shabanpour, S. V. Razavi-Termeh, y T. Abuhmed [47] Realizaron una predicción espacial de la leishmaniasis cutánea utilizando tres algoritmos de aprendizaje automático (árbol de decisión (DT), regresión de vector de soporte (SVR) y regresión lineal (LR)). La base de datos espacial se creó utilizando datos recopilados sobre el número de enfermedades en la provincia de Isfahan de 2011 a 2018, así como diez parámetros ambientales (temperatura, humedad, lluvia, altitud, pendiente, velocidad del viento, índice de vegetación de diferencia normalizada, número de días soleados, número de días con heladas y distancia al arroyo) que afectan la incidencia de leishmaniasis. Además, en este estudio se empleó el método difuso para reducir la incertidumbre y evaluar el efecto de los factores ambientales en la prevalencia de la enfermedad.

Por lo anterior, en la literatura no se encontró trabajos que implementaran técnicas de aprendizaje automático para determinar el desenlace terapéutico de los pacientes con LC utilizando las imágenes de las lesiones, Por lo tanto, este nuevo enfoque será un aporte significativo a la literatura y a la investigación frente a la LC, contribuyendo en el mejoramiento del tratamiento para los pacientes.

4. PREPARACIÓN DE LOS DATOS

En este capítulo, se presenta de manera detallada el proceso de recopilación y construcción del conjunto de datos crucial para la investigación. Además, se explican las diversas técnicas y enfoques implementados con el objetivo de optimizar los resultados obtenidos.

4.1. Construcción del conjunto de datos a partir de las imágenes de las lesiones y el resultado terapéuticos de los pacientes

4.1.1. Entendimiento de los datos

Para esta investigación, se utilizaron datos otorgados por el CIDEIM de la ciudad de Cali. Se dispuso de un archivo que comprendía 323 imágenes de pacientes con LC provenientes de diversas regiones de Colombia. Estas imágenes se dividieron como 158 clasificadas como cura y 165 clasificadas como falla, permitiendo así un análisis exhaustivo de ambos grupos. Las imágenes fueron capturadas durante un número específico de días, basado en la metodología propuesta por P. Olliaro [48] para simplificar los procedimientos (ver Figura 12). Siguiendo esta metodología, el CIDEIM llevó a cabo la evaluación de los resultados del tratamiento en tres instancias, basándose en observaciones visuales y en la experiencia médica. Este enfoque les posibilitó determinar si el resultado del tratamiento del paciente podía clasificarse como un éxito o un fallo (ver Tabla 1):

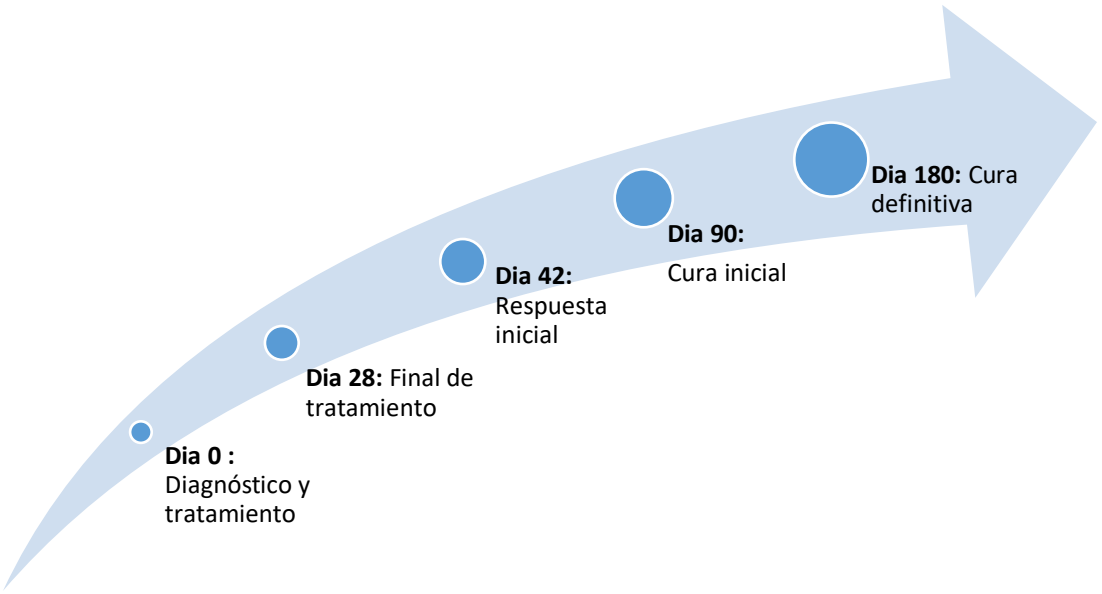


Figura 12. Representación esquemática del tratamiento
Fuente: Tomado de [48]

Tabla 1. Días para evaluar el tratamiento

Días	Descripción
Día 42 o 63 (6 o 9 semanas)	Permite observar una respuesta inicial, identificando fallas tempranas.
Día 90 (3 meses)	Cura inicial (Se puede concluir si el tratamiento falla)
Día 180 o 360 (6 o 12 meses)	Cura definitiva (Se puede concluir si el tratamiento falla)

Fuente: Elaboración propia

Además, se dispuso de un archivo de Excel que contenía variables relevantes para el entendimiento de las imágenes, como el tipo de medicamento suministrado, el número de heridas, la semana en que se tomó la foto, el número de historia clínica, el desenlace terapéutico y las observaciones asociadas a cada paciente.

Las imágenes utilizadas fueron cargadas en el entorno de Google Colab utilizando un script de Python. Posteriormente, se realizaron tareas de almacenamiento, categorización y división en conjuntos de entrenamiento, validación y prueba para el preprocesamiento de los datos.

4.1.2. Preparación de las imágenes

El total de imágenes tiene un peso de almacenamiento de 678 MB, categorizadas en 158 clasificadas como cura (48.9 %) y 165 clasificadas como falla (51.1 %). Cada una de estas imágenes ha sido minuciosamente examinada para evaluar su calidad y detectar cualquier elemento inusual en su composición. Este proceso meticuloso tiene como objetivo elevar la calidad del entrenamiento de nuestra red neuronal convolucional (CNN), con la mira puesta en obtener resultados óptimos. Dentro de este conjunto de datos, hemos identificado diversos elementos inusuales en las imágenes, los cuales se ilustran en la Figura 13.

Figura 13. Imagen con objeto inusual
Fuente: Elaboración propia

Para abordar esta problemática, se propone la aplicación de técnicas de recorte de imagen que permitan preservar su calidad original, o en su defecto, la edición de esta con la finalidad de ocultar objetos inusuales utilizando el tono de piel de la persona en la imagen. Este enfoque busca mejorar la calidad y coherencia del conjunto de imágenes suministrado (ver Figura 14).

Figura 14. Imagen sin objeto inusual
Fuente: Elaboración propia

Por otro lado, se procedió a la eliminación de aquellas imágenes que presentaban niveles de oscuridad considerable, lo cual generaba complicaciones en el proceso de entrenamiento de la red neuronal convolucional (ver Figura 15). Esta decisión se fundamenta en el reconocimiento de que la calidad de las imágenes desempeña un papel crucial en el desempeño de dicha red. Al remover las imágenes demasiado oscuras, se busca optimizar la capacidad de la red para extraer patrones y características significativas durante su fase de aprendizaje.

Figura 15. Imagen con nivel de oscuridad alto
Fuente: Elaboración propia

No obstante, la depuración de datos no se limitó únicamente a las imágenes con baja luminosidad. También se llevó a cabo la exclusión de aquellas capturas que carecían de un enfoque óptimo y una composición fotográfica adecuada (ver Figura 16). Esta medida se adoptó conscientemente, reconociendo que la calidad visual y la claridad de los elementos presentes en las imágenes son elementos esenciales para garantizar la precisión y la eficacia de la red neuronal.

Figura 16. Imagen sin claridad en los objectos.
Fuente: Elaboración propia

El total de las imágenes se cargó en Google Colab, creando un directorio para cada una de las clases de las imágenes. Así mismo estas imágenes se redimensionaron a un tamaño de 224 x 224 pixeles para su fácil uso. En la Figura 17 se muestran algunas de las imágenes utilizadas en los conjuntos de entrenamiento, validación y prueba, junto con su respectiva clasificación como cura o falla.

Figura 17. Clasificación de desenlace terapéutico.
Fuente: Elaboración propia

El conjunto de datos se dividió en imágenes destinadas a entrenamiento, validación y prueba. Se utilizaron 227 imágenes (70%) para entrenar los modelos, 48 imágenes (15%) para validar los modelos y otras 48 imágenes (15%) para probar los modelos en conjuntos de imágenes no utilizados previamente.

En el marco de este estudio, dada la limitada disponibilidad de imágenes en nuestro conjunto de datos, se emplea la técnica de aumento de datos como medida estratégica. Esta técnica implica la generación de instancias adicionales de cada imagen mediante la introducción de variaciones controladas. La estrategia de aumento de datos se ejecuta mediante la modificación estocástica del tono, saturación, contraste y brillo, lo cual contribuye a enriquecer la diversidad y complejidad del conjunto de datos. Además, se introduce aleatoriedad en la orientación de las imágenes mediante rotaciones en ángulos de 0, 90, 180 o 270 grados.

Un componente adicional de complejidad visual es incorporado a través de la aplicación de desenfoque gaussiano con un sigma aleatorio. Cabe destacar que todas estas modificaciones se realizan con una probabilidad del 50%, asegurando así una variabilidad adecuada sin comprometer la integridad del conjunto de datos. Este enfoque meticuloso busca mitigar las

limitaciones inherentes a la escasez de imágenes disponibles, garantizando al mismo tiempo la robustez y representatividad del conjunto de datos en el análisis subsiguiente. En la Tabla 2 se presenta el resumen de los datos y en el anexo 1 se presenta el código utilizado.

Tabla 2. Parámetros utilizados para la modificación de las imágenes.

Parámetro	Valor utilizado
Aplicación de transformación aleatoria	0.5
Modificar el contraste	(-0.3, 0.2)
Modificar el brillo	(-5, 3)
Rotar la imagen	[0, 90, 180, 270]
Desenfoque gaussiano	(5, 5)
Número de imágenes	80
Tamaño de las imágenes	225 x 225

Fuente: Elaboración propia

En una vertiente complementaria, se llevó a cabo una fusión de cada imagen de manera que permitiera visualizar la progresión de la leishmaniasis, desde su fase inicial hasta su desarrollo subsiguiente (ver Figura 18). El propósito subyacente de esta estrategia radica en elevar la precisión de la CNN. La adopción de esta medida surge como resultado de una serie de reuniones exhaustivas y deliberaciones sostenidas con los directores de la tesis. Este enfoque se establece como un paso fundamental para enriquecer la representación visual y la capacidad discriminativa del modelo, contribuyendo así a la efectividad global de la investigación. Para cumplir con esto se utilizó el código del Anexo 2.

Figura 18. Conjunto de imágenes que ilustran la comparativa entre el antes y el después del efecto del tratamiento.
Fuente: Elaboración propia

Concluyendo la fase preparatoria del conjunto de imágenes, se obtiene el siguiente conjunto de datos que servirá como base para el entrenamiento y validación de los modelos. En este contexto, se han empleado un total de 18,160 imágenes con el propósito de constituir la muestra de entrenamiento.

5. CONSTRUCCIÓN DE LOS MODELOS

5.1. Seleccionar los modelos para predecir el desenlace terapéutico de los pacientes teniendo en cuenta las imágenes de las lesiones

En este capítulo, se proporciona una descripción detallada de los ocho modelos empleados para predecir la evolución de los pacientes con LC. Inicialmente, se presentan las arquitecturas de estos modelos, seguido por una explicación detallada del proceso de optimización. Posteriormente, se detallan los parámetros del modelo más efectivo obtenido durante dicho proceso de optimización.

5.1.1. Modelo Propio

En el desarrollo de este primer modelo, los hiperparámetros se configuraron conforme a las directrices proporcionadas por la literatura especializada en el campo [49][50][51] y la experiencia acumulada por los investigadores del proyecto. La estructura del modelo se resume de manera visual en la Tabla 3, donde :

Layer: Indica el nombre o tipo de capa dentro de la arquitectura de la red neuronal. Puede ser una capa convolucional, una capa completamente conectada, una capa de normalización, entre otras. Cada tipo de capa realiza operaciones específicas en los datos.

Output Shape: Describe la forma de la salida de esa capa específica. Esta información es útil para entender cómo cambian las dimensiones de los datos a medida que se propagan a través de la red.

Param #: Representa el número de parámetros en cada capa. Estos parámetros son los valores que la red aprenderá durante el proceso de entrenamiento para ajustarse a los datos de entrada y generar las salidas deseadas.

Tabla 3. Hiperparámetros definidos para el modelo propio

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d (MaxPooling2D)	(None, 146, 146, 32)	0
conv2d_1 (Conv2D)	(None, 144, 144, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 71, 71, 64)	0
conv2d_2 (Conv2D)	(None, 69, 69, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 34, 34, 128)	0
flatten (Flatten)	(None, 147968)	0
dense (Dense)	(None, 256)	37880064
dense_1 (Dense)	(None, 1)	257
Total params:	**37973569 (144.86 MB)**	

Fuente: Elaboración propia

Para esta Modelo Propio, en la primera capa de convolución 2D incorpora 32 filtros, un kernel de dimensiones 3x3, y una función de activación Rectified Linear Unit (ReLU), con una dimensión de entrada especificada como (225, 255, 3). Posteriormente, se implementa una capa de reducción máxima (max pooling) 2D con un tamaño de pool de 2x2 y un stride de 2.

La segunda capa de convolución 2D se configura con 64 filtros, un kernel de 3x3, y utiliza la función de activación ReLU. Se añade otra capa de reducción máxima 2D con un tamaño de pool de 2x2 y un stride de 2.

En la tercera capa de convolución 2D se configura con 64 filtros, un kernel de 3x3, y utiliza la función de activación ReLU. Se añade otra capa de reducción máxima 2D con un tamaño de pool de 2x2 y un stride de 2.

Se introduce una tercera capa de convolución 2D que emplea 128 filtros, un kernel de 3x3 y la función de activación ReLU. Seguidamente, se incorpora una nueva capa de reducción máxima 2D con un tamaño de pool de 2x2 y un stride de 2.

La salida resultante se aplanó para conectar con capas densas (fully connected), incluyendo una capa densa con 256 unidades y función de activación ReLU. La capa de salida se define como densa con una unidad y función de activación sigmoide, especialmente adecuada para problemas de clasificación binaria. El modelo se compila utilizando el optimizador Adam, la función de pérdida binary_crossentropy y la métrica de precisión (accuracy).

5.1.2. VGG16 utilizando transfer learning

En el caso de la red VGG16, se implementa transfer learning, adaptando la arquitectura según las exigencias específicas de la investigación. Los hiperparámetros de esta estructura se definen considerando la experiencia de los investigadores, con el propósito de analizar detalladamente el rendimiento del modelo.

En la Tabla 4, se muestra la configuración inicial del modelo, El tamaño de la imagen de entrada se establece 250 × 250 × 3. La red VGG16 es una red neuronal convolucional que consta de varias capas, incluyendo capas de entrada, capas convolucionales (Conv2D), capas de agrupación máxima (MaxPooling2D), y capas completamente conectadas (Dense). Cada capa tiene una forma de salida específica y un número de parámetros asociados.

Las capas se organizan en bloques (block1, block2, etc.), cada uno con un conjunto específico de capas Conv2D seguidas por una capa MaxPooling2D. Al final del listado se encuentran las capas ‘flatten’ y ‘dense’, que son parte del clasificador conectado a la parte superior del Modelo Propio VGG16. Los totales al final indican el número total de parámetros, los parámetros entrenables y no entrenables. Los parámetros entrenables son aquellos que el modelo puede aprender y ajustar durante el entrenamiento, mientras que los parámetros no entrenables son fijos.

Tabla 4. Hiperparámetros definidos para el modelo VGG16

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv4 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv4 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv4 (Conv2D)	(None, 14, 14, 512)	2359808

block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 100)	2508900
dense_1 (Dense)	(None, 100)	10100
dense_2 (Dense)	(None, 100)	10100
dense_3 (Dense)	(None, 1)	101
Total params:	**22553585 (86.04 MB)**	

Fuente: Elaboración propia

5.1.3. VGG19 utilizando transfer learning.

En el marco de la red VGG19, se lleva a cabo la implementación de transfer learning, adaptando dicha red a los requisitos particulares de la investigación destinada a resolver un problema binario. Los hiperparámetros fueron definidos en base a la experiencia acumulada por los investigadores.

Tabla 5. Hiperparámetros definidos para el modelo VGG19

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0

block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv4 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv4 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv4 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 100)	2508900
dense_1 (Dense)	(None, 100)	10100
dense_2 (Dense)	(None, 100)	10100
dense_3 (Dense)	(None, 1)	101
Total params:	**22553585**	

Fuente: Elaboración propia

En la Tabla 5, se presenta la configuración inicial del modelo, con una dimensión de imagen de entrada de 250 × 250 × 3 píxeles. La arquitectura de la red VGG19 comprende diversas capas, incluyendo capas de entrada, capas convolucionales (Conv2D), capas de agrupación máxima (MaxPooling2D) y capas completamente conectadas (Dense). Cada capa tiene una salida

específica y un conjunto de parámetros asociados. Las capas están organizadas en bloques (block1, block2, etc.), cada uno compuesto por capas Conv2D seguidas de una capa MaxPooling2D. Al final del esquema se encuentran las capas 'flatten' y 'dense', integradas en el clasificador conectado en la parte superior del modelo VGG19. Los totales al final reflejan el número total de parámetros, diferenciando entre parámetros entrenables (ajustables durante el entrenamiento) y no entrenables (constantes).

5.1.4. Red-Net50 utilizando transfer learning.

Se utiliza la arquitectura ResNet50 como punto de partida, una red neuronal convolucional profunda ampliamente reconocida por su popularidad y eficacia (Ver Tabla 6). En la configuración de esta red, se han especificado las siguientes consideraciones: el parámetro "weights" se estableció en "imagenet", posibilitando la carga de pesos preentrenados provenientes del conjunto de datos ImageNet. Por otro lado, la opción "include_top=False" fue seleccionada para excluir las capas completamente conectadas al final de ResNet50, ya que se añadirán capas personalizadas específicas para la tarea en cuestión.

Con el propósito de transformar la salida tridimensional de la parte convolucional en un vector unidimensional, se implementó la capa "Flatten()". Además, se definieron dos capas densas con los valores predeterminados de "Dense (100, activation='relu')". Cada una de estas capas densas cuenta con 100 neuronas y emplea la función de activación ReLU. La elección de la activación sigmoide es común en problemas de clasificación binaria, ya que produce una salida en el rango de 0 a 1, interpretada como la probabilidad de pertenencia a la clase positiva.

La instrucción "Model(inputs=base_model.input, outputs=predictions)" combina el modelo (ResNet50) con las capas personalizadas (capas densas y de salida), dando forma al modelo completo. En este contexto, se establece que las capas del modelo base no son entrenables, lo que implica que sus pesos no se actualizarán durante el entrenamiento; únicamente las capas personalizadas serán objeto de entrenamiento.

Tabla 6. Hiperparámetros definidos para el modelo ResNet50

Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	[(None, 224, 224, 3)]	0	[]
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	['input_2 [0] [0]']
conv1_conv (Conv2D)	(None, 112, 112, 64)	9472	['conv1_pad [0] [0]']
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	256	['conv1_conv [0] [0]']
conv1_relu (Activation)	(None, 112, 112, 64)	0	['conv1_bn [0] [0]']
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	['conv1_relu [0] [0]']
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	['pool1_pad [0] [0]']
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4160	['pool1_pool [0] [0]']
conv2_block1_1_bn (BatchNormalization)	(None, 56, 56, 64)	256	['conv2_block1_1_conv [0] [0]']
conv2_block1_1_relu (Activation)	(None, 56, 56, 64)	0	['conv2_block1_1_bn [0] [0]']
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36928	['conv2_block1_1_relu [0] [0]']
conv2_block1_2_bn (BatchNormalization)	(None, 56, 56, 64)	256	['conv2_block1_2_conv [0] [0]']
conv2_block1_2_relu (Activation)	(None, 56, 56, 64)	0	['conv2_block1_2_bn [0] [0]']
conv2_block1_0_conv (Conv2D)	(None, 56, 56, 256)	16640	['pool1_pool [0] [0]']
conv2_block1_3_conv (Conv2D)	(None, 56, 56, 256)	16640	['conv2_block1_2_relu [0] [0]']
conv2_block1_0_bn (BatchNormalization)	(None, 56, 56, 256)	1024	['conv2_block1_0_conv [0] [0]']

conv2_block1_3_bn (BatchNormalization)	(None, 56, 56, 256)	1024	['conv2_block1_3_conv [0] [0]']
conv2_block1_add (Add)	(None, 56, 56, 256)	0	['conv2_block1_0_bn [0] [0]','conv2_block1_3_bn [0] [0]']
conv2_block1_out (Activation)	(None, 56, 56, 256)	0	['conv2_block1_add [0] [0]']
conv2_block2_1_conv (Conv2D)	(None, 56, 56, 64)	16448	['conv2_block1_out [0] [0]']
conv2_block2_1_bn (BatchNormalization)	(None, 56, 56, 64)	256	['conv2_block2_1_conv [0] [0]']
conv2_block2_1_relu (Activation)	(None, 56, 56, 64)	0	['conv2_block2_1_bn [0] [0]']
conv2_block2_2_conv (Conv2D)	(None, 56, 56, 64)	36928	['conv2_block2_1_relu [0] [0]']
conv2_block2_2_bn (BatchNormalization)	(None, 56, 56, 64)	256	['conv2_block2_2_conv [0] [0]']
conv2_block2_2_relu (Activation)	(None, 56, 56, 64)	0	['conv2_block2_2_bn [0] [0]']
conv2_block2_3_conv (Conv2D)	(None, 56, 56, 256)	16640	['conv2_block2_2_relu [0] [0]']
conv2_block2_3_bn (BatchNormalization)	(None, 56, 56, 256)	1024	['conv2_block2_3_conv [0] [0]']
conv2_block2_add (Add)	(None, 56, 56, 256)	0	['conv2_block1_out [0] [0]',
conv2_block2_out (Activation)	(None, 56, 56, 256)	0	['conv2_block2_add [0] [0]']
conv2_block3_1_conv (Conv2D)	(None, 56, 56, 64)	16448	['conv2_block2_out [0] [0]']
conv2_block3_1_bn (BatchNormalization)	(None, 56, 56, 64)	256	['conv2_block3_1_conv [0] [0]']
conv2_block3_1_relu (Activation)	(None, 56, 56, 64)	0	['conv2_block3_1_bn [0] [0]']

conv2_block3_2_conv (Conv2D)	(None, 56, 56, 64)	36928	['conv2_block3_1_relu [0] [0]']
conv2_block3_2_bn (BatchNormalization)	(None, 56, 56, 64)	256	['conv2_block3_2_conv [0] [0]']
conv2_block3_2_relu (Activation)	(None, 56, 56, 64)	0	['conv2_block3_2_bn [0] [0]']
conv2_block3_3_conv (Conv2D)	(None, 56, 56, 256)	16640	['conv2_block3_2_relu [0] [0]']
conv2_block3_3_bn (BatchNormalization)	(None, 56, 56, 256)	1024	['conv2_block3_3_conv [0] [0]']
conv2_block3_add (Add)	(None, 56, 56, 256)	0	['conv2_block2_out [0] [0]',
conv2_block3_out (Activation)	(None, 56, 56, 256)	0	['conv2_block3_add [0] [0]']
conv3_block1_1_conv (Conv2D)	(None, 28, 28, 128)	32896	['conv2_block3_out [0] [0]']
conv3_block1_1_bn (BatchNormalization)	(None, 28, 28, 128)	512	['conv3_block1_1_conv [0] [0]']
conv3_block1_1_relu (Activation)	(None, 28, 28, 128)	0	['conv3_block1_1_bn [0] [0]']
conv3_block1_2_conv (Conv2D)	(None, 28, 28, 128)	147584	['conv3_block1_1_relu [0] [0]']
conv3_block1_2_bn (BatchNormalization)	(None, 28, 28, 128)	512	['conv3_block1_2_conv [0] [0]']
conv3_block1_2_relu (Activation)	(None, 28, 28, 128)	0	['conv3_block1_2_bn [0] [0]']
conv3_block1_0_conv (Conv2D)	(None, 28, 28, 512)	131584	['conv2_block3_out [0] [0]']
conv3_block1_3_conv (Conv2D)	(None, 28, 28, 512)	66048	['conv3_block1_2_relu [0] [0]']
conv3_block1_0_bn (BatchNormalization)	(None, 28, 28, 512)	2048	['conv3_block1_0_conv [0] [0]']

conv3_block1_3_bn (BatchNormalization)	(None, 28, 28, 512)	2048	['conv3_block1_3_conv [0] [0]']
conv3_block1_add (Add)	(None, 28, 28, 512)	0	['conv3_block1_0_bn [0] [0]',
conv3_block1_out (Activation)	(None, 28, 28, 512)	0	['conv3_block1_add [0] [0]']
conv3_block2_1_conv (Conv2D)	(None, 28, 28, 128)	65664	['conv3_block1_out [0] [0]']
conv3_block2_1_bn (BatchNormalization)	(None, 28, 28, 128)	512	['conv3_block2_1_conv [0] [0]']
conv3_block2_1_relu (Activation)	(None, 28, 28, 128)	0	['conv3_block2_1_bn [0] [0]']
conv3_block2_2_conv (Conv2D)	(None, 28, 28, 128)	147584	['conv3_block2_1_relu [0] [0]']
conv3_block2_2_bn (BatchNormalization)	(None, 28, 28, 128)	512	['conv3_block2_2_conv [0] [0]']
conv3_block2_2_relu (Activation)	(None, 28, 28, 128)	0	['conv3_block2_2_bn [0] [0]']
conv3_block2_3_conv (Conv2D)	(None, 28, 28, 512)	66048	['conv3_block2_2_relu [0] [0]']
conv3_block2_3_bn (BatchNormalization)	(None, 28, 28, 512)	2048	['conv3_block2_3_conv [0] [0]']
conv3_block2_add (Add)	(None, 28, 28, 512)	0	['conv3_block1_out [0] [0]',
conv3_block2_out (Activation)	(None, 28, 28, 512)	0	['conv3_block2_add [0] [0]']
conv3_block3_1_conv (Conv2D)	(None, 28, 28, 128)	65664	['conv3_block2_out [0] [0]']
conv3_block3_1_bn (BatchNormalization)	(None, 28, 28, 128)	512	['conv3_block3_1_conv [0] [0]']
conv3_block3_1_relu (Activation)	(None, 28, 28, 128)	0	['conv3_block3_1_bn [0] [0]']

conv3_block3_2_conv (Conv2D)	(None, 28, 28, 128)	147584	['conv3_block3_1_relu [0] [0]']
conv3_block3_2_bn (BatchNormalization)	(None, 28, 28, 128)	512	['conv3_block3_2_conv [0] [0]']
conv3_block3_2_relu (Activation)	(None, 28, 28, 128)	0	['conv3_block3_2_bn [0] [0]']
conv3_block3_3_conv (Conv2D)	(None, 28, 28, 512)	66048	['conv3_block3_2_relu [0] [0]']
conv3_block3_3_bn (BatchNormalization)	(None, 28, 28, 512)	2048	['conv3_block3_3_conv [0] [0]']
conv3_block3_add (Add)	(None, 28, 28, 512)	0	['conv3_block2_out [0] [0]',
conv3_block3_out (Activation)	(None, 28, 28, 512)	0	['conv3_block3_add [0] [0]']
conv3_block4_1_conv (Conv2D)	(None, 28, 28, 128)	65664	['conv3_block3_out [0] [0]']
conv3_block4_1_bn (BatchNormalization)	(None, 28, 28, 128)	512	['conv3_block4_1_conv [0] [0]']
conv3_block4_1_relu (Activation)	(None, 28, 28, 128)	0	['conv3_block4_1_bn [0] [0]']
conv3_block4_2_conv (Conv2D)	(None, 28, 28, 128)	147584	['conv3_block4_1_relu [0] [0]']
conv3_block4_2_bn (BatchNormalization)	(None, 28, 28, 128)	512	['conv3_block4_2_conv [0] [0]']
conv3_block4_2_relu (Activation)	(None, 28, 28, 128)	0	['conv3_block4_2_bn [0] [0]']
conv3_block4_3_conv (Conv2D)	(None, 28, 28, 512)	66048	['conv3_block4_2_relu [0] [0]']
conv3_block4_3_bn (BatchNormalization)	(None, 28, 28, 512)	2048	['conv3_block4_3_conv [0] [0]']
conv3_block4_add (Add)	(None, 28, 28, 512)	0	['conv3_block3_out [0] [0]',

conv3_block4_out (Activation)	(None, 28, 28, 512)	0	['conv3_block4_add [0] [0]']
conv4_block1_1_conv (Conv2D)	(None, 14, 14, 256)	131328	['conv3_block4_out [0] [0]']
conv4_block1_1_bn (BatchNormalization)	(None, 14, 14, 256)	1024	['conv4_block1_1_conv [0] [0]']
conv4_block1_1_relu (Activation)	(None, 14, 14, 256)	0	['conv4_block1_1_bn [0] [0]']
conv4_block1_2_conv (Conv2D)	(None, 14, 14, 256)	590080	['conv4_block1_1_relu [0] [0]']
conv4_block1_2_bn (BatchNormalization)	(None, 14, 14, 256)	1024	['conv4_block1_2_conv [0] [0]']
conv4_block1_2_relu (Activation)	(None, 14, 14, 256)	0	['conv4_block1_2_bn [0] [0]']
conv4_block1_0_conv (Conv2D)	(None, 14, 14, 1024)	525312	['conv3_block4_out [0] [0]']
conv4_block1_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	['conv4_block1_2_relu [0] [0]']
conv4_block1_0_bn (BatchNormalization)	(None, 14, 14, 1024)	4096	['conv4_block1_0_conv [0] [0]']
conv4_block1_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4096	['conv4_block1_3_conv [0] [0]']
conv4_block1_add (Add)	(None, 14, 14, 1024)	0	['conv4_block1_0_bn [0] [0]',
conv4_block1_out (Activation)	(None, 14, 14, 1024)	0	['conv4_block1_add [0] [0]']
conv4_block2_1_conv (Conv2D)	(None, 14, 14, 256)	262400	['conv4_block1_out [0] [0]']
conv4_block2_1_bn (BatchNormalization)	(None, 14, 14, 256)	1024	['conv4_block2_1_conv [0] [0]']
conv4_block2_1_relu (Activation)	(None, 14, 14, 256)	0	['conv4_block2_1_bn [0] [0]']

conv4_block2_2_conv (Conv2D)	(None, 14, 14, 256)	590080	['conv4_block2_1_relu [0] [0]']
conv4_block2_2_bn (BatchNormalization)	(None, 14, 14, 256)	1024	['conv4_block2_2_conv [0] [0]']
conv4_block2_2_relu (Activation)	(None, 14, 14, 256)	0	['conv4_block2_2_bn [0] [0]']
conv4_block2_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	['conv4_block2_2_relu [0] [0]']
conv4_block2_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4096	['conv4_block2_3_conv [0] [0]']
conv4_block2_add (Add)	(None, 14, 14, 1024)	0	['conv4_block1_out [0] [0]',
conv4_block2_out (Activation)	(None, 14, 14, 1024)	0	['conv4_block2_add [0] [0]']
conv4_block3_1_conv (Conv2D)	(None, 14, 14, 256)	262400	['conv4_block2_out [0] [0]']
conv4_block3_1_bn (BatchNormalization)	(None, 14, 14, 256)	1024	['conv4_block3_1_conv [0] [0]']
conv4_block3_1_relu (Activation)	(None, 14, 14, 256)	0	['conv4_block3_1_bn [0] [0]']
conv4_block3_2_conv (Conv2D)	(None, 14, 14, 256)	590080	['conv4_block3_1_relu [0] [0]']
conv4_block3_2_bn (BatchNormalization)	(None, 14, 14, 256)	1024	['conv4_block3_2_conv [0] [0]']
conv4_block3_2_relu (Activation)	(None, 14, 14, 256)	0	['conv4_block3_2_bn [0] [0]']
conv4_block3_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	['conv4_block3_2_relu [0] [0]']
conv4_block3_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4096	['conv4_block3_3_conv [0] [0]']
conv4_block3_add (Add)	(None, 14, 14, 1024)	0	['conv4_block2_out [0] [0]',

conv4_block3_out (Activation)	(None, 14, 14, 1024)	0	['conv4_block3_add [0] [0]']
conv4_block4_1_conv (Conv2D)	(None, 14, 14, 256)	262400	['conv4_block3_out [0] [0]']
conv4_block4_1_bn (BatchNormalization)	(None, 14, 14, 256)	1024	['conv4_block4_1_conv [0] [0]']
conv4_block4_1_relu (Activation)	(None, 14, 14, 256)	0	['conv4_block4_1_bn [0] [0]']
conv4_block4_2_conv (Conv2D)	(None, 14, 14, 256)	590080	['conv4_block4_1_relu [0] [0]']
conv4_block4_2_bn (BatchNormalization)	(None, 14, 14, 256)	1024	['conv4_block4_2_conv [0] [0]']
conv4_block4_2_relu (Activation)	(None, 14, 14, 256)	0	['conv4_block4_2_bn [0] [0]']
conv4_block4_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	['conv4_block4_2_relu [0] [0]']
conv4_block4_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4096	['conv4_block4_3_conv [0] [0]']
conv4_block4_add (Add)	(None, 14, 14, 1024)	0	['conv4_block3_out [0] [0]',
conv4_block4_out (Activation)	(None, 14, 14, 1024)	0	['conv4_block4_add [0] [0]']
conv4_block5_1_conv (Conv2D)	(None, 14, 14, 256)	262400	['conv4_block4_out [0] [0]']
conv4_block5_1_bn (BatchNormalization)	(None, 14, 14, 256)	1024	['conv4_block5_1_conv [0] [0]']
conv4_block5_1_relu (Activation)	(None, 14, 14, 256)	0	['conv4_block5_1_bn [0] [0]']
conv4_block5_2_conv (Conv2D)	(None, 14, 14, 256)	590080	['conv4_block5_1_relu [0] [0]']
conv4_block5_2_bn (BatchNormalization)	(None, 14, 14, 256)	1024	['conv4_block5_2_conv [0] [0]']

conv4_block5_2_relu (Activation)	(None, 14, 14, 256)	0	['conv4_block5_2_bn [0] [0]']
conv4_block5_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	['conv4_block5_2_relu [0] [0]']
conv4_block5_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4096	['conv4_block5_3_conv [0] [0]']
conv4_block5_add (Add)	(None, 14, 14, 1024)	0	['conv4_block4_out [0] [0]',
conv4_block5_out (Activation)	(None, 14, 14, 1024)	0	['conv4_block5_add [0] [0]']
conv4_block6_1_conv (Conv2D)	(None, 14, 14, 256)	262400	['conv4_block5_out [0] [0]']
conv4_block6_1_bn (BatchNormalization)	(None, 14, 14, 256)	1024	['conv4_block6_1_conv [0] [0]']
conv4_block6_1_relu (Activation)	(None, 14, 14, 256)	0	['conv4_block6_1_bn [0] [0]']
conv4_block6_2_conv (Conv2D)	(None, 14, 14, 256)	590080	['conv4_block6_1_relu [0] [0]']
conv4_block6_2_bn (BatchNormalization)	(None, 14, 14, 256)	1024	['conv4_block6_2_conv [0] [0]']
conv4_block6_2_relu (Activation)	(None, 14, 14, 256)	0	['conv4_block6_2_bn [0] [0]']
conv4_block6_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	['conv4_block6_2_relu [0] [0]']
conv4_block6_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4096	['conv4_block6_3_conv [0] [0]']
conv4_block6_add (Add)	(None, 14, 14, 1024)	0	['conv4_block5_out [0] [0]',
conv4_block6_out (Activation)	(None, 14, 14, 1024)	0	['conv4_block6_add [0] [0]']
conv5_block1_1_conv (Conv2D)	(None, 7, 7, 512)	524800	['conv4_block6_out [0] [0]']

conv5_block1_1_bn (BatchNormalization)	(None, 7, 7, 512)	2048	['conv5_block1_1_conv [0] [0]']
conv5_block1_1_relu (Activation)	(None, 7, 7, 512)	0	['conv5_block1_1_bn [0] [0]']
conv5_block1_2_conv (Conv2D)	(None, 7, 7, 512)	2359808	['conv5_block1_1_relu [0] [0]']
conv5_block1_2_bn (BatchNormalization)	(None, 7, 7, 512)	2048	['conv5_block1_2_conv [0] [0]']
conv5_block1_2_relu (Activation)	(None, 7, 7, 512)	0	['conv5_block1_2_bn [0] [0]']
conv5_block1_0_conv (Conv2D)	(None, 7, 7, 2048)	2099200	['conv4_block6_out [0] [0]']
conv5_block1_3_conv (Conv2D)	(None, 7, 7, 2048)	1050624	['conv5_block1_2_relu [0] [0]']
conv5_block1_0_bn (BatchNormalization)	(None, 7, 7, 2048)	8192	['conv5_block1_0_conv [0] [0]']
conv5_block1_3_bn (BatchNormalization)	(None, 7, 7, 2048)	8192	['conv5_block1_3_conv [0] [0]']
conv5_block1_add (Add)	(None, 7, 7, 2048)	0	['conv5_block1_0_bn [0] [0]',
conv5_block1_out (Activation)	(None, 7, 7, 2048)	0	['conv5_block1_add [0] [0]']
conv5_block2_1_conv (Conv2D)	(None, 7, 7, 512)	1049088	['conv5_block1_out [0] [0]']
conv5_block2_1_bn (BatchNormalization)	(None, 7, 7, 512)	2048	['conv5_block2_1_conv [0] [0]']
conv5_block2_1_relu (Activation)	(None, 7, 7, 512)	0	['conv5_block2_1_bn [0] [0]']
conv5_block2_2_conv (Conv2D)	(None, 7, 7, 512)	2359808	['conv5_block2_1_relu [0] [0]']
conv5_block2_2_bn (BatchNormalization)	(None, 7, 7, 512)	2048	['conv5_block2_2_conv [0] [0]']

conv5_block2_2_relu (Activation)	(None, 7, 7, 512)	0	['conv5_block2_2_bn [0] [0]']
conv5_block2_3_conv (Conv2D)	(None, 7, 7, 2048)	1050624	['conv5_block2_2_relu [0] [0]']
conv5_block2_3_bn (BatchNormalization)	(None, 7, 7, 2048)	8192	['conv5_block2_3_conv [0] [0]']
conv5_block2_add (Add)	(None, 7, 7, 2048)	0	['conv5_block1_out [0] [0]',
conv5_block2_out (Activation)	(None, 7, 7, 2048)	0	['conv5_block2_add [0] [0]']
conv5_block3_1_conv (Conv2D)	(None, 7, 7, 512)	1049088	['conv5_block2_out [0] [0]']
conv5_block3_1_bn (BatchNormalization)	(None, 7, 7, 512)	2048	['conv5_block3_1_conv [0] [0]']
conv5_block3_1_relu (Activation)	(None, 7, 7, 512)	0	['conv5_block3_1_bn [0] [0]']
conv5_block3_2_conv (Conv2D)	(None, 7, 7, 512)	2359808	['conv5_block3_1_relu [0] [0]']
conv5_block3_2_bn (BatchNormalization)	(None, 7, 7, 512)	2048	['conv5_block3_2_conv [0] [0]']
conv5_block3_2_relu (Activation)	(None, 7, 7, 512)	0	['conv5_block3_2_bn [0] [0]']
conv5_block3_3_conv (Conv2D)	(None, 7, 7, 2048)	1050624	['conv5_block3_2_relu [0] [0]']
conv5_block3_3_bn (BatchNormalization)	(None, 7, 7, 2048)	8192	['conv5_block3_3_conv [0] [0]']
conv5_block3_add (Add)	(None, 7, 7, 2048)	0	['conv5_block2_out [0] [0]',
conv5_block3_out (Activation)	(None, 7, 7, 2048)	0	['conv5_block3_add [0] [0]']
flatten_2 (Flatten)	(None, 100352)	0	['conv5_block3_out [0] [0]']
dense_4 (Dense)	(None, 100)	1003530	['flatten_2 [0] [0]']
dense_5 (Dense)	(None, 100)	10100	['dense_4 [0] [0]']

dense_6 (Dense)	(None, 100)	10100	['dense_5 [0] [0]']
dense_7 (Dense)	(None, 1)	101	['dense_6 [0] [0]']

Fuente: Elaboración propia

5.2. Optimización de los modelos para predecir el desenlace terapéutico

5.2.1. Modelo Propio utilizando Grid search

En la configuración de este modelo, se empleó la búsqueda definida mediante una técnica conocida como búsqueda aleatoria (Grid search). La búsqueda aleatoria se caracteriza por ser un método exhaustivo destinado a la identificación de hiperparámetros óptimos. En este enfoque, se define un espacio de búsqueda como un dominio acotado de valores de hiperparámetros, seleccionándose puntos dentro de dicho dominio de manera aleatoria. Este procedimiento permite evaluar el rendimiento asociado a cada combinación de hiperparámetros [52].
Por lo tanto, proponemos una arquitectura que utiliza la búsqueda aleatoria de hiperparámetros dentro de un espacio de búsqueda predefinido. La arquitectura CNN propuesta parte del Modelo Propio incorporando hiperparámetros nuevos (Ver Tabla 7).

Tabla 7. Hiperparámetros definidos en el modelo propio utilizando Grid search

Hiperparámetros	Nombre	Valores probados
num_conv_layers	Número de capas convolucionales	(1,2,3)
num_filters	Numero de filtros por capa	(32,64)
kernel_size	Tamaño del kernel	(3,3), (5,5)
pool_size	Tamaño del pooling	(2,2), (3,3)
strides	Strides	(1, 2) (2, 3)
num_dense_layers	Numero de capas densas	(2,3)
num_units	Número de unidades por capa densa	(64, 128, 256)

optimizer	Optimizador	(Adam, RMsprop)
Dropout	Dropout	(0.25, 0.5, 0.8)
learning_rate	Tasa de aprendizaje	(0.01, 0.001)

Fuente: Elaboración propia

En el marco de este segundo modelo, se introducen de manera estratégica nuevos hiperparámetros, mejorando la configuración inicial del Modelo Propio. Destacan elementos cruciales como el optimizador, el dropout, y la tasa de aprendizaje, todos ellos meticulosamente seleccionados para potenciar el proceso de entrenamiento y elevar la precisión del modelo a niveles significativos.

El modelo se inicia con una capa convolucional que emplea num_filters filtros de tamaño kernel_size, con una función de activación Rectified Linear Unit (ReLU). Posteriormente, se aplica una capa de reducción máxima (max pooling) con un tamaño de pool de pool_size y un stride de strides, seguida de una capa de dropout que desactiva aleatoriamente un porcentaje (dropout) de las neuronas para prevenir el sobreajuste.

Para permitir la construcción de modelos más complejos, la función admite la adición de múltiples capas convolucionales y de pooling. Esto se logra mediante un bucle que itera num_conv_layers-1 veces, agregando capas convolucionales, de reducción máxima y dropout en cada iteración. Después de estas capas, se aplica una capa de aplanado (Flatten()) para convertir la salida en un vector unidimensional, preparándola para la conexión con capas totalmente conectadas.

En la siguiente fase, la función agrega capas densas (fully connected) a la red. Este proceso se repite num_dense_layers veces, cada vez utilizando una capa densa con num_units unidades y una función de activación ReLU. Finalmente, se añade una capa de salida densa con una sola unidad y una función de activación sigmoide, apropiada para problemas de clasificación binaria.

En cuanto al optimizador, la función permite elegir entre 'Adam' y 'RMSprop', ajustando la tasa de aprendizaje según el valor proporcionado (learning_rate). La compilación del modelo se realiza con la función de pérdida 'binary_crossentropy', adecuada para problemas de clasificación binaria, y la métrica de precisión ('accuracy').

La automatización completa de la búsqueda del número óptimo de capas agrega un nivel adicional de eficiencia y precisión. Tradicionalmente, la elección del número de capas era una tarea empírica que requería experiencia y numerosas iteraciones de prueba y error. Sin embargo, mediante la automatización, podemos explorar diferentes arquitecturas de CNN con diferentes números de capas y evaluar su rendimiento de manera sistemática.

5.2.2. VGG16 utilizando Grid search

Dentro del ámbito de esta red neuronal, proponemos una arquitectura que utiliza la exploración aleatoria de hiperparámetros en un espacio previamente establecido. Para este modelo, hemos delimitado la búsqueda del número de capas densas a los valores (1, 2, 3) y el número de filtros en el rango de (128, 256, 512, 1024) (Ver Tabla 8). Esta configuración de la red neuronal convolucional se deriva de un modelo base al que se le incorporan hiperparámetros adicionales, introduciendo así nuevas dimensiones y ajustes para potenciar su rendimiento

Tabla 8. Hiperparámetros definidos para el modelo VGG16 utilizando Grid search

Parámetro	Valor
Num_dense_layers	[2,3,4]
Num_units	[128,256,512,1024]

Fuente: Elaboración propia

El código comienza definiendo una función llamada build_model, que construye un modelo de clasificación binaria utilizando la arquitectura preentrenada VGG16 como base. Esta función toma dos parámetros opcionales: num_dense_layers y num_units, que determinan la cantidad de capas densas y la cantidad de unidades en cada capa densa, respectivamente. Luego, se configura un objeto EarlyStopping que monitorea la pérdida en el conjunto de validación y detiene el entrenamiento si no hay mejoras durante cinco épocas, restaurando los mejores pesos. Se establece un diccionario llamado param_grid_part1 que especifica las combinaciones de hiperparámetros a probar en una búsqueda en cuadrícula. Los parámetros incluyen el número de capas densas y el número de unidades por capa densa.

A continuación, se crea un objeto KerasClassifier que utiliza la función build_model y se configura para ejecutar el modelo en modo silencioso (verbose=0). Se emplea GridSearchCV para realizar una búsqueda en cuadrícula con validación cruzada (10-fold) y la métrica de evaluación es el F1-score. La búsqueda se realiza con los datos de entrenamiento y prueba extraídos de generadores de flujo de datos.

Después de ajustar el modelo utilizando la búsqueda en cuadrícula, se extraen el mejor modelo (best_model_part1) y los mejores parámetros (best_params_part1). Estos resultados pueden utilizarse para construir y entrenar el modelo final con los hiperparámetros óptimos encontrados durante la búsqueda en cuadrícula.

5.2.3. VGG19 utilizando Grid search

En el contexto de esta red neuronal, proponemos una arquitectura que emplea la búsqueda aleatoria de hiperparámetros en un espacio predefinido. Para este modelo, hemos delimitado la búsqueda del número de capas densas a los valores (1, 2, 3) y el número de filtros en el rango de (128, 256, 512, 1024). Esta arquitectura de CNN se origina a partir de un modelo base al que se le incorporan hiperparámetros adicionales (Ver Tabla 9)

Tabla 9. Hiperparámetros definidos para el modelo VGG19 utilizando Grid search

Parámetro	Valor
Num_dense_layers	[2,3,4]
Num_units	[128,256,512,1024]

Fuente: Elaboración propia

Este código en Python está diseñado para realizar una búsqueda de hiperparámetros en un modelo de red neuronal convolucional (CNN) utilizando la arquitectura preentrenada VGG19. A continuación, se proporciona una descripción detallada del código:

La función build_model define la arquitectura del modelo. Utiliza la arquitectura VGG19 con pesos preentrenados en el conjunto de datos ImageNet. Se congela la parte convolucional de VGG19

para que los pesos no se actualicen durante el entrenamiento. La arquitectura del modelo personalizado consiste en agregar la VGG19 como una capa, aplanar la salida y agregar capas totalmente conectadas con una función de activación ReLU. La última capa tiene una activación sigmoide para realizar una clasificación binaria. El modelo se compila con el optimizador Adam y la función de pérdida de entropía cruzada binaria. Se define un callback EarlyStopping para detener el entrenamiento si la pérdida en el conjunto de validación no mejora después de cierto número de épocas y restaurar los mejores pesos.

En la "Parte 1", se configura una cuadrícula de hiperparámetros que se probarán, especificando diferentes números de capas densas (num_dense_layers) y unidades en las capas densas (num_units). Luego, se utiliza la clase KerasClassifier para encapsular el modelo con los parámetros especificados y se emplea GridSearchCV para realizar una búsqueda exhaustiva de hiperparámetros utilizando validación cruzada de 10 pliegues y evaluación basada en la métrica F1. Los datos de entrenamiento y prueba se extraen de generadores de flujo de datos, y el modelo se ajusta a los datos de entrenamiento. Durante el entrenamiento, se utiliza el callback EarlyStopping para evitar el sobreajuste. Finalmente, se almacena el mejor modelo encontrado y sus hiperparámetros asociados en las variables best_model_part1 y best_params_part1, respectivamente.

5.2.4. Red-Net50 utilizando Grid search

En el contexto de esta red neuronal, proponemos una arquitectura que emplea la búsqueda aleatoria de hiperparámetros en un espacio predefinido. Para este modelo, hemos delimitado la búsqueda del número de capas densas a los valores (1, 2, 3) y el número de filtros en el rango de (128, 256, 512, 1024) (Ver Tabla 10).

Tabla 10. Hiperparámetros definidos para el modelo Red-Net50 utilizando Grid search

Parámetro	Valor
Num_dense_layers	[2,3,4]
Num_units	[128,256,512,1024]

Fuente: Elaboración propia

1. Definición del Modelo: La función build_model crea un modelo de CNN con ResNet50 como base, preentrenada en el conjunto de datos ImageNet. La capa de entrada se ajusta a dimensiones específicas (width_shape y height_shape). La red ResNet50 se establece como no entrenable para conservar sus características preentrenadas. Luego, se agregan capas densas y una capa de salida con activación sigmoide. El modelo se compila con el optimizador Adam, la pérdida binaria de entropía cruzada y la métrica de precisión.

2. Configuración de Early Stopping: Se establece el objeto early_stopping, que actúa como un mecanismo de parada anticipada durante el entrenamiento. Si la pérdida en el conjunto de validación no mejora después de 5 épocas, se restauran los mejores pesos del modelo.

3. Búsqueda de Hiperparámetros: Se define un conjunto de hiperparámetros (param_grid_part1) que serán explorados durante la búsqueda, incluyendo opciones para el número de capas densas (num_dense_layers) y el número de unidades en estas capas (num_units). Un clasificador Keras (KerasClassifier) se configura con la función build_model como constructor y otros parámetros, como verbose = 0 para controlar la cantidad de información durante el entrenamiento. Se utiliza GridSearchCV para realizar una búsqueda exhaustiva de hiperparámetros. El conjunto de datos de entrenamiento se divide en 10 partes para la validación cruzada, y la métrica de evaluación es la puntuación F1.

4. Preparación y Entrenamiento del Modelo: Imágenes y etiquetas se extraen del generador de flujo de datos. El modelo se ajusta utilizando los datos de entrenamiento durante 5 épocas, con validación en los datos de prueba. Se utiliza el objeto early_stopping como callback para monitorear la pérdida de validación.

5. Resultados de la Búsqueda de Hiperparámetros: best_model_part1 y best_params_part1 almacenan el mejor modelo y los mejores parámetros encontrados durante la búsqueda, respectivamente.

5.3. Entrenar los modelos a partir de la base de datos procesada

En esta sección, detallaremos los procedimientos llevados a cabo durante el entrenamiento de todos los modelos. Para una mayor claridad, optaremos por explicar las dos fases utilizando un único modelo como referencia, dado que la metodología empleada se replica de manera coherente en los demás. En la primera etapa, seguimos las pautas establecidas por la literatura y

la experiencia de los investigadores, definiendo hiperparámetros iniciales. En la segunda fase, llevamos a cabo una búsqueda exhaustiva con el objetivo de optimizar los hiperparámetros de los ocho modelos propuestos.

La importancia de este procedimiento reside en la capacidad de los modelos para discernir patrones y relaciones a partir de los datos procesados. Esta habilidad, a su vez, facilita la realización de predicciones precisas y significativas. La naturaleza iterativa de este proceso nos proporciona la flexibilidad necesaria para seleccionar el modelo que mejor se ajuste a nuestros datos y que cumpla con los objetivos establecidos en nuestras predicciones.

5.3.1. Codificación de los modelos a partir de hiperparámetros definidos

Modelo Propio

La automatización completa de la búsqueda del número óptimo de capas agrega un nivel. Los modelos se ejecutaron en una máquina virtual que fue configurada con una memoria RAM de 50 GB y un procesador Intel® Xeon® CPU E5-2660 v3 a 2.60 GHz, 2597 Mhz, con 2 núcleos físicos y 2 núcleos lógicos. La configuración de los hiperparámetros de la CNN para el entrenamiento de la red se llevó a cabo en tres etapas, con el objetivo de minimizar el costo computacional. Estas etapas fueron cuidadosamente estructuradas teniendo en cuenta tanto la importancia relativa de los valores de los hiperparámetros como la experiencia de los investigadores involucrados en el proyecto.

Etapa 1:

Aquí se define un objeto de EarlyStopping, que es una técnica para detener el entrenamiento del modelo antes de que haya ocurrido un sobreajuste. La monitorización se realiza en función de la pérdida en el conjunto de validación ('val_loss'). El entrenamiento se detendrá después de 10 épocas si no se observa una mejora en la pérdida en el conjunto de validación, y se restaurarán los pesos del modelo al mejor conjunto de pesos.

Se define un conjunto de hiperparámetros que se utilizarán para la búsqueda. Se especifican diferentes opciones para el número de capas convolucionales, el número de filtros, el tamaño del kernel, el tamaño de la piscina y los pasos de la capa convolucional. Se crea un clasificador Keras utilizando la función build_model. Este clasificador se utilizará en la búsqueda de cuadrícula para evaluar diferentes combinaciones de hiperparámetros.

Se realiza la búsqueda de cuadrícula utilizando la clase GridSearchCV de scikit-learn. La búsqueda se realiza sobre las combinaciones de hiperparámetros definidas en param_grid_part1. El modelo se entrena con el conjunto de datos de entrenamiento (training_set) durante 30 épocas, y la métrica de puntuación es la puntuación F1 (definida por 'scoring='f1'). Además, se utiliza early stopping durante el entrenamiento para evitar el sobreajuste. Al finalizar la búsqueda de cuadrícula, se obtiene el mejor modelo (best_model_part1) y los mejores parámetros (best_params_part1) basados en la puntuación F1 (Ver Tabla 11).

Tabla 11. Etapa 1 para valores utilizados en la búsqueda de hiperparámetros para el modelo propio

Parametros	Valores
Num_conv_layers	[1,2,3]
Num_filters	[32,64]
Kernel_size	[(3,3)], [(5,5)]
Pool_size	[(2,2)], [(3,3)]
Strides	[1,2]

Fuente: Elaboración propia

Etapa 2

Se especifican hiperparámetros adicionales para la búsqueda, centrados en las capas densas del modelo. Incluye opciones para el número de capas densas, el número de unidades (neuronas) en cada capa, el tamaño del kernel y el optimizador. Se configura el mejor modelo obtenido en la Parte 1 de la búsqueda de hiperparámetros (best_model_part1) con los mejores parámetros encontrados en esa parte (Ver Tabla 12).

Tabla 12. Etapa 2 para valores utilizados en la búsqueda de hiperparámetros para el modelo propio

Parametros	Valores
Num_dense_layers	[2,3]
Num_units	[64,128,256]
optimizer	[Adam], [RMsprop]

Fuente: Elaboración propia

Etapa 3

Se especifican hiperparámetros adicionales para la búsqueda, centrados en la tasa de abandono y la tasa de aprendizaje. Se prueban diferentes valores de tasa de abandono y tasas de aprendizaje para optimizar el rendimiento del modelo. Se configura el mejor modelo obtenido en la Parte 2 de la búsqueda de hiperparámetros (best_model_part2) con los mejores parámetros encontrados en esa parte (Ver Tabla 13).

Tabla 13. Etapa 3 para valores utilizados en la búsqueda de hiperparámetros para el modelo propio

Parametros	Valores
dropout	[0.25, 0.5, 0.8]
Learning_rate	[0.01, 0.001]

Fuente: Elaboración propia

El proceso de identificación y ajuste de hiperparámetros desempeña un papel crucial en el perfeccionamiento de la precisión y eficiencia de nuestras Redes Neuronales Convolucionales (CNN). A pesar de que el costo computacional asociado puede ser significativo, representa una inversión imprescindible para alcanzar predicciones más precisas y confiables. En este contexto, la presente red fue entrenada utilizando los hiperparámetros óptimos, los cuales se detallan en la Tabla 14 para proporcionar una visión clara y transparente de la configuración adoptada.

Tabla 14. Hiperparámetros utilizados para el modelo propio

Hiperparámetros	Nombre	Valores probados
num_conv_layers	Número de capas convolucionales	3
num_filters	Numero de filtros por capa	64
kernel_size	Tamaño del kernel	(5,5)
pool_size	Tamaño del pooling	(2,2)
strides	Strides	2
num_dense_layers	Numero de capas densas	2
num_units	Número de unidades por capa densa	128
optimizer	Optimizador	Adam
Dropout	Dropout	0.25
learning_rate	Tasa de aprendizaje	0.01

Fuente: Elaboración propia

VGG16 utilizando Grid search

Dentro del ámbito de esta red neuronal, hemos desarrollado una arquitectura que se vale de la exploración aleatoria de hiperparámetros dentro de un espacio predefinido. Como resultado de este enfoque, los parámetros óptimos para la red se detallan en la Tabla 15. Este enfoque no solo nos proporciona una configuración precisa, sino que aumenta el rendimiento de la red a niveles superiores.

Tabla 15. Hiperparámetros utilizados para el modelo VGG16

Parámetro	Valor
Num_dense_layers	2
Num_units	128

Fuente: Elaboración propia

VGG19 utilizando Grid search

Esta arquitectura, basada en la VGG19 preen trenada, ha facilitado la identificación de los hiperparámetros óptimos para lograr predicciones altamente precisas. Además, hemos incorporado hiperparámetros adicionales a la red, como se ilustra en la Figura 16, ampliando así su capacidad y perfeccionando su rendimiento.

Tabla 16. Hiperparámetros utilizados para el modelo VGG19

Parámetro	Valor
Num_dense_layers	2
Num_units	128

Fuente: Elaboración propia

Red-Net50 utilizando Grid search

La arquitectura propuesta, fundamentada en la preentrenada Red-Net50, ha sido instrumental en la determinación de hiperparámetros óptimos, culminando en predicciones de alta precisión. Además, se ha enriquecido la red mediante la incorporación de hiperparámetros adicionales, detallados en la Figura 17, lo que ha ampliado su capacidad y perfeccionado de manera notable su rendimiento.

Tabla 17. Hiperparámetros utilizados para el modelo RedNet50

Parámetro	Valor
Num_dense_layers	2
Num_units	128

Fuente: Elaboración propia

6. ANÁLISIS DE RESULTADOS

6.1. Evaluar los modelos con métricas propias del aprendizaje automático.

Exploramos y comparamos diferentes modelos de CNN para el procesamiento de imágenes. En el modelo propio de CNN de nuestro experimento, este modelo obtuvo un accuracy de 0.7294, una precisión de 0.7648, un recall de 0.7524 y un puntaje F1 de 0.7586. Aunque estos resultados son respetables, los modelos más avanzados tienden a superar al CNN base.

El modelo VGG16 es una arquitectura de CNN que se utiliza comúnmente para el procesamiento de imágenes. Al aplicar la técnica de Transfer Learning, pudimos mejorar notablemente la precisión y eficiencia del modelo. En nuestro experimento, el modelo VGG16 con Transfer Learning obtuvo una precisión de 0.9053, una precisión de 0.9201, un recall de 0.8937 y un puntaje F1 de 0.9011.

Similar al VGG16, el VGG19 es otra arquitectura de CNN. Aunque los resultados del VGG19 con Transfer Learning fueron ligeramente inferiores a los del VGG16, aún superaron al modelo propio de CNN. En nuestro experimento, el modelo VGG19 con Transfer Learning obtuvo una precisión de 0.8710, una precisión de 0.8241, un recall de 0.8014 y un puntaje F1 de 0.8311.

Finalmente, el modelo ResNet50 es una arquitectura de CNN que se utiliza para tareas de procesamiento de imágenes. En nuestro experimento, el modelo ResNet50 con Transfer Learning obtuvo una precisión de 0.8595, una precisión de 0.8894, un recall de 0.8409 y un puntaje F1 de 0.8499. En general, los modelos con Transfer Learning (VGG16, VGG19, ResNet50) tienden a tener un rendimiento superior en comparación con el modelo propio de CNN (Ver Tabla 18). Sin embargo, también es importante tener en cuenta otros factores como la complejidad del modelo, el tiempo de entrenamiento y los recursos computacionales disponibles (Ver Figura 19).

Tabla 18. Resultados de las métricas de los modelos CNN definiendo los hiperparámetros

Modelos	Accuracy	Precision	Recall	F1 Score
Modelo propio	0.7294	0.7648	0.7524	0.7586
VGG16 con Transfer Learning	0.9053	0.9201	0.8937	0.9011
VGG19 con Transfer Learning	0.8710	0.8241	0.8014	0.8311
ResNet50 con Transfer Learning	0.8595	0.8894	0.8409	0.8499

Fuente: Elaboración propia

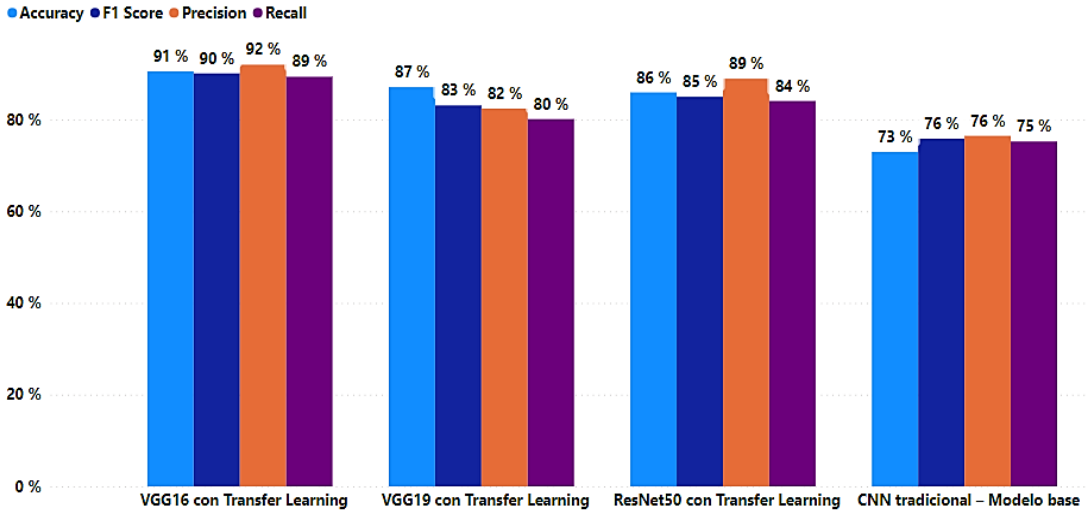


Figura 19. Métricas de evaluación para los modelos CNN definiendo los hiperparámetros
Fuente: Elaboración propia

En la Tabla 19, se evidencia que, a pesar de que el Modelo Propio exhibe el rendimiento más destacado en el conjunto de entrenamiento, el modelo VGG16 con Transfer Learning supera a los demás en los conjuntos de validación y prueba. Este resultado sugiere que el modelo VGG16 podría ser el más robusto y generalizable entre las cuatro configuraciones evaluadas.

- **Modelo Propio:** Este modelo tiene el rendimiento más alto en el conjunto de entrenamiento con un AUC de 0.9893 y en el conjunto de validación con un AUC de

0.7704. Sin embargo, su rendimiento en el conjunto de prueba es menor con un AUC de 0.8236.

- **VGG16 con Transfer Learning:** Este modelo tiene un rendimiento ligeramente menor en el conjunto de entrenamiento con un AUC de 0.9800 en comparación con el Modelo Propio. Sin embargo, supera al Modelo Propio tanto en el conjunto de validación con un AUC de 0.8721 como en el conjunto de prueba con un AUC de 0.8597.

- **VGG19 con Transfer Learning:** Este modelo tiene un rendimiento inferior en el conjunto de entrenamiento con un AUC de 0.9304. Aunque mejora en el conjunto de validación con un AUC de 0.8601, no logra superar al VGG16 en el conjunto de prueba.

- **ResNet50 con Transfer Learning:** Este modelo tiene las métricas AUC más bajas entre todos los modelos listados en el conjunto de entrenamiento, validación y prueba.

Tabla 19. Métricas de evaluación AUC para modelos CNN definiendo los hiperparámetros

Modelos	Training	Validation	Test
Modelo propio	0.9893	0.7704	0.8236
VGG16 con Transfer Learning	0.9800	0.8721	0.8597
VGG19 con Transfer Learning	0.9304	0.8601	0.8745
ResNet50 con Transfer Learning	0.8157	0.7949	0.8163

Fuente: Elaboración propia

A continuación, en la Tabla 20 se presentan los resultados de los cuatro experimentos de CNN para las métricas de evaluación, el tiempo de duración, el número de épocas y la ROC curva.

Tabla 20. Características de los modelos de CNN definiendo los hiperparámetros

Números	Modelo propio	VGG16 con Transfer Learning	VGG19 con Transfer Learning	ResNet50 con Transfer Learning
Duración	3h 57m 34s	3h 50m 4s	3h 32m 15s	4h 30m 50s
Épocas	40	30	30	30
ROC curve	0.88	0.96	0.83	0.95

Fuente: Elaboración propia

La Tabla 21 presenta un resumen de los resultados obtenidos por los modelos que emplean la técnica de búsqueda de rejilla (grid search) para la optimización de hiperparámetros. Se detallan los valores resultantes para cada uno de los parámetros evaluados.

En el modelo propio donde se utiliza grid search, es eficiente y efectivo para tareas sencillas, pero puede no ser suficiente para tareas más complejas. Este modelo obtuvo una precisión (accuracy) de 0.8870, una precisión (precision) de 0.9012, un recall de 0.8900 y un F1 Score de 0.9075. En el modelo VGG16 es conocido por su arquitectura profunda y su capacidad para manejar imágenes más complejas. Tiene una mayor precisión en comparación con las CNN tradicionales, pero también requiere más recursos computacionales. El modelo obtuvo una precisión (accuracy) de 0.9120, una precisión (precision) de 0.9225, un recall de 0.9024 y un F1 Score de 0.9187. El modelo VGG19 Similar a VGG16 pero aún más profundo, ofreciendo una precisión aún mayor a costa de una mayor necesidad de recursos computacionales y tiempo de entrenamiento (Ver Figura 20).

Tabla 21. Resultados de las métricas de los modelos CNN utilizando grid search

Modelos	Accuracy	Precision	Recall	F1 Score
Modelo propio	0.8870	0.9012	0.8900	0.9075
VGG16 grid search	0.9120	0.9225	0.9024	0.9187
VGG19 grid search	0.8614	0.8240	0.8413	0.8217
ResNet50 grid search	0.8500	0.8711	0.8317	0.8412

Fuente: Elaboración propia

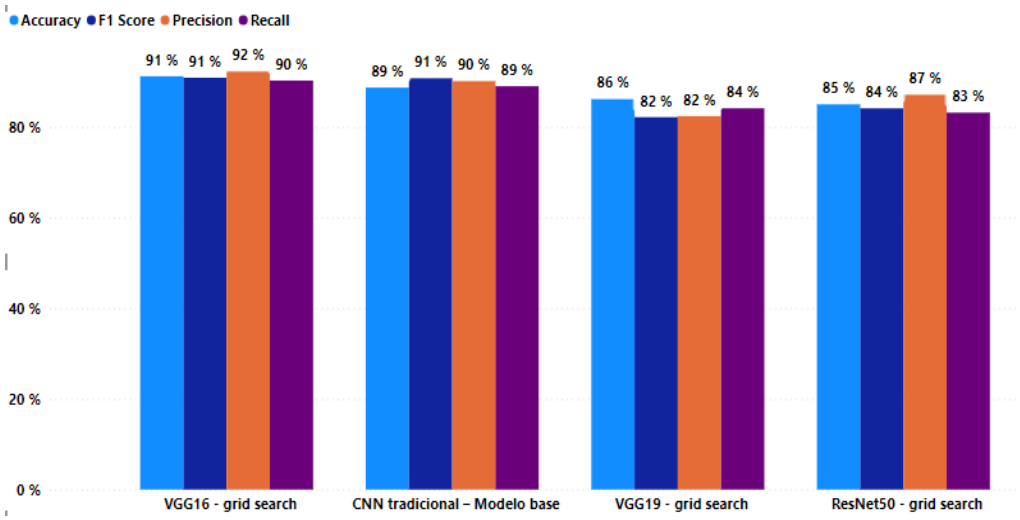


Figura 20. Análisis descriptivo para los modelos CNN utilizando grid search
Fuente: Elaboración propia

En la Tabla 22, se destaca que, a pesar de que el Modelo Propio muestra un rendimiento sólido en los conjuntos de entrenamiento y validación, el modelo VGG16 con Transfer Learning supera a los demás en los conjuntos de entrenamiento y prueba. Este hallazgo sugiere que el modelo VGG16 podría ser la opción más robusta y generalizable entre las cuatro configuraciones analizadas.

- **Modelo Propio:** Este modelo tiene un rendimiento alto en el conjunto de entrenamiento con un AUC de 0.9712 y en el conjunto de validación con un AUC de 0.8715. Su rendimiento en el conjunto de prueba es bastante bueno con un AUC de 0.9103.

- **VGG16 con Transfer Learning:** Este modelo tiene un rendimiento superior en el conjunto de entrenamiento con un AUC de 0.9819 y en el conjunto de prueba con un AUC de 0.8998, pero un poco menor en el conjunto de validación con un AUC de 0.8634.

- **VGG19 con Transfer Learning:** Este modelo tiene un rendimiento inferior en el conjunto de entrenamiento con un AUC de 0.9240, en el conjunto de validación con un AUC de 0.8314 y en el conjunto de prueba con un AUC de 0.8267.

- **ResNet50 con Transfer Learning:** Este modelo presenta las métricas más bajas entre los cuatro modelos: 0.8568 en entrenamiento, 0.8046 en validación y 0.8843 en prueba.

Tabla 22. Métricas de evaluación AUC para modelos CNN definiendo los hiperparámetros con grid search

Modelos	Training	Validation	Test
Modelo propio	0.9712	0.8715	0.9103
VGG16 con Transfer Learning	0.9819	0.8634	0.8998
VGG19 con Transfer Learning	0.9240	0.8314	0.8267
ResNet50 con Transfer Learning	0.8568	0.8046	0.8843

Fuente: Elaboración propia

En la Tabla 23 se presentan los resultados de los cuatro experimentos de CNN para las métricas de evaluación, el tiempo de duración, el número de épocas y la ROC curva.

Tabla 23. Características de los experimentos de CNN utilizando grid search

Números	Modelo propio	VGG16 con Transfer Learning	VGG19 con Transfer Learning	ResNet50 con Transfer Learning
Duración	3h 57m 34s	3h 50m 4s	3h 32m 15s	4h 30m 50s
Épocas	40	30	30	30
ROC curve	0.91	0.92	0.82	0.93

Fuente: Elaboración propia

Los cuatro modelos evaluados exhiben comportamientos bastante similares, evidenciando una mejora significativa en comparación con los modelos de CNN. Uno de los modelos que mejor se adapta a las imágenes es el basado en VGG16, optimizado mediante el método de búsqueda en rejilla (grid search), que se presenta como una opción eficaz para la predicción y clasificación de pacientes como cura o falla.

6.2. Proponer un modelo computacional que permita predecir el desenlace terapéutico

En el ámbito del procesamiento de imágenes, este estudio propone un enfoque innovador basado en la búsqueda aleatoria (Random Search). Este modelo se destaca por su asertividad y precisión, ofreciendo resultados superiores a los métodos convencionales. El modelo propuesto es único en su tipo, ya que se desarrolló específicamente para el procesamiento de un conjunto de imágenes únicas. Estas imágenes presentaban desafíos particulares en términos de color, nitidez y toma de fotografías, factores que fueron fundamentales para mejorar el resultado final.

6.2.1. Discusión

Después de llevar a cabo la validación de los ocho experimentos y analizar diversas métricas, incluyendo el comportamiento en los conjuntos de entrenamiento, validación y prueba, queda evidente que tanto el modelo VGG16 como el modelo propio, optimizados mediante la técnica de búsqueda en rejilla (grid search), destacan. La elección de estos modelos se respalda en sus resultados superiores en los conjuntos de validación y prueba. En la Figura 21, se presenta la comparación del Área bajo la Curva ROC para el modelo propio (en la figura izquierda) y el modelo VGG16 (en la figura derecha).

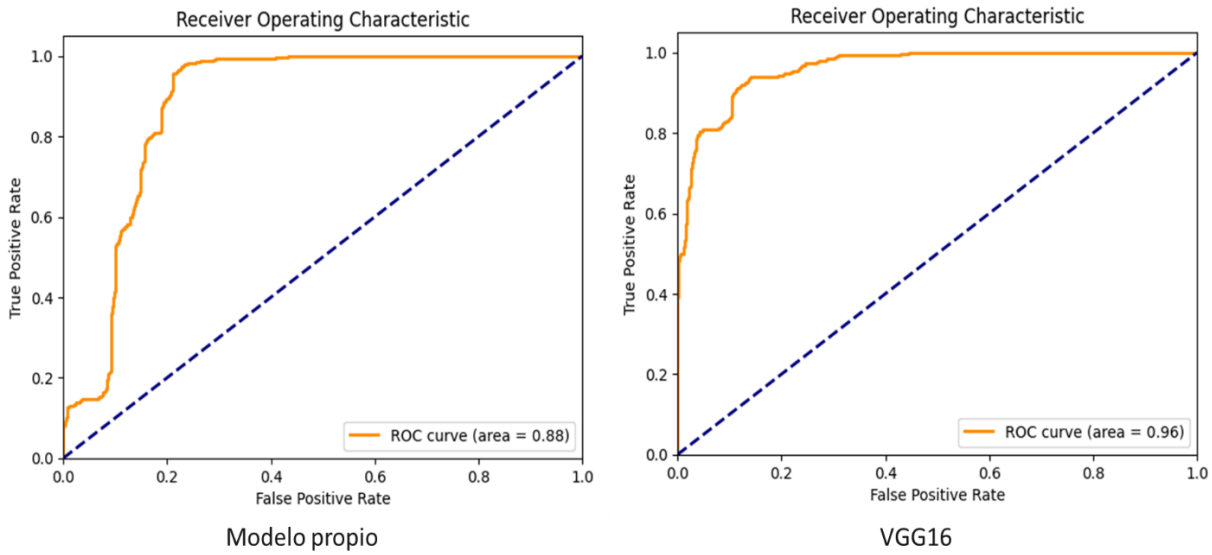


Figura 21. Evaluación de la curva ROC del modelo propio y VGG16
Fuente: Elaboración propia

Evaluando los resultados de ambos modelos en la matriz de confusión se calcula las métricas de precisión, sensibilidad y F1-score, esta última será a partir de la cual se tomó la decisión de elegir el mejor modelo para clasificación, dado que ha sido diseñada para funcionar bien cuando se tienen distribuciones de clases desiguales, con un alto valor de precisión en la clase mayoritaria (negativos) y un bajo recall en la clase minoritaria (positivos). Ver Tabla 24.

Tabla 24. Métricas de la matriz de confusión.

Modelos	Accuracy	Precision	Recall	F1 Score
Modelo propio	0.8870	0.9012	0.8900	0.9075
VGG16 - grid search	0.9120	0.9225	0.9024	0.9187

Fuente: Elaboración propia

Aunque el rendimiento del modelo propio es sólido, el modelo VGG16–grid-search supera al modelo propio en todas las métricas evaluadas, sugiriendo que podría ser la opción más robusta y generalizable entre ambos modelos. La métrica del F1-score alcanzó un notable 91.87%, lo cual representa un resultado positivo. Este puntaje respalda la capacidad del modelo para lograr un equilibrio entre precisión y exhaustividad en la clasificación de los pacientes.

7. CONCLUSIONES Y TRABAJOS FUTUROS

7.1. CONCLUSIONES

- Durante el transcurso de este proyecto, se desarrollaron y validaron modelos de aprendizaje de máquina utilizando imágenes a color de las lesiones de la LC como entrada. En el proceso, se determinó que el modelo que mejor se adaptó a las métricas de evaluación fue el modelo VGG16 donde se utiliza Transfer Learning y grid search. Los resultados obtenidos en la matriz de confusión revelaron un F1-score del 91.87%, respaldado por una precisión del 91.20% y una sensibilidad del 90.24%. Estos resultados sustentan la eficacia y aplicabilidad de este enfoque de Machine Learning en el diagnóstico de lesiones cutáneas de la LC.

- En el desarrollo de esta investigación, se ha confirmado que los modelos de Transfer Learning exhiben una ventaja significativa en contraste con aquellos entrenados desde cero, como los modelos CNN base. Este hallazgo subraya la relevancia crucial de la transferencia de aprendizaje, especialmente en entornos donde la disponibilidad de datos es restringida. Este enfoque no solo ha demostrado ser eficaz en optimizar el rendimiento de los modelos, sino que también destaca su utilidad estratégica para superar desafíos asociados con conjuntos de datos limitados, consolidando así su papel fundamental en el avance de la investigación y aplicación práctica en diversos campos.

- En la evaluación de los modelos, se ha evidenciado que, si bien los enfoques basados en Transfer Learning demuestran un rendimiento destacado, la eficacia de una CNN experimenta una notable mejora al emplear la técnica de grid search para la optimización de los hiperparámetros. Aunque este método conlleva un costo computacional significativo, su aplicación se revela como una estrategia más eficiente para obtener modelos afinados y adaptados de manera precisa a las particularidades del conjunto de datos.

- Se abordó la desafiantemente baja cantidad de datos mediante la implementación de técnicas de DataAugmentation. Esta estrategia es un recurso efectivo para mitigar el sobreajuste, al tiempo que fortaleció la robustez del modelo, Además, la investigación enfatizó la importancia crucial de asegurar que las imágenes carezcan de cualquier elemento que pueda

impactar negativamente en el rendimiento del modelo. Se destacaron consideraciones como la eliminación de fotos oscuras, la presencia de objetos inusuales, la nitidez de las imágenes y el ángulo de captura fotográfica. Estas precauciones adicionales no solo contribuyeron a la optimización del modelo, sino que también subrayaron la necesidad de una selección cuidadosa y preparación meticulosa de los datos para lograr resultados más confiables y generalizables.

- La implementación de la técnica de emparejamiento de imágenes en esta investigación ha supuesto una mejora sustancial en el entrenamiento de las CNN. Al proporcionar al modelo conjuntos de imágenes que representan el estado antes y después de la evolución de la herida, este enfoque ha facilitado un entrenamiento más preciso y detallado de los modelos. Esta estrategia no solo ha enriquecido la capacidad predictiva de las CNN, sino que también ha posibilitado la captura más eficiente de las variaciones sutiles en la progresión de las heridas. En última instancia, esta innovadora aproximación ha contribuido a un nivel más profundo de comprensión y precisión en la interpretación de imágenes médicas.

- En relación con las limitaciones del proyecto, fue imperativo ajustar el cronograma de trabajo debido a que los tiempos de ejecución de los modelos resultaron considerablemente elevados, a pesar de haber sido implementados en una máquina virtual con una configuración robusta.

7.2. TRABAJOS FUTUROS

Como dirección para trabajos futuros, sería pertinente explorar alternativas de redes convolucionales adicionales, aparte de la que se empleó en esta investigación. Se podrían considerar modelos de la familia EfficientNet, así como de VGG o ResNet, por mencionar algunas opciones. Además, un aspecto clave para mejorar la precisión consiste en incrementar la cantidad de datos disponibles para el entrenamiento, procurando especialmente que la distribución de imágenes por cada clase sea más equitativa que la utilizada en este estudio.

8. ANEXOS

Anexo 1 Código para la preparación de las imágenes

```python
import os
import cv2
import random

# Rutas de entrada y salida
ruta_entrada = 'IMAGENES/FALLA/13597 - GLUCANTIME - FALLA/SEMANA 3 C6-F1'
ruta_salida = 'IMAGENES/FALLA/13597 - GLUCANTIME - FALLA/DATA23'

# Asegúrate de que la carpeta de salida existe
if not os.path.exists(ruta_salida):
    os.makedirs(ruta_salida)

# Lista de archivos en la carpeta de entrada
archivos = os.listdir(ruta_entrada)

# Función para redimensionar la imagen
def resize_image(imagen, width, height):
    return cv2.resize(imagen, (width, height), interpolation = cv2.INTER_AREA)

# Función para aplicar transformaciones aleatorias
def apply_random_transform(imagen):
    probabilidad = random.random()  # Generar un número aleatorio entre 0 y 1

    if probabilidad < 0.5:
        # Modificar el tono, saturación, contraste y brillo de manera aleatoria
        alpha = 1 + random.uniform(-0.3, 0.2)  # Modificar el contraste
        beta = 0 + random.uniform(-5, 3)     # Modificar el brillo

        imagen = cv2.convertScaleAbs(imagen, alpha=alpha, beta=beta)

        # Rotar la imagen a 0, 90, 180 o 270 grados de manera aleatoria
        angle = random.choice([0, 90, 180, 270])
        if angle != 0:
            imagen = cv2.rotate(imagen, cv2.ROTATE_90_CLOCKWISE if angle == 90 else cv2.ROTATE_180 if angle == 180 else cv2.ROTA

        # Aplicar desenfoque gaussiano con sigma aleatorio
        sigma = random.uniform(0.5, 1.5)
        imagen = cv2.GaussianBlur(imagen, (5, 5), sigmaX=sigma)

    return imagen

# Número de imágenes generadas por cada imagen de entrada
num_imagenes_generadas = 102

# Iterar a través de los archivos y aplicar transformaciones
for archivo in archivos:
    ruta_completa_entrada = os.path.join(ruta_entrada, archivo)

    imagen_original = cv2.imread(ruta_completa_entrada)

    # Redimensiona la imagen original a las dimensiones deseadas
    imagen_original = resize_image(imagen_original, width=250, height=250)

    for i in range(num_imagenes_generadas):
        imagen_transformada = apply_random_transform(imagen_original)


        # Generar un nombre de archivo único para cada imagen transformada
        nombre_salida = f"{os.path.splitext(archivo)[0]}_transformada_{i}{os.path.splitext(archivo)[1]}"
        ruta_completa_salida = os.path.join(ruta_salida, nombre_salida)

        cv2.imwrite(ruta_completa_salida, imagen_transformada)

print(f"Transformaciones completadas. Se han generado {len(archivos) * num_imagenes_generadas} imágenes en total.")
```

Fuente: Elaboración propia

Anexo 2. Código para unir las imágenes

```python
import os
from PIL import Image

# Rutas de las carpetas que contienen las imágenes
carpeta_dato1 = "IMAGENES/FALLA/13723 - GLUCANTIME - FALLA/DATA12"
carpeta_dato2 = "IMAGENES/FALLA/13723 - GLUCANTIME - FALLA/DATA15"
carpeta_destino = "IMAGENES/FALLA/13723 - GLUCANTIME - FALLA/MAC30"

# Crea la carpeta donde se guardarán las imágenes unidas
os.makedirs(carpeta_destino, exist_ok=True)

# Obtiene la lista de nombres de archivos en cada carpeta
archivos_dato1 = os.listdir(carpeta_dato1)
archivos_dato2 = os.listdir(carpeta_dato2)

# Asegúrate de que ambas carpetas tienen la misma cantidad de imágenes
assert len(archivos_dato1) == len(archivos_dato2), "Las carpetas deben tener la misma cantidad de imágenes"

# Recorre las imágenes en parejas
for archivo1, archivo2 in zip(archivos_dato1, archivos_dato2):
    # Abre las imágenes
    imagen1 = Image.open(os.path.join(carpeta_dato1, archivo1))
    imagen2 = Image.open(os.path.join(carpeta_dato2, archivo2))

    # Obtiene las dimensiones de las imágenes
    ancho1, alto1 = imagen1.size
    ancho2, alto2 = imagen2.size

    # Crea una nueva imagen con el ancho combinado y el mayor alto
    nueva_imagen = Image.new('RGB', (ancho1 + ancho2, max(alto1, alto2)))

    # Pega las imágenes en la nueva imagen
    nueva_imagen.paste(imagen1, (0, 0))
    nueva_imagen.paste(imagen2, (ancho1, 0))

    # Guarda la nueva imagen en la carpeta_destino
    nueva_imagen.save(os.path.join(carpeta_destino, f"{archivo1}_unida.jpg"))
```

Fuente: Elaboración propia

9. REFERENCIAS BIBLIOGRÁFICAS

[1] M. Azim, S. A. Khan, S. Ullah, S. Ullah, y S. I. Anjum, «Therapeutic advances in the topical treatment of cutaneous leishmaniasis: A review», *PLoS Negl. Trop. Dis.*, vol. 15, n.º 3, p. e0009099, mar. 2021, doi: 10.1371/journal.pntd.0009099.

[2] Ministerio de salud de Colombia, «Leishmaniasis», 2022.

[3] H. J. C. de Vries y H. D. Schallig, «Cutaneous Leishmaniasis: A 2022 Updated Narrative Review into Diagnosis and Management Developments», *Am. J. Clin. Dermatol.*, vol. 23, n.º 6, pp. 823-840, nov. 2022, doi: 10.1007/s40257-022-00726-8.

[4] V. U. kumar *et al.*, «The Possible Role of Selected Vitamins and Minerals in the Therapeutic Outcomes of Leishmaniasis», *Biol. Trace Elem. Res.*, vol. 201, n.º 4, pp. 1672-1688, abr. 2023, doi: 10.1007/s12011-022-03311-6.

[5] M. del M. Castro *et al.*, «Cutaneous leishmaniasis treatment and therapeutic outcomes in special populations: A collaborative retrospective study», *PLoS Negl. Trop. Dis.*, vol. 17, n.º 1, p. e0011029, ene. 2023, doi: 10.1371/journal.pntd.0011029.

[6] L. Pinto-García, «Poisonously single-minded: public health implications of the pharmaceuticalization of leishmaniasis in Colombia», *Crit. Public Health*, vol. 32, n.º 5, pp. 619-629, oct. 2022, doi: 10.1080/09581596.2021.1918640.

[7] S. Trejos y S. Bermúdez, «Leishmaniasis cutánea en Colombia: una mirada epidemiológica», *Rev. EUGENIO ESPEJO*, vol. 17, n.º 1, pp. 5-7, dic. 2022, doi: 10.37135/ee.04.16.02.

[8] D. C. Torres, M. Ribeiro-Alves, G. A. S. Romero, A. M. R. Dávila, y E. Cupolillo, «Assessment of drug resistance related genes as candidate markers for treatment outcome prediction of cutaneous leishmaniasis in Brazil», *Acta Trop.*, vol. 126, n.º 2, pp. 132-141, may 2013, doi: 10.1016/j.actatropica.2013.02.002.

[9] A. Saha, S. Roy, y A. Ukil, «Cytokines and Signaling Networks Regulating Disease Outcomes in Leishmaniasis», *Infect. Immun.*, vol. 90, n.º 8, pp. e00248-22, ago. 2022, doi: 10.1128/iai.00248-22.

[10] M. Bamorovat *et al.*, «A novel diagnostic and prognostic approach for unresponsive patients with anthroponotic cutaneous leishmaniasis using artificial neural networks», *PLOS ONE*, vol. 16, n.º 5, p. e0250904, may 2021, doi: 10.1371/journal.pone.0250904.

[11] K. Thomsen, L. Iversen, T. L. Titlestad, y O. Winther, «Systematic review of machine learning for diagnosis and prognosis in dermatology», *J. Dermatol. Treat.*, vol. 31, n.º 5, pp. 496-510, jul. 2020, doi: 10.1080/09546634.2019.1682500.

[12] B. Eapen, «Artificial intelligence in dermatology: A practical introduction to a paradigm shift», *Indian Dermatol. Online J.*, vol. 11, n.º 6, p. 881, 2020, doi: 10.4103/idoj.IDOJ_388_20.

[13] Y. Lau, J. M. Gutierrez, M. Volkovs, y S. Zuberi, «Drug repurposing for Leishmaniasis with Hyperbolic Graph Neural Networks», Bioinformatics, preprint, feb. 2023. doi:

10.1101/2023.02.11.528117.
[14] S. Kumar, T. Arif, A. S. Alotaibi, M. B. Malik, y J. Manhas, «Advances Towards Automatic Detection and Classification of Parasites Microscopic Images Using Deep Convolutional Neural Network: Methods, Models and Research Directions», *Arch. Comput. Methods Eng.*, dic. 2022, doi: 10.1007/s11831-022-09858-w.
[15] A. N. S. Maia-Elkhoury *et al.*, «Interacción entre los determinantes medioambientales y socioeconómicos para el riesgo para leishmaniasis cutánea en América Latina», *Rev. Panam. Salud Pública*, vol. 45, p. 1, abr. 2021, doi: 10.26633/RPSP.2021.49.
[16] L. C. Yañez, H. A. T. Martín, I. R. M. Yañez, y M. S. Rodríguez, «Leishmaniasis visceral. Presentación de un caso y revisión de la entidade», *MULTIMED*, vol. 27, n.º 0, Art. n.º 0, feb. 2023.
[17] J. S. Serna-Trejos y S. G. Bermúdez-Moyano, «Caracterización epidemiológica de la leishmaniasis cutánea en Colombia, 2022», *Metro Cienc.*, vol. 30, n.º 3, Art. n.º 3, oct. 2022, doi: 10.47464/MetroCiencia/vol30/3/2022/85-88.
[18] R. Pearson, «Leishmaniasis (leishmaniosis) - Infecciones», Manual MSD versión para público general. Accedido: 4 de abril de 2023. [En línea]. Disponible en: https://www.msdmanuals.com/es-co/hogar/infecciones/infecciones-parasitarias-protozoos-extraintestinales/leishmaniasis-leishmaniosis
[19] J. Alvar, S. Yactayo, y C. Bern, «Leishmaniasis and poverty», *Trends Parasitol.*, vol. 22, n.º 12, pp. 552-557, dic. 2006, doi: 10.1016/j.pt.2006.09.004.
[20] «Boletín epidemiologico 2022». Accedido: 4 de abril de 2023. [En línea]. Disponible en: https://www.ins.gov.co/buscador-eventos/BoletinEpidemiologico/2022_Bolet%C3%ADn_epidemiologico_semana_25.pdf
[21] V. G. Ramírez, M. A. B. Garcia, y C. A. S. Limas, «Guía para la atención clínica integral del paciente con leishmaniasis», 2021.
[22] *Manual de procedimientos para vigilancia y control de las leishmaniasis en las Américas*. Organización Panamericana de la Salud, 2019. doi: 10.37774/9789275320631.
[23] D. M. Pigott *et al.*, «Global distribution maps of the leishmaniases», *eLife*, vol. 3, p. e02851, jun. 2014, doi: 10.7554/eLife.02851.
[24] T. del Rosal Rabes, F. Baquero-Artigao, C. Gómez Fernández, M. J. García Miguel, y R. de Lucas Laguna, «Tratamiento de la leishmaniasis cutánea con anfotericina B liposomal», *An. Pediatría*, vol. 73, n.º 2, pp. 101-102, ago. 2010, doi: 10.1016/j.anpedi.2010.05.014.
[25] I. Abadías-Granado, A. Diago, P. A. Cerro, A. M. Palma-Ruiz, y Y. Gilaberte, «Leishmaniasis cutánea y mucocutánea», *Actas Dermo-Sifiliográficas*, vol. 112, n.º 7, pp. 601-618, jul. 2021, doi: 10.1016/j.ad.2021.02.008.
[26] CDC, «CDC - Leishmaniasis». Accedido: 4 de abril de 2023. [En línea]. Disponible en: https://www.cdc.gov/parasites/leishmaniasis/biology.html
[27] L. Rouhiainen, *Inteligencia artificial: 101 cosas que debes saber hoy sobre nuestro futuro*.

Alienta Editorial, 2018.

[28] R. M. Aguilar, J. M. Torres, y C. A. Martín, «Aprendizaje Automático en la Identificación de Sistemas. Un Caso de Estudio en la Predicción de la Generación Eléctrica de un Parque Eólico», *Rev. Iberoam. Automática E Informática Ind.*, vol. 16, n.º 1, p. 114, dic. 2018, doi: 10.4995/riai.2018.9421.

[29] D. Conway y J. White, *Machine Learning for Hackers*. 2012. Accedido: 4 de abril de 2023. [En línea]. Disponible en: https://www.oreilly.com/library/view/machine-learning-for/9781449330514/

[30] K. Ayyadevara, *Neural Networks with Keras Cookbook*. 2019. Accedido: 4 de abril de 2023. [En línea]. Disponible en: https://www.oreilly.com/library/view/neural-networks-with/9781789346640/

[31] R. Yamashita, M. Nishio, R. K. G. Do, y K. Togashi, «Convolutional neural networks: an overview and application in radiology», *Insights Imaging*, vol. 9, n.º 4, pp. 611-629, ago. 2018, doi: 10.1007/s13244-018-0639-9.

[32] H. R. Roth *et al.*, «Efficient False Positive Reduction in Computer-Aided Detection Using Convolutional Neural Networks and Random View Aggregation», en *Deep Learning and Convolutional Neural Networks for Medical Image Computing*, L. Lu, Y. Zheng, G. Carneiro, y L. Yang, Eds., en Advances in Computer Vision and Pattern Recognition. , Cham: Springer International Publishing, 2017, pp. 35-48. doi: 10.1007/978-3-319-42999-1_3.

[33] F. Liu y L. Yang, «A Novel Cell Detection Method Using Deep Convolutional Neural Network and Maximum-Weight Independent Set», en *Deep Learning and Convolutional Neural Networks for Medical Image Computing*, L. Lu, Y. Zheng, G. Carneiro, y L. Yang, Eds., en Advances in Computer Vision and Pattern Recognition. , Cham: Springer International Publishing, 2017, pp. 63-72. doi: 10.1007/978-3-319-42999-1_5.

[34] G. F. S. Silva, T. P. Fagundes, B. C. Teixeira, y A. D. P. Chiavegatto Filho, «Machine Learning for Hypertension Prediction: a Systematic Review», *Curr. Hypertens. Rep.*, vol. 24, n.º 11, pp. 523-533, nov. 2022, doi: 10.1007/s11906-022-01212-6.

[35] A. Falor, M. Hirani, H. Vedant, P. Mehta, y D. Krishnan, «A Deep Learning Approach for Detection of SQL Injection Attacks Using Convolutional Neural Networks», en *Proceedings of Data Analytics and Management*, vol. 91, D. Gupta, Z. Polkowski, A. Khanna, S. Bhattacharyya, y O. Castillo, Eds., en Lecture Notes on Data Engineering and Communications Technologies, vol. 91. , Singapore: Springer Singapore, 2022, pp. 293-304. doi: 10.1007/978-981-16-6285-0_24.

[36] J. A. Martínez Pérez y P. S. Pérez Martin, «La curva ROC», *Med. Fam. SEMERGEN*, vol. 49, n.º 1, p. 101821, ene. 2023, doi: 10.1016/j.semerg.2022.101821.

[37] A. Zapeta Hernández, G. A. Galindo Rosales, H. J. Juan Santiago, y M. Martínez Lee, «Métricas de rendimiento para evaluar el aprendizaje automático en la clasificación de imágenes petroleras utilizando redes neuronales convolucionales», *Cienc. Lat. Rev. Científica*

Multidiscip., vol. 6, n.º 5, pp. 4624-4637, nov. 2022, doi: 10.37811/cl_rcm.v6i5.3420.

[38] D. S. Comas, A. Amalfitano, G. J. Meschino, y V. L. Ballarin, «Interpretación y visualización de características en texturas mediante Redes Neuronales Convolucionales», en *2022 IEEE Biennial Congress of Argentina (ARGENCON)*, San Juan, Argentina: IEEE, sep. 2022, pp. 1-8. doi: 10.1109/ARGENCON55245.2022.9939876.

[39] N. Steyve, P. Steve, M. Ghislain, S. Ndjakomo, y E. pierre, «Optimized real-time diagnosis of neglected tropical diseases by automatic recognition of skin lesions», *Inform. Med. Unlocked*, vol. 33, p. 101078, 2022, doi: 10.1016/j.imu.2022.101078.

[40] M. Górriz, A. Aparicio, B. Raventós, V. Vilaplana, E. Sayrol, y D. López-Codina, «Leishmaniasis Parasite Segmentation and Classification Using Deep Learning», en *Articulated Motion and Deformable Objects*, vol. 10945, F. J. Perales y J. Kittler, Eds., en Lecture Notes in Computer Science, vol. 10945. , Cham: Springer International Publishing, 2018, pp. 53-62. doi: 10.1007/978-3-319-94544-6_6.

[41] M. Zare *et al.*, «A machine learning-based system for detecting leishmaniasis in microscopic images», *BMC Infect. Dis.*, vol. 22, n.º 1, p. 48, dic. 2022, doi: 10.1186/s12879-022-07029-7.

[42] Y. Wu, Y. Yang, H. Nishiura, y M. Saitoh, «Deep Learning for Epidemiological Predictions», en *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, Ann Arbor MI USA: ACM, jun. 2018, pp. 1085-1088. doi: 10.1145/3209978.3210077.

[43] J. Galeano, A. Zarzycki, M. C. Torres-Madronero, L. Restrepo, J. Q. Pulgarin, y S. M. Robledo, «Characterization of Cutaneous Leishmaniasis Ulcers Utilizing a Multispectral Imaging System», en *2021 XXIII Symposium on Image, Signal Processing and Artificial Vision (STSIVA)*, Popayán, Colombia: IEEE, sep. 2021, pp. 1-5. doi: 10.1109/STSIVA53688.2021.9592002.

[44] R. J. King, D. H. Campbell-Lendrum, y C. R. Davies, «Predicting Geographic Variation in Cutaneous Leishmaniasis, Colombia», *Emerg. Infect. Dis.*, vol. 10, n.º 4, pp. 598-607, abr. 2004, doi: 10.3201/eid1004.030241.

[45] C. Ferro *et al.*, «Environmental Risk Factors for the Incidence of American Cutaneous Leishmaniasis in a Sub-Andean Zone of Colombia (Chaparral, Tolima)», *Am. J. Trop. Med. Hyg.*, vol. 82, n.º 2, pp. 243-250, feb. 2010, doi: 10.4269/ajtmh.2010.09-0218.

[46] M. Pérez-Flórez, C. B. Ocampo, C. Valderrama-Ardila, y N. Alexander, «Spatial modeling of cutaneous leishmaniasis in the Andean region of Colombia», *Mem. Inst. Oswaldo Cruz*, vol. 111, n.º 7, pp. 433-442, jun. 2016, doi: 10.1590/0074-02760160074.

[47] N. Shabanpour, S. V. Razavi-Termeh, A. Sadeghi-Niaraki, S.-M. Choi, y T. Abuhmed, «Integration of machine learning algorithms and GIS-based approaches to cutaneous leishmaniasis prevalence risk mapping», *Int. J. Appl. Earth Obs. Geoinformation*, vol. 112, p. 102854, ago. 2022, doi: 10.1016/j.jag.2022.102854.

[48] P. Olliaro *et al.*, «Methodology of Clinical Trials Aimed at Assessing Interventions for Cutaneous Leishmaniasis», *PLoS Negl. Trop. Dis.*, vol. 7, n.º 3, p. e2130, mar. 2013, doi:

10.1371/journal.pntd.0002130.
[49] Y. Nie, A. S. Zamzam, y A. Brandt, «Resampling and data augmentation for short-term PV output prediction based on an imbalanced sky images dataset using convolutional neural networks», *Sol. Energy*, vol. 224, pp. 341-354, ago. 2021, doi: 10.1016/j.solener.2021.05.095.
[50] Y. Zhu *et al.*, «Converting tabular data into images for deep learning with convolutional neural networks», *Sci. Rep.*, vol. 11, n.º 1, p. 11325, may 2021, doi: 10.1038/s41598-021-90923-y.
[51] J. N. Heidenreich, M. B. Gorji, y D. Mohr, «Modeling structure-property relationships with convolutional neural networks: Yield surface prediction based on microstructure images», *Int. J. Plast.*, vol. 163, p. 103506, abr. 2023, doi: 10.1016/j.ijplas.2022.103506.
[52] S. Andradóttir, «Chapter 20 An Overview of Simulation Optimization via Random Search», en *Handbooks in Operations Research and Management Science*, vol. 13, Elsevier, 2006, pp. 617-631. doi: 10.1016/S0927-0507(06)13020-0.
[53] Z. Rguibi, A. Hajami, D. Zitouni, A. Elqaraoui, y A. Bedraoui, «CXAI: Explaining Convolutional Neural Networks for Medical Imaging Diagnostic», *Electronics*, vol. 11, n.º 11, p. 1775, jun. 2022, doi: 10.3390/electronics11111775.
[54] R. Pandey, A. Sahai, y H. Kashyap, «Implementing convolutional neural network model for prediction in medical imaging», en *Artificial Intelligence and Machine Learning for EDGE Computing*, Elsevier, 2022, pp. 189-206. doi: 10.1016/B978-0-12-824054-0.00024-1.
[55] A. Altameem, C. Mahanty, R. C. Poonia, A. K. J. Saudagar, y R. Kumar, «Breast Cancer Detection in Mammography Images Using Deep Convolutional Neural Networks and Fuzzy Ensemble Modeling Techniques», *Diagnostics*, vol. 12, n.º 8, p. 1812, jul. 2022, doi: 10.3390/diagnostics12081812.
[56] J. M. H. Noothout *et al.*, «Knowledge distillation with ensembles of convolutional neural networks for medical image segmentation», *J. Med. Imaging*, vol. 9, n.º 05, may 2022, doi: 10.1117/1.JMI.9.5.052407.