UNIÓN DE ASOCIACIONES DE INGENIEROS TÉCNICOS INDUSTRIALES Y GRADUADOS EN INGENIERÍA DE LA RAMA INDUSTRIAL DE ESPAÑA (UAITIE)

"CONVOCATORIA 2021"

VI PREMIO NACIONAL DE INICIACIÓN A LA INVESTIGACIÓN TECNOLÓGICA

DIAGNÓSTICO DE MALARIA MEDIANTE EL USO DE REDES NEURONALES CONVOLUCIONALES

Autores:
Pilar Sierra Segovia
Laura Ortiz Cedeño
Lucía Zamacona Gómez

Informática, 2ºBachillerato

Tutor: Jorge Rastrollo Romero
Mayo, 2021

ÍNDICE

1. INTRODUCCIÓN ..... ¡Error! Marcador no definido.

2. OBJETIVOS 2

3. METODOLOGÍA ..... ¡Error! Marcador no definido.2

3.1 Inteligencia artificial ..... ¡Error! Marcador no definido.

. Red neuronal convolucional (CNN) 3

4. RESULTADOS 4

4.1 Herramientas empleadas ..... ¡Error! Marcador no definido.

4.2 Procedimientos seguidos 6

I. Instalación de los entornos de programación en el PC 6

II. Análisis del software y adaptación del código según los requerimientos del estudio. 6

III. Recogida y análisis de resultados.....8

5. CONCLUSIONES 15

6. BIBLIOGRAFÍA 17

7. TABLAS..... ¡Error! Marcador no definido.

TABLA I ..... ¡Error! Marcador no definido.

TABLA II ..... ¡Error! Marcador no definido.

TABLA III ..... ¡Error! Marcador no definido.

1. INTRODUCCIÓN-Resumen

La malaria es una de las enfermedades parasitarias que más impacto tiene en nuestro mundo, destacando en las áreas en vías de desarrollo como lo son Asia, América Latina o África; dentro de estas poblaciones las mujeres embarazadas junto con los niños son los grandes afectados.

Es transmitida por la picadura de un mosquito hembra del género *Anopheles*, el cual ha sido anteriormente infectado por el parásito transmisor de la malaria, el *Plasmodium*. Por otro lado, también es posible el contagio por transfusiones sanguíneas y de la madre al feto.

Algunos de los factores de riesgo determinantes para la aparición de la enfermedad son, en cuanto a las características ambientales de la zona, la temperatura, la humedad, las precipitaciones y el agua estancada; resultando más perjudicados los países de zonas tropicales y subtropicales, entre otras razones porque el frío acaba con el parásito. Otros factores que hay que tener en cuenta son la condición de vida, viviendas de la población y la construcción y calidad de la misma [1].

Esta enfermedad puede ser combatida de diversos modos, como la utilización de mosquiteras, insecticidas o antibióticos para paliar los síntomas. En muchas zonas el parásito (*Plasmodium falciparum*) se ha hecho inmune a estos insecticidas y antibióticos, dando lugar a la necesidad de la búsqueda de medidas alternativas para acabar con el mismo.

Este proyecto apuesta por soluciones basadas en la detección automática de estos parásitos en muestras de sangre que, a la larga, resultará el medio más eficiente en tiempo y capital.

Palabras clave

Red neuronal convolucional, malaria, deep learning.

2. OBJETIVOS

Este proyecto tiene como objetivo el análisis de la malaria y las distintas técnicas de detección automática. Se realizará una prueba de técnica de detección por análisis de imágenes de muestras de sangre de células infectadas y no infectadas.

El objetivo general del presente proyecto es:

- ✓ Contribuir a la mejora de los sistemas de reducción de enfermedades parasitaria como la malaria, asociado al objetivo de desarrollo sostenible número III.

Por otro lado, los objetivos específicos son:

- ✓ Estudio de la enfermedad y factores involucrados.
- ✓ Estudio de técnicas de detección automática de malaria.
- ✓ Prueba de técnicas de detección automática.
- ✓ A partir de una CNN (red neuronal convolucional), alcanzar unos valores suficientemente fiables para una posible aplicación con fines de diagnóstico médico.
- ✓ Búsqueda de aplicaciones de la CNN en otras patologías, como puede ser la del COVID-19.

3. METODOLOGÍA

3.1.1. Red neuronal convolucional (CNN)

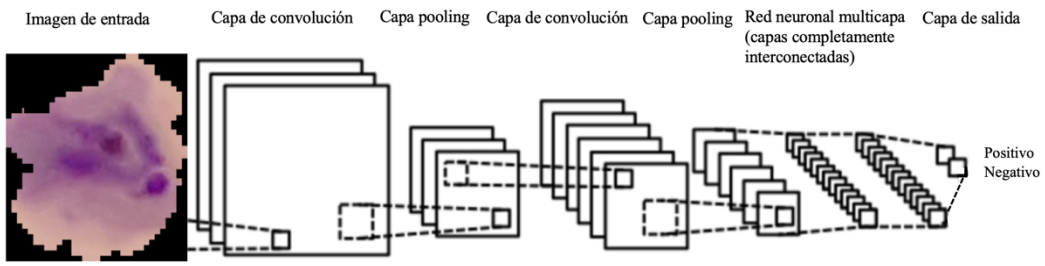

Figura 6: Arquitectura de una CNN.
Fuente: Elaboración propia.

Las redes neurales convolucionales, conocidas también como *Convolutional Neural Networks* (CNN) son una clase de redes neuronales que han favorecido enormemente en el campo del Deep learning al reconocimiento visual de patrones como pueden ser

imágenes y video. El objetivo de las CNN es la identificación de las características de las imágenes para la posterior clasificación de las imágenes en diferentes grupos [24].

La estructura de este tipo de redes es similar a la de las redes neuronales, la mayor diferencia se encuentra en que la entrada es una imagen representada como una matriz tridimensional, esto mismo ocurre con las capas de neuronas, organizadas asimismo de

un modo tridimensional. Las salidas son los diferentes grupos en los que se archivan las imágenes.

En las CNN hay cuatro tipos de capas diferenciados, la capa de entrada, la capa conocida como convolucional, *pooling* y capa de salida.

Las capas convolucionales se diferencian de las capas ocultas de las redes neuronales en que las neuronas de la misma no están completamente interconectadas, en vez de estos se conectan a ciertas neuronas de la capa precedente, la de entrada, con quien comparte el mismo número de pesos y bias, reduciendo el número de parámetros que contiene la CNN.

Estas neuronas conectan las dos primeras capas son las encargadas de calcular el producto entre la matriz de los pesos que comparten, al igual que el bias y el valor de los pixeles de esta franja que es compartida. La solución de esta operación es enviada a la función de activación, que en este tipo de redes neuronales suele ser la función *Rectified Linear Unit* (ReLU), la que calcula y expresa el máximo (0,x). Esta función viene determinada como (ReLU (x) = 0 si x = 0). El resultado final se expresa en unos mapas de activación que detectan las características que servirán para la final clasificación de las imágenes de entrada en grupos.

Estas características son definidas por el filtro o *kernel*, una mezcla de la matriz de pesos compartida y el valor de bias. Se conoce como convolución a los pixeles filtrados mediante el *kernel* [25].

4. RESULTADOS

Para la elaboración de este trabajo se emplearon diferentes fuentes de código de software libre a partir de redes neuronales convolucionales.

El primer experimento se ha basado en una iniciación en el manejo de *deep learning* y redes neuronales. Consistió en entender el funcionamiento de la detección automática de imágenes a través de un entorno online, sencillo y guiado (https://cutt.ly/1hJlo36).

El proyecto consistió en la introducción de imágenes en dos bloques, diferenciando entre parasitadas y libres de parásitos; para el posterior entrenamiento de la CNN. Cuando el entrenamiento concluyó, se realizaron una serie de pruebas para verificar la fiabilidad de la red. Una vez esta fue comprobada y se obtuvo un porcentaje elevado, se prosiguió a la

creación de un juego por medio de un entorno didáctico Scratch 3. Se creó un programa con bloques que permitiesen introducir nuevas imágenes, la posterior recogida de la etiqueta con la que reconoce el modelo de aprendizaje automático la imagen y un indicador de la fiabilidad del reconocimiento

Las limitaciones y sencillez del experimento derivaron en la exhaustiva búsqueda de otros códigos más complejos que permitiesen ser modificados, personificados, además de adecuarse a los objetivos del proyecto perseguidos.

Se llevó a cabo la búsqueda de un código abierto programado en Python que permitiese la clasificación de imágenes. Finalmente se seleccionaron dos de ellos, que se adaptaron para que funcionasen con las imágenes disponibles.

El primero código no se ajustaba completamente a los objeticos del proyecto ya que no consistía en la clasificación binaria de imágenes de malaria, sino una clasificación múltiple de deportes. Se extrajo de un artículo explicativo (https://cutt.ly/ihJlgcy) en el que guiadamente enseñaba a crear una CNN capaz de clasificar 70.000 imágenes de 10 deportes diferentes.

Independientemente de que las imágenes de las que se buscaba su clasificación eran imágenes parasitadas y no infectadas de malaria, para el estudio del funcionamiento del software fue necesaria la descarga de las imágenes empleadas en el código analizado. Estas fueron 77.000 imágenes de los diez deportes más populares del mundo: fútbol, baloncesto, golf, fútbol americano, tenis, Fórmula1, ciclismo, boxeo, beisball y natación. Se contaba con una media de entre 5000 y 9000 imágenes de cada deporte, de un tamaño 21×28 pixeles en color. Estas se descargan comprimidas y se descomprimía en los diez directorios según el tipo de deporte. Las imágenes se dividieron en sets de 80-20 destinadas al entrenamiento y test, respectivamente. Al mismo tiempo, las imágenes de entrenamiento se subdividen en 80-20 destinadas a entrenamiento y validación en cada EPOCH o época de aprendizaje. Para este mismo código se descargaron imágenes de malaria a través de la base de datos kaggle, que contenía un total de 27,558 imágenes divididas en dos directorios parasitized y uninfected. Para poder integrar estas imágenes en el código trabajado se tuvo que reducir el tamaño de las imágenes de modo que este coincidiese con el de las imágenes de deportes. Esto se realizó a partir de un programa ejecutado en Matlab.

Este código preparado para discernir entre varias categorías no dio resultados óptimos lo que conllevó a la experimentación con el segundo código previamente seleccionado.
Se trataba de un código binario, orientado a la clasificación de imágenes de muestras de malaria, diferenciando entre infectadas y no infectadas. Con este descubrimiento, se llegaba al objetivo final. El código de dominio público sobre el que se trabajó se extrajo de la página web https://cutt.ly/ehJzCy2.

En los siguientes apartados se detallan las herramientas empleadas y los procedimientos seguidos.

4.1 Procedimientos seguidos

I. Instalación de los entornos de programación en el PC

Fue necesaria la instalación de diferentes plataformas que posibilitasen la lectura y ejecución del código (con lenguaje interpretado Python), su posterior modificación y adaptación. Estos fueron, ya mencionados anteriormente, la Suite Anaconda 3 y el entorno Spyder.

II. Análisis del software y adaptación del código según los requerimientos del estudio.

1. Importar librerías

Para que el código pueda ejecutarse, el primer paso que hay que seguir es importar la librería. Las librerías o bibliotecas son archivos de código de a disposición del desarrollador de software, que se deben importar al principio del programa para poderlos usar y así facilitar el desarrollo de diversas funciones y solucionar problemas concretos de la app. Las librerías que se importan se obtuvieron desde el entorno Anaconda 3 y se trata de librerías *Keras* (biblioteca de Redes Neuronales de Código Abierto escrita en Python), específica para experimentos en redes neuronales.

2. Búsqueda, obtención y clasificación de imágenes

Las imágenes empleadas se descargaron de la base de datos kaggle, que contenía un total de 27,558 imágenes divididas en dos directorios parasitized y uninfected.

Se crean los diferentes directorios atendiendo a dos criterios, el tipo de imagen que van a contener y la función que se les va a dar. Las imágenes se organizan en los directorios

“A” y “B”, diferenciando entre las muestras infectadas (parasitized) y no infectadas (uninfected), respectivamente.

El siguiente paso consiste en renombrar las imágenes con los nombres de “pos” y “neg”, según perezcan al directorio A o B, respectivamente.

El propio código, gracias a la librería Shutil, define las rutas de imagen en sets de entrenamiento (3000), validación (1000) y test (probando).

La resolución de las imágenes fue ajustada a 150x150 pixeles (target_size= (150, 150)), esta vez el cambio lo realizó una librería, en concreto ImageDataGenerator, generada por Keras. Esta misma librería lleva a cabo la conversión del código a formato binario(class_mode='binary'). El class_mode se especifica como binario en el que las entradas de la red se separarán en dos carpetas de salida. Ante esto ImageDataGenerator será capaz de reconocer cada clase con la etiqueta correspondiente (“*pos”* o *“neg”*). Está librería también define la cantidad de muestras que se propagarán a través de la red (batch_size=20), divide los sets de entrenamiento en 20 lotes (batch) y se consigue una reducción de memoria.

3. Arquitectura de la red

En la propia red se ha utilizado un modelo secuencial compuesto por una serie de capas (layers) agregadas en secuencia. Hay Estas se han obtenido mediante la descarga de librerías de Keras.

Las capas convolcionales se encuentran interconectadas entre sí y son capaces de extraer información de cada imagen creando patrones de pixeles con características de cada región de la imagen. Dicha información será transmitida por las diferentes capas ocultas hasta llegar a la capa de salida, se alternan redes de convolución y redes poolings.

Las funciones de activación son las encargadas de devolver una salida partiendo de unos valores de entrada. Las funciones de activación empleadas fueron:

- ◊ Sigmoid: acota los valores de entrada en un rango de (0,1), para la obtención de un valor de salida binario. Así, como se observa en la gráfica, para los valores muy negativos de x la función tiende a 0 y para valores de x muy positivos, a 1.
- ◊ ReLU: esta función de activación no continua se encarga de que la capa de convoluvión identique un patrón en la imagen y transferirlo a la siguiente capa de convoluvión. La desventaja que presenta es que cuando las entradas son cero o negativas, el gradiente es cero. Solo pasan a la capa contigua los valores positivos [31].

4. Compilación

◊ Optimizador

El optimizador en las redes neuronales se encarga de optimizar los valores de los parámetros para reducir lo máximo posible el error de la CNN, esto se realiza mediante un proceso conocido como "*backpropagation*". Para el cálculo de este error se calcula la derivada de la matriz de parámetros, teniendo en cuenta que las CNN están compuestas por una matriz de pesos. En esta red se utilizó el optimizador RMSprop (optimizer=optimizers.RMSprop (lr=10^{-x})), descargado desde Keras, mantiene un promedio móvil del cuadrado de los gradientes de manera que evita la acumulación de errores y disminuye la posibilidad de fallo.

◊ Loss y accuraccy

Loss es la suma de los errores cometidos para cada ejemplo en los conjuntos de entrenamiento o validación. La función de pérdida muestra el desfase que existe entre la respuesta proporcionada por la red y los valores reales. Se busca minimizar esta función, para que los resultados sean fiables. La función empleada, al tratarse de un sistema binario, ha sido *'binary_crossentropy'*. Accuracy es el porcentaje de precisión de los valores de salida de la red, comparándolos con los valores reales.

5. Entrenamiento

Una vez la red está construida se procede al entrenamiento de los conjuntos de datos. Esto se hace con la función *fit* que toma matrices *numpy* y se encarga del entrenamiento en épocas con previa separación en lotes [32].

Los "*steps per epoch*" determinan el número de lotes (*batches*) que se seleccionan para una época de entrenamiento. En este experimento se han seleccionado 20 pasos por época, lo que quiere decir que la red se entrenará durante 20 lotes para completar una época.

Los "*validations steps*" se refieren a lo mismo que *"steps per epoch"* pero en lugar de para épocas de entrenamiento, para épocas con el conjunto de datos de validación.

III. Recogida y análisis de resultados.

Una vez se han ajustado los datos se ejecuta la CNN. Atendiendo a la respuesta de la red, se manipuló el código para optimizar los resultados de salida. Así, tras la variación

de parámetros y varias ejecuciones, se logró aumentar los niveles de precisión (*accuracy*) y estabilizar las pérdidas (*loss*).

Algunos de los parámetros clave con los que se trabajó son el valor de la x del optimizador [(optimizer=optimizers. RMSprop (lr=10^{-x})], las épocas (*epochs*), los pasos por época (*steps per epochs*), y los pasos por validación (*validation steps*).

Se han realizado varios experimentos manipulando las diferentes variables del código hasta conseguir unos valores óptimos. Los parámetros modificados son los mostrados en las tablas 2, 3, 4, 5 y 6. Con los valores de salida proporcionados por la red, se construyeron gráficas de resultados (Figuras 15, 16, 18, 19, 21, 22, 23, 24, 25, 26).

Datos por defecto del software: se ejecutó el código con los parámetros que venían integrados en el código descargado y se observó que los valores de pérdidas (en especial los de validación) no conseguían alcanzar niveles suficientemente bajos. Además, se observa en la figura 11 como las pérdidas de validación experimentan un crecimiento enorme en las últimas épocas, desestabilizando por completo los resultados. La precisión, aunque alcanzaba valores altos no se estabilizaba y existe una diferencia considerable entre los valores de precisión de entrenamiento y de validación. La respuesta de esta red no se podría emplear con fines médicos debido al alto nivel de errores que presenta.

Tabla 2: Variables por defecto del software

x	Epochs	Steps per epochs	Validation_steps
5	30	100	200

Fuente: Elaboración propia.

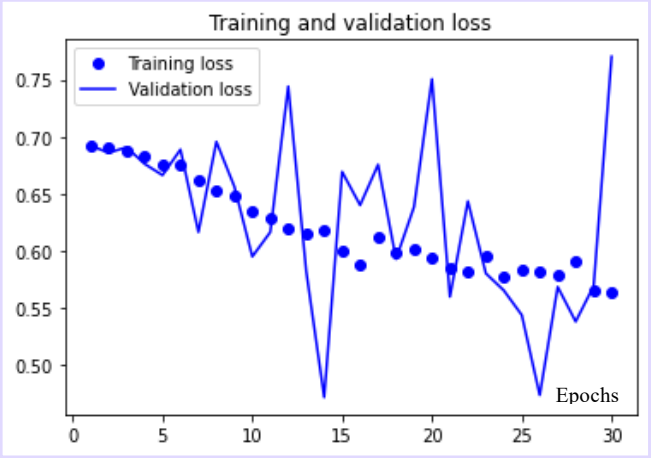

Figura 15: Gráfica resultados de pérdidas (Por defecto del software).

Fuente: Elaboración propia.

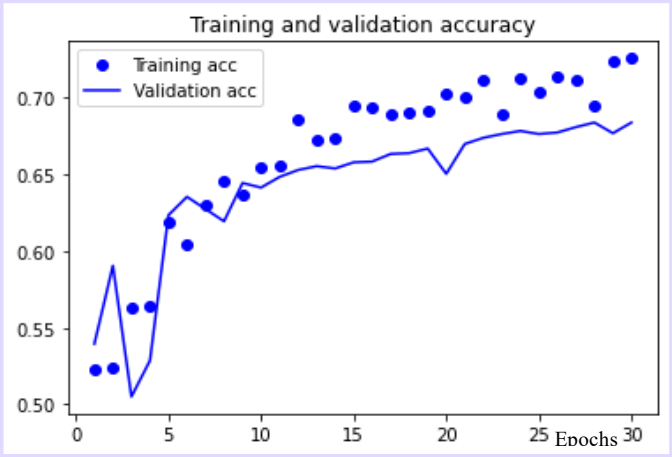

Figura 16: Gráfica resultados de precisión (Por defecto del software)

Fuente: Elaboración propia.

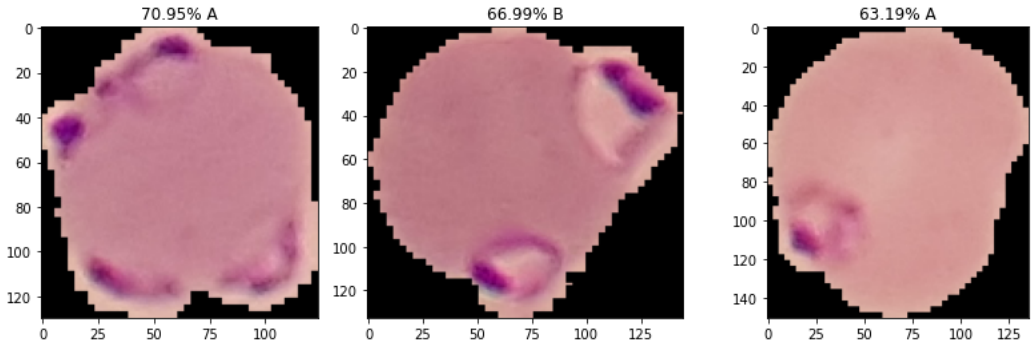

Figura 17: Imágenes de dos verdaderos positivos y un falso negativo (Por defecto del software).

Fuente: Elaboración propia.

Experimento 1: se ejecutó el código con los parámetros de la Tabla 3 y se observó que tanto los valores de pérdidas como los de validación habían mejorado notablemente con respecto al código por defecto del software. Sin embargo, los valores de pérdidas de validación seguían sin estabilizarse y los valores de las pérdidas de entrenamiento y validación eran muy dispares. La precisión alcanzaba valores óptimos, superior al 90 %, aparentemente estables y sin grandes variaciones entre los sets de entrenamiento y validación.

Tabla 3: Variables Experimento 1.

x	Epochs	Steps per epoch	Validation_step
3	20	50	70

Fuente: Elaboración propia.

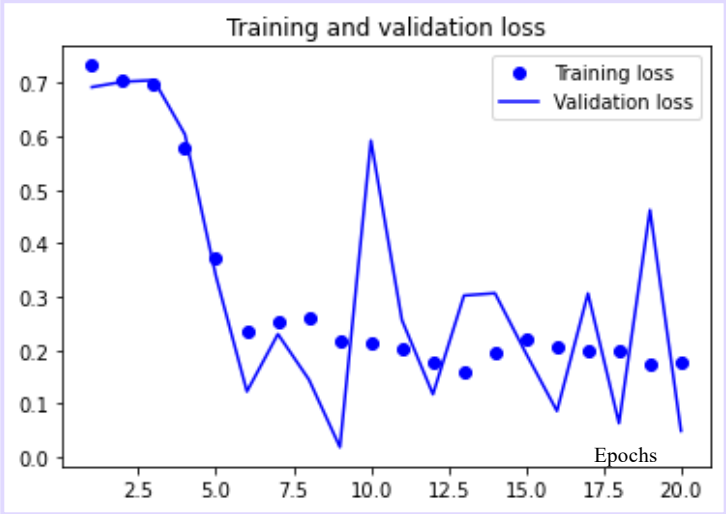

Figura 18: Gráfica resultados de pérdidas. Experimento 1

Fuente: Elaboración propia.

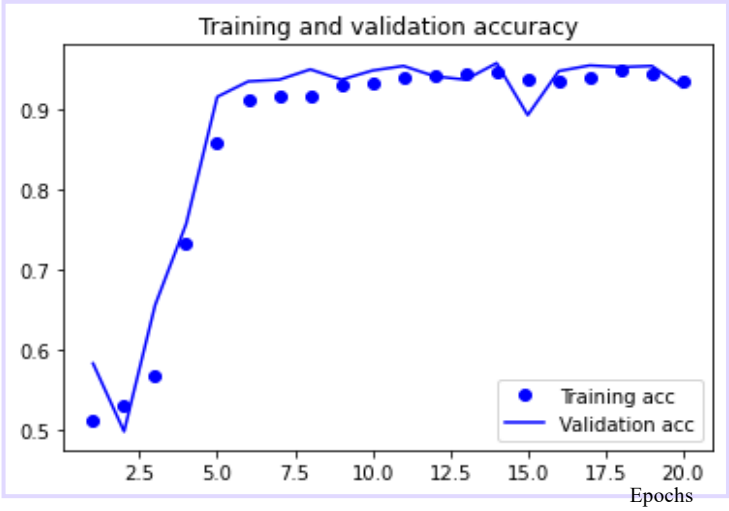

Figura 19: Gráfica resultados de precisión. Experimento 1

Fuente: Elaboración propia.

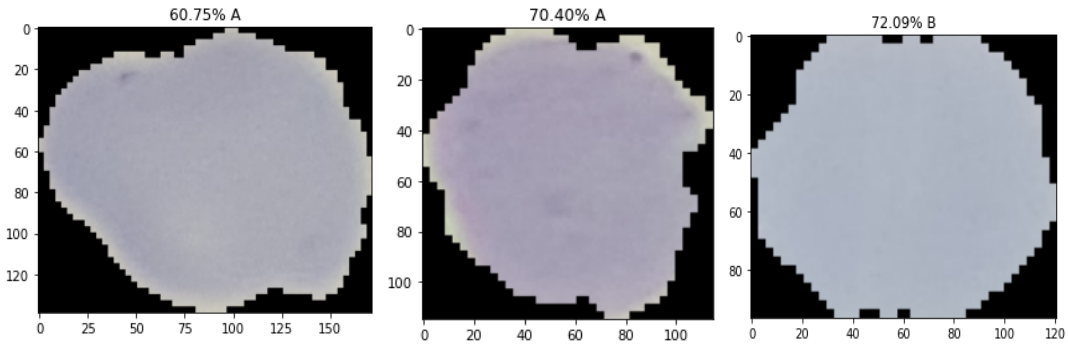

Figura 20: Imágenes de dos falsos positivos y un verdadero negativo. Experimento 1

Fuente: Elaboración propia.

Experimento 2: se ejecutó el código aumentado el numero de épocas y pasos de entrenamiento y de validación (Tabla 4) y se observó como los valores de pérdidas, aunque se estabilizasen y no hubiese demasiado desfase entre los valores de entrenamiento (0,5855) y validación (0,5692), habían aumentado con respecto al *Experimento 1*. Al mismo tiempo, los valores de precisión de entrenamiento (70,75 %) y de validación (67,4 %) presentaban un serio decrecimiento y no alcanzaban niveles tan fiables como el previo experimento.

Tabla 4: Variables Experimento 2.

x	Epochs	Steps per epoch	Validation_step
5	30	100	200

Fuente: Elaboración propia.

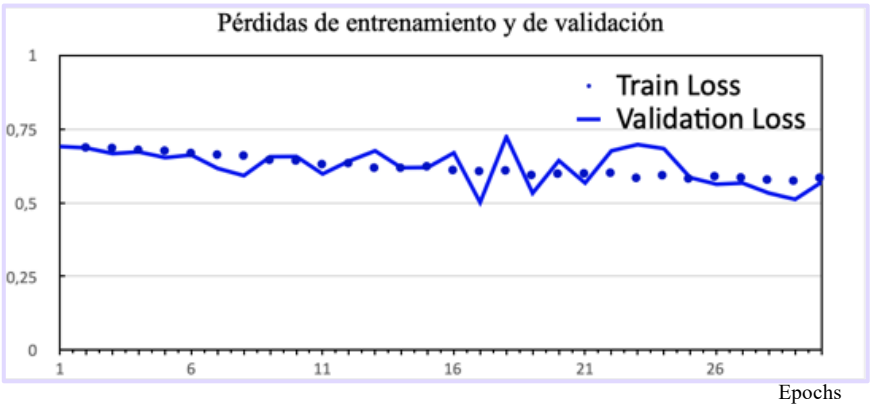

Figura 21: Gráfica resultados de pérdidas. Experimento 2. Fuente: elaboración propia.

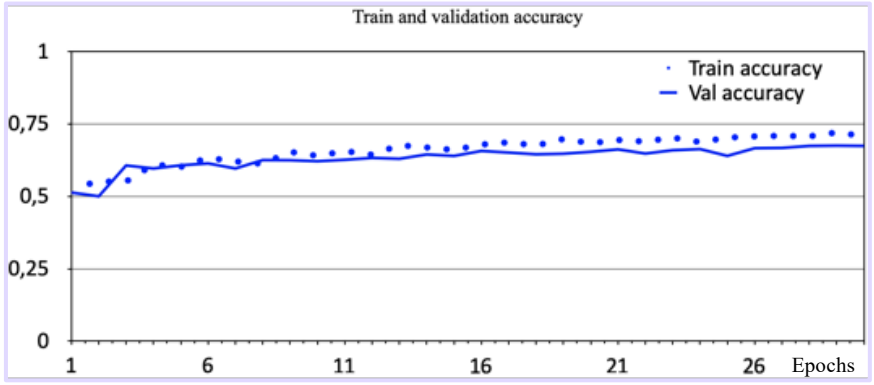

Figura 22: Gráfica resultados de precisión. Experimento 2. Fuente: elaboración propia.

Experimento 3: tras una tercera modificación de los parámetros y un nuevo proceso de entrenamineto, se recogieron los resultados. Estos muestran como las pérdidas de entrenamiento y validación siguen sin estabilizarse, pero sufren una notable mejora con respecto a los anteriores experimentos, obteniendo unos niveles de pérdidas de entrenamineto (0,1574) y de validación (0,2012) relativamente bajos.

En lo referido a la precisión, en este cuarto código se han alcanzado valores óptimos: estabilidad, mínimo desfase entre precisión de entrenamineto y de validación. En la última etapa, se alcanzan valores de entrenamineto (95,3 %) y validación (95,65 %) elevados y, en consecuencia, adecuados para la posible aplicación de diagnóstico médico.

Tabla 5: Variables Experimento 3.

x	Epochs	Steps per epoch	Validation_step
3	70	100	200

Fuente: elaboración propia.

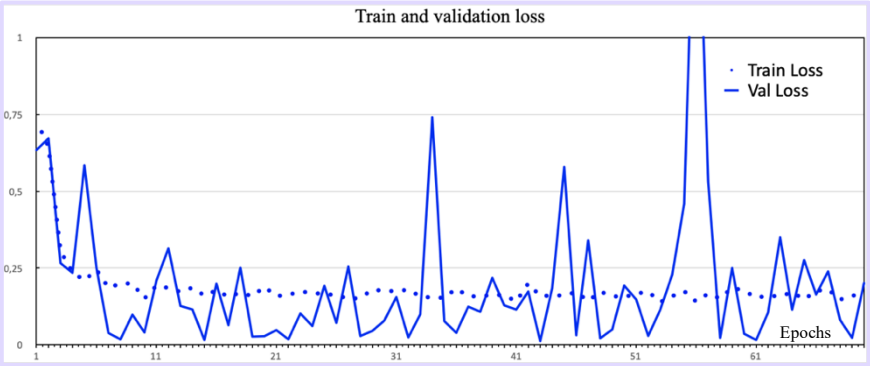

Figura 23: Gráfica resultados de pérdidas. Experimento 3. Fuente: elaboración propia.

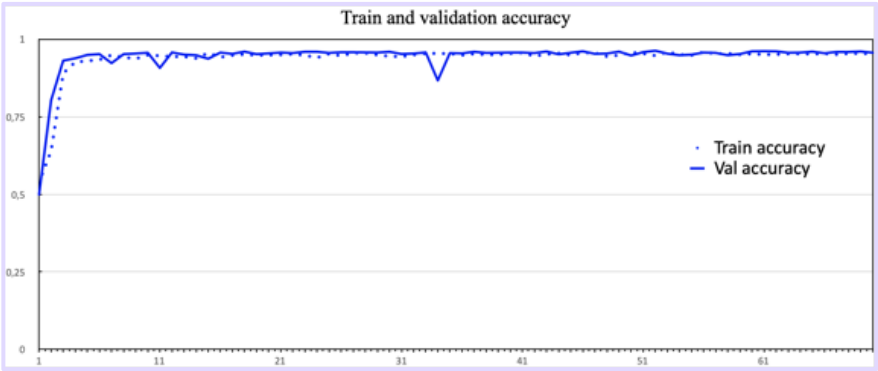

Figura 24: Gráfica resultados de precisión. Experimento 3. Fuente: elaboración propia.

Experimento 4: al observar los óptimos resultados de precisión del experimento 3, se mantuvo la x del optimizador y se cambiaron el número de pasos, aumentando los pasos

por época y disminuyendo los de validación. Tras correr el código y recoger los resultados, se aprecia la precisión ha aumentado incluso más con respecto a previos experimentos. Las pérdidas experimentan una ligera disminución, y se presentan más estables que cualquiera de los experimentos anteriores.

Tabla 6: Variables Experimento 4.

x	Epochs	Steps per epoch	Validation_step
3	50	200	100

Fuente: elaboración propia.

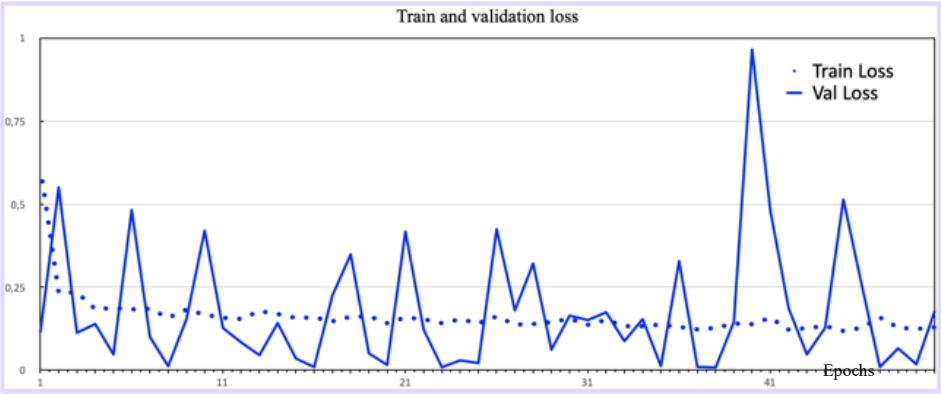

Figura 25: Gráfica resultados de pérdidas. Experimento 4.

Fuente: elaboración propia.

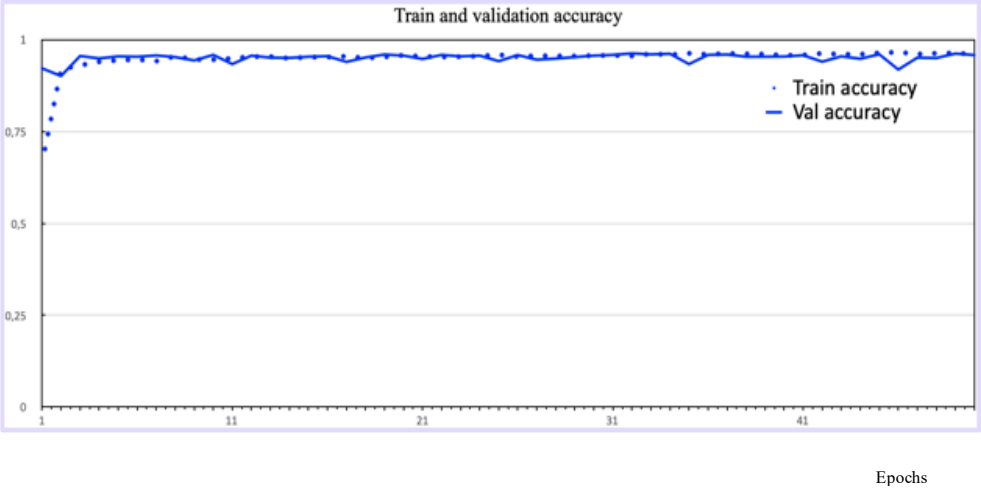

Figura 26: Gráfica resultados de precisión. Experimento 4.

Fuente: elaboración propia.

5. CONCLUSIONES

- ◊ Tras varias ejecuciones del código, se consiguió alcanzar unos resultados con unos valores de precisión de entrenamiento (95,97 %) y de validación (95,8 %) muy elevados y certeros. Además de unas pérdidas de entrenamiento (0,1292) y de validación (0,1770) mínimas.
- ◊ Si la x es muy pequeña, permite un margen de error muy grande en el algoritmo y los resultados no pueden alcanzar una precisión muy alta. Si el margen de error es muy pequeño, valores altos de x, el algoritmo no converge. Por ello, hay que elegir un valor intermedio, el mejor que se ha encontrado es x=3.
- ◊ El aumento de las épocas supone un aumento de la precisión. Cuanto más se entrena a la red, mejor es su respuesta.
- ◊ Los resultados mejoran notablemente si el número de pasos por épocas se incrementa con respecto al número de los pasos de validación. En los resultados de salida la importancia del entrenamiento tiene más peso que la de validación.
- ◊ Como se ha detallado anteriormente, con el experimento 4 se han alcanzado los objetivos perseguidos, obteniéndose los mejores resultados de precisión y con pérdidas mínimas.
- ◊ Podría considerarse su aplicación en el diagnóstico de malaria con fines médicos en aquellos países donde la escasez de medios imposibilita la detección de la enfermedad y causa grandes estragos en sus poblaciones. Este método mejorado supondría una mayor eficacia de diagnóstico, ligado a una reducción de la tasa de mortalidad por dicho padecimiento.
- ◊ Como una posible mejora, cabría mencionar la posibilidad del aumento de las imágenes y su calidad; y, un estudio más profundo de la red y sus parámetros.
- ◊ Una futura línea de investigación sería la detección automática mediante CNN de otras enfermedades, como puede ser la del COVID-19.

5.1. Futura línea de investigación

Para comprobar que el mismo código empleado para el diagnóstico de la enfermedad palúdica era funcional en otras patologías se llevo a cabo una búsqueda de bases de

datos de imágenes de sangre pertenecientes a pacientes infectados con COVID-19. No se encontraron muestras de sangre, pero sí imágenes de radiografías de pulmones. Estos directorios se bajaron desde la misma plataforma de la que se obtuvieron las imágenes de malaria (*Kaggle.com*).

Sin variar los parámetros con los que se consiguieron los mejores resultados del presente proyecto *(Experimento 4)*, se introdujeron las imágenes encontradas y se adaptó el código para que encajase con las imágenes de COVID-19: nombre, directorios en los que se encontraban y número.

Se ejecutó el código y se obtuvieron los resultados mostrados en las figuras 27 y 28.

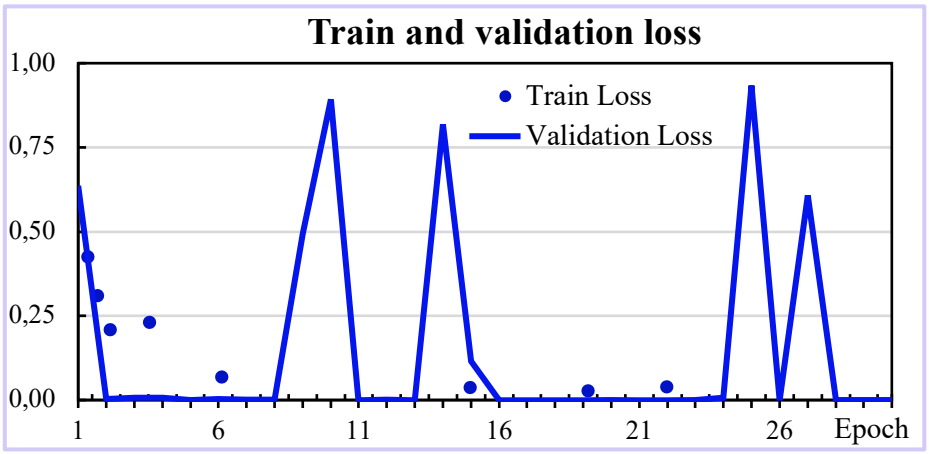

Figura 27: Gráfica resultados de pérdidas. Experimento COVID-19. Fuente: elaboración propia.

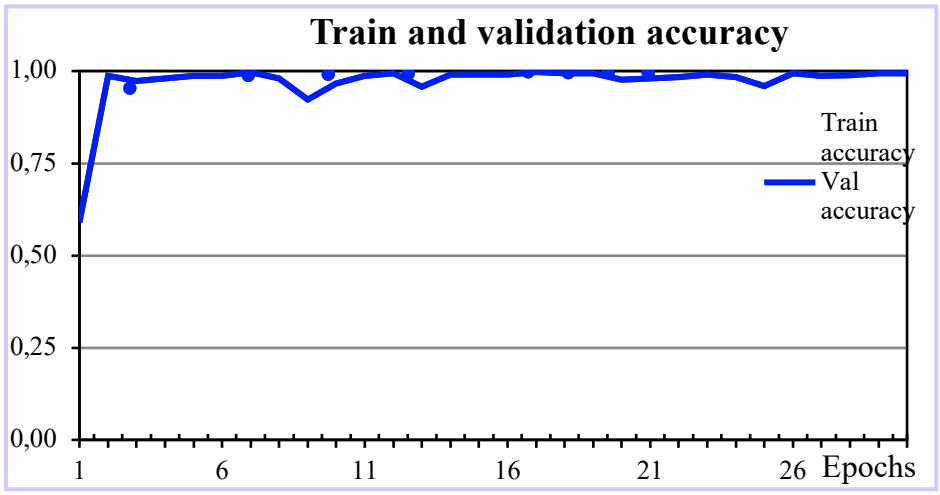

Figura 28: Gráfica resultados de precisión. Experimento COVID-19. Fuente: elaboración propia.

Los resultados extraídos de este último experimento mostraron en su última época, unos elevados y óptimos valores de precisión de entrenamiento (100 %) y de validación (99,33 %). Asimismo, apenas se aprecia diferencias significativas entre los valores de entrenamiento y validación, la precisión ha logrado estabilizarse.

A pesar de que en la gran mayoría de las épocas presentan unos valores de error mínimos, en otras las pérdidas de entrenamiento crecen repentinamente, imposibilitando la estabilidad de estas. Por otro lado, las pérdidas de validación sí han alcanzado estabilidad.

No obstante, las limitaciones se encuentran en la dificultad que existe en la obtención y elaboración de las imágenes de radiografías. Para que esta línea de investigación pueda tener utilidad práctica habría que emplear muestras más accesibles como puede ser imágenes de muestras de sangre o de saliva, cuya disponibilidad no se encuentra actualmente disponible en bases de datos públicas.

6. BIBLIOGRAFÍA

1. Barrada, Y., Castillo, P., Orozco, M. M., Rojas, G., Mejía, R., Ybarra, L. P., & Rondón, R. O. (2019). *Artículo Original*. *7*(1), 17–25. https://www.scipedia.com/public/Barrada_et_al_2019
2. Marí, R. B., & Peydró, R. J. (2008). Malaria en España: Aspectos entomológicos y perspectivas de futuro. *Revista Espanola de Salud Publica*, *82*(5), 467–479. https://doi.org/10.1590/s1135-57272008000500003
3. Global Malaria Programme: WHO Global. (2019). World malaria report 2019. In *WHO Regional Office for Africa*. https://www.who.int/news-room/fact-sheets/detail/malaria

Figura 29: Radiografía de paciente con COVID-19 (izquierda) y sano (derecha).
Experimento COVID-19.

4. https://lamalaria.com/formacion/evolucion-plasmodium-ciclo-biologico-malaria/
5. Cox FE (2010). Historia del descubrimiento de los parásitos de la malaria y sus vectores. Parásitos y vectores , 3 (1), 5. https://doi.org/10.1186/1756-3305-3-5
6. Acuña, Mauricio (2016). Prevalencia de Malaria por infección mixta de especies de plasmodium, frente a infecciones simples. Tesis para optar por el Título Profesional de Bioquìmico Clínico. Carrera de Bioquímica Clínica. Quito: UCE. 93 p.

7. https://medlineplus.gov/spanish/ency/article/000621.htm
8. Janosik, S. M. (2005). MALARIA 2010-2019. NASPA Journal, 42(4), 1. https://doi.org/10.1017/CBO9781107415324.004
9. Tobón, A. (2009). Danger signs in the malaria patient. *Biomédica : Revista Del Instituto Nacional de Salud*, *29*(2), 320–329. https://doi.org/10.7705/biomedica.v29i2.33
10. Salud, M. D. E. (n.d.). *Guía de atención de la malaria*. https://policia.gov.co/sites/default/files/49-MALARIA.pdf
11. https://www.who.int/es/news-room/fact-sheets/detail/malaria
12. https://cutt.ly/8hCi99r
13. *TRABAJO FIN DE GRADO TÍTULO : RTS , S / AS01 , ¿ El fin de la Malaria ? Autor : Amir Diab Cáceres Tutor : Juan José Nogal*. (2018). 1–20.
14. https://www.who.int/es/campaigns/world-malaria-day/world-malaria-day-2020
15. WHO. (2018). WHO | The World malaria report 2018. In *Who*. www.who.int/malaria%0Ahttps://apps.who.int/iris/bitstream/handle/10665/275867/9789241565653-eng.pdf?ua=1%0Ahttps://www.who.int/malaria/publications/world-malaria-report-2018/en/; consulté le 22/03/2019%0Ahttps://www.who.int/malaria/media/world-malaria-rep.
16. Global Malaria Programme: WHO Global. (2019). World malaria report 2019. In *WHO Regional Office for Africa*. https://www.who.int/news-room/fact-sheets/detail/malaria
17. García, A. (2012). INTELIGENCIA ARTIFICIAL. Fundamentos, práctica y aplicaciones. *RC Libros*, 285.
18. Raúl Benítez, Gerard Escudero, Samir Kanaan, D. M. R. (n.d.). Inteligencia artificial avanzada. *Editorial UOC*, 298 páginas.
19. Castillo, E., & Hadi, A. S. (1997). *Sistemas Expertos y Modelos de Redes Probabilísticas*.639.
20. Izaurieta, F., & Saavedra, C. (1999). Redes Neuronales Artificiales. *Charlas de Fisica*, 1–15. https://doi.org/10.1016/S0210-5691(05)74198-X
21. https://www.youtube.com/watch?v=eNIqz_noix8

22. Matich, D. J. (2001). Redes Neuronales: Conceptos Básicos y Aplicaciones. *Historia*, 55. ftp://decsai.ugr.es/pub/usuarios/castro/Material-Redes-Neuronales/Libros/matich-redesneuronales.pdf

23. Marín Diazaraque, J. M. (2007). Introducción a las redes neuronales aplicadas. *Manual Data Mining*, 1–31. halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema3dm.pdf http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema3dm.pdf

24. Delrieux, C. (2018). *DETECCIÓN DE EQUIPOS DE PROTECCIÓN PERSONAL*.

25. Nuñez, F. J. (2016). Diseño de un sistema de reconocimiento automático de matrículas de vehículos mediante una red neuronal convolucional.

26. https://www.ediciones eni.com/open/mediabook.aspx?idR=063c06e80706643a6614738d888a492d

27. https://aprendeia.com/introduccion-a-numpy-python-1/

28. https://aprendeia.com/introduccion-a-numpy-python-1/

29. https://aprendeconalf.es/docencia/python/manual/pandas/

30. https://rico-schmidt.name/pymotw-3/shutil/

31. https://sitiobigdata.com/2019/06/22/relu-funciones-activacion/#

32. https://www.apsl.net/blog/2018/02/02/tensor-flow-para-principiantes-vi-uso-de-la-api-keras/