Uso del algoritmo Local Binary Pattern Uniforme para el procesamiento de imagines de TC torácica en pacientes Covid-19

Ana Evelia Hernández Aguirre[1], Alan David Blanco Miranda[2], David Asael Gutiérrez Hernández[1]

[1]Tecnológico Nacional de México. campus León.León, Guanajuato, México.
[2]Universidad Tecnológica de León. León, Guanajuato, México.

Resumen

En el año 2019 fue cuando surgió el llamado COVID-19 (enfermedad de Coronavirus-19), nombrado así desde el 11 de febrero del 2020. A partir de estos años se empezó a enfocar principalmente en cómo combatir, curar o contrarrestar esta enfermedad. Actualmente sigue siendo prioridad descubrir avances que puedan ser de ayuda para el área médica en el tema del COVID-19. En esta investigación se procesan las imágenes de TC (Tomografías computarizadas) de la región torácica de pacientes con COVID-19 por medio del algoritmo LBP (Local Binary Pattern) uniforme. Se realizó una segmentación en cada una de las imágenes para poder recortarlas y centrarnos en el área de interés (pulmones), de esta manera evitar lo más posible otras áreas no necesarias (fuera del área de interés) y tener las imágenes preparadas para el algoritmo LBP, realizando una clasificación binaria (COVID-19 y No COVID-19) con una red neuronal perceptrón multicapa, dando a conocer los resultados de 11 experimentos con su respectiva exactitud, precisión y sensibilidad.

Abstract

In the year 2019 was when the so-called COVID-19 (Coronavirus-19 disease) arose, named like this since February 11, 2020. From these years on, the focus began mainly on how to combat, cure or counteract this illness. Currently, it is still a priority to discover advances that can be of help to the medical area on the subject of COVID-19. In this research, CT (Computerized Tomography) images of the thoracic region of patients with COVID-19 are processed by means of the uniform LBP (Local Binary Pattern) algorithm. A segmentation was performed on each of the images to be able to cut them and focus on the area of interest (lungs), thus avoiding as much as possible other unnecessary areas (outside the area of interest) and having the images prepared for the algorithm. LBP, performing a binary classification (COVID-19 and Non-COVID-19) with a multilayer perceptron neural network, revealing the results of 11 experiments with their respective accuracy, precision and sensitivity.

Palabras clave: Tomografía computarizada, Covid-19, algoritmo LBP, Clasificación
Keywords: CT scan, Covid-19, LBP algorithm, Classification.

1. INTRODUCCIÓN

El coronavirus SARS-Cov-2 es un virus que apareció en china y se fue extendiendo rápidamente a todo el mundo causando así una pandemia que ha afectado económicamente, laboralmente y sobre todo a la salud de cada persona. Esta enfermedad se denominó con el nombre de Covid-19. Se presentan diferentes tipos de síntomas los cuales afectan a cualquier tipo de persona, pero los más afectados han sido los adultos mayores. En un artículo de investigación [1] realizado en el año 2020 nos dan a conocer cómo es que los casos de personas fallecidas pueden llegar a aumentar de manera inesperada, el cómo es que México rápidamente llego a ser el octavo país con más casos de contagio. En solo 13 días, las victimas fallecidas de ser originalmente 32 mil a la fecha de 17 de junio de 2020, a aproximadamente 39 mil 500 para la fecha del 7 de julio de 2020. Lo que quiere decir que eran un promedio de 575 muertes diarias. Los médicos se han apoyado del resultado en el procesamiento de imágenes para la extracción de algún área de interés aplicando técnicas de inteligencia artificial. Y de esta manera llegar a un diagnóstico y/o prevenir enfermedades. En ocasiones no se ha logrado identificar la proporción del área afectada de los pacientes con COVID-19 o no es muy preciso el resultado, y por consecuente se le da un tratamiento que no es totalmente adecuado para la propagación de la enfermedad que ya tiene en los pulmones dicho paciente, lo cual causa aún más afectaciones que podrían conllevar a la

muerte si no es tratado a tiempo. Es por lo anterior que se busca clasificar si una TC (Tomografía Computarizada) es de un paciente con Covid-19 o con algún otro tipo de enfermedad. Realizando un preprocesamiento a las imágenes para utilizarlas de entrada al algoritmo LBP obteniendo los resultados para su respectiva clasificación en 11 experimentos realizados. Lo anterior haciendo uso del lenguaje de programación Python junto con OpenCV.

2. MARCO TEÓRICO

a. Tomografía computarizada

La Tomografía Computarizada o computada (TC) hace uso de rayos X para formar las imágenes, un anillo recorre al paciente para poder obtener las distintas densidades en las imágenes que finalmente se obtendrán. El uso de la TC fue aproximadamente hace 14 años este tipo de imágenes con los avances tecnológicos las imágenes ahora son de mayor calidad y se obtienen en menor tiempo, para de esta manera aportar la información de forma más clara [2].

En específico la TC de tórax es de importancia en el diagnóstico de Covid-19 ya que se puede descartar otras enfermedades, saber que tan dañado está el pulmón. Por lo anterior se recomienda realizar una TC torácica en pacientes con una elevada sospecha de daño pulmonar [3].

Figura 1. TC torácica de un paciente con covid 19 [4]

b. Imágenes a escala de grises

Las imágenes a escalas de grises están compuestas de intensidades de grises que varía desde el color negro hasta el color blanco, en estas imágenes se usa 8 bits para representar cada pixel por lo que solo se cuenta con 256 intensidades o escala de grises [5].

Figura 2. Ejemplo de escala de grises. [5]

c. Segmentación binaria con Thresh Otsu Open CV

Este método es uno de los más simples cuyo objetivo es dividir la imagen en 2 valores (0 y 1) utilizando técnicas estadísticas, como la varianza que es una medida de dispersión de los niveles de grises en la imagen [6], [7]. Lo anterior para poder enfocarnos en nuestra área de interés de la totalidad de la imagen.

ISSN: 2007-4786

Figura 3. TC torácica segmentada con Thresh otsu OpenCV, donde nuestra área de interés son los pulmones. [fuente propia]

d. Skimage-morphology OpenCV

Este método realiza operaciones morfológicas y elementos de estructuración que se definen en [8]. En general este método utiliza las formas y áreas de la imagen con una segmentación binaria previamente realizada. En esta investigación se hace uso de *skimage.morphology.remove_small_holes* [9] que rellena huecos con un área menor o igual a 1000 y *skimage.morphology.remove_small_objects* [10] que elimina objetos chicos con un área menor o igual a 1000.

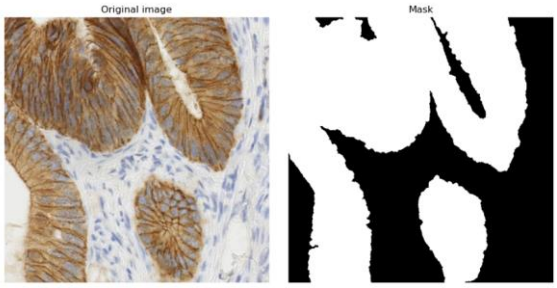

Figura 4. Ejemplo de una imagen original (imagen de la izquierda) y de una imagen binarizada (imagen de la derecha) ya quitándole los objetos chicos y rellenando los huecos pequeños con el método de Skimage_morphology [8]

e. Región de interés en una imagen (ROI)

En el análisis de imágenes existen casos donde no se requiere toda la información que la imagen proporción, es ahí donde necesitamos centrarnos en la región de interés (ROI). La región de interés es la zona más importante para nuestro proceso de clasificación o segmentación [11], en esta investigación nuestra zona de interés son los pulmones en las imágenes médicas (Tomografías Computarizadas) usadas.

Figura 5. Representación del área de interés (pulmones) de una TC [fuente propia]

f. Elipse

Se conoce geométricamente a una elipse como los puntos de un plano, los cuales sus sumas de las distancias a dos puntos fijos (focos), siempre es constante. La longitud constante de la elipse es también llamada eje mayor ya sea paralelo al eje x o al eje y [12].

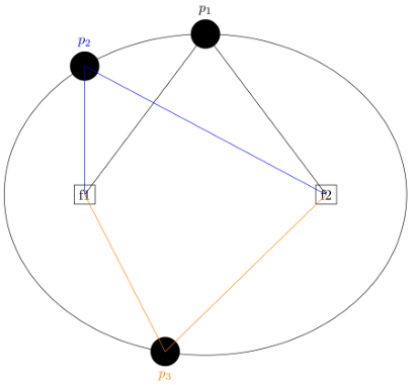

Figura 6. Imagen de una elipse donde se observa dos focos (f1 y f2), al igual que tres puntos diferentes (p1, p2 y p3), mostrando que la suma de las distancias a los focos des de un punto es constante [12].

g. LBP uniforme

El LBP uniforme está basado en el LBP original [13] en el cual a cada pixel de una imagen analiza su vecindario (ocho vecinos), estudiando si el nivel de gris en cada pixel supera un determinado umbral (ecuación (1)) y codificando dicha comparación mediante un numero binario (Figura 7). La diferencia con el LBP uniforme es que se da cuenta que la información relevante de la textura es descrita mediante un patrón uniforme el cual contiene como máximo dos transiciones binarias 0-1 o 1-0 (como se muestra en la Tabla I) de la cadena binaria resultante LBP [14]. Teniendo la cadena binaria se realiza la multiplicación de los pesos calculados por cada vecino iniciando por la esquina superior izquierda con 2^0 hasta llegar a 2^7, siguiendo la dirección de las manecillas del reloj y de esta forma obtener el resultado del nuevo valor que remplazara el valor del pixel central del vecindario (como se muestra en Figura 8).

Tabla 1. ejemplos de cadenas binarias uniformes y no uniformes. información basada en [14]

Binario	Uniforme	No uniforme
11011101		X
10111101		X
11100011	X	
11111001	X	

$$LBP_{P,R} = \sum_{P=0}^{P-1} S(g_P - g_c)2^P, S(x) = 1\ si\ x \geq 0, 0\ si\ x < 0. \quad (1)$$

Donde P es el número de vecinos que se va a considerar, R es el tamaño del vecindario, g_c es el valor del pixel central, g_P es el valor de cada uno de los P pixeles del vecindario.

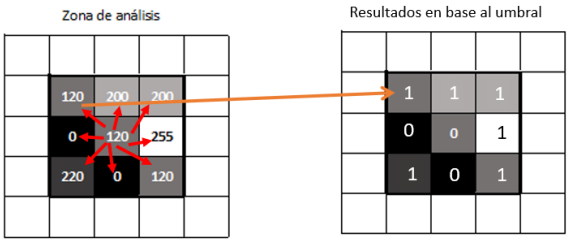

Figura 7. Zona de análisis simulando un área de píxeles de una imagen, su comparación del píxel central y sus vecinos alrededor, para llegar a la cadena binaria resultante de la comparación. Información basada en [14]

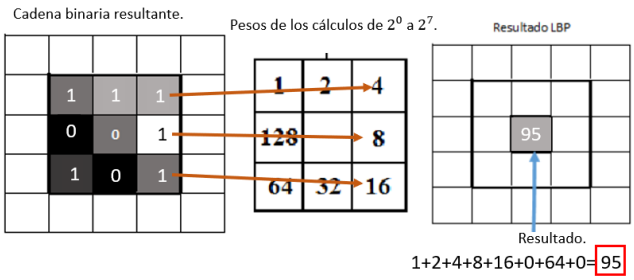

Figura 8. Cálculo del nuevo valor que remplazara al píxel central del vecindario. Información basada en [14]

El LBP uniforme solo usa 58 de los 256 patrones totales usando 8 vecinos, por que usa la restricción de patrones uniformes, sin esta restricción serían los 255 patrones que utiliza el LBP original [15]. A continuación, en la Figura 9 se muestra un resumen del proceso del LBP uniforme descrito anteriormente.

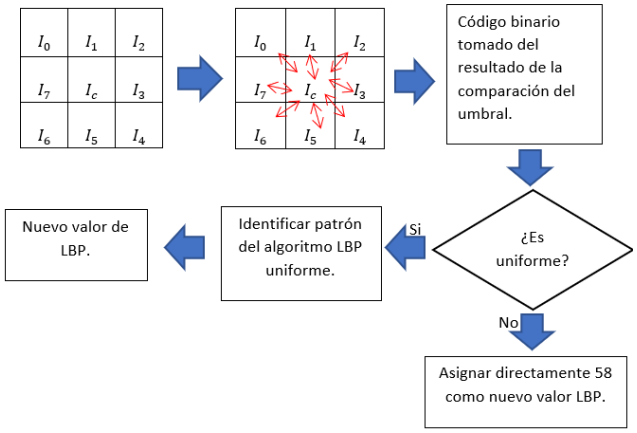

Figura 9. Proceso del algoritmo LBP uniforme, tomando 8 vecinos y en la parte central su umbral (valor central I_c). Información basada en [14]

h. Perceptrón multicapa

El perceptrón multicapa (MPL) también llamado "redes neuronales feed-forward" [16] contiene un conjunto de neuronas organizadas por capas. Las neuronas están conectadas entre ellas de manera que las salidas de una capa son las entradas de la capa siguiente. El número de capas y neuronas depende del propósito para el cual se utilizará MLP. Este tipo de red utiliza aprendizaje supervisado donde se conocen los valores de salida para poder compararlos con los de la red neuronal, si llegara a existir alguna diferencia en los resultados la red se ajustará [16] [17].

Este tipo de red neuronal consiste en 3 capas:

- Una capa de entrada (primera capa), la cual recibe la información.
- Una cantidad de capas ocultas (intermedias), son las encargadas de hacer el proceso de la red neuronal.
- Una capa de salida (última capa), nos proporciona el resultado de clasificación [16].

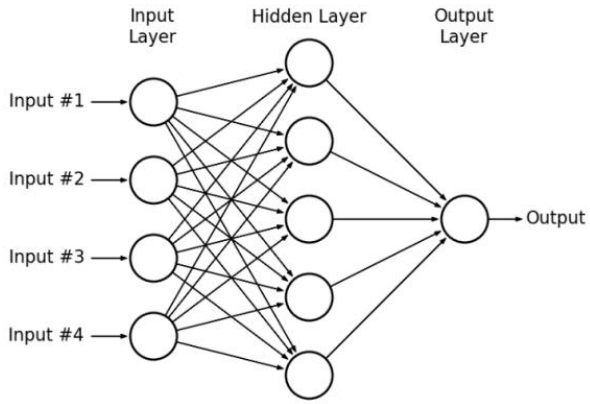

Figura 10. Ejemplo de arquitectura de una MLP. Imagen tomada de [16]

El entrenamiento de un MLP ajusta los pesos para reducir el error resultante entre la salida de la red y nuestras etiquetas, se utiliza una función de error de la ecuación (2) para lo anterior [18].

$$E(w) \equiv \frac{1}{2}\sum_{d\in D}(t_d - O_d)^2 \qquad \textbf{(2)}$$

Donde:

- D es el conjunto de patrones de entrenamiento,
- t_d es la salida esperada,
- O_d es la salida calculada por la red.

Para el entrenamiento también se hace uso de Backpropagation, algoritmo basado en el descenso de gradiente y la regla de la cadena [18]. Los pesos utilizados son inicializados aleatoriamente y en el entrenamiento estos pesos van cambiando conforme se reduce el error, mediante la regla delta de la ecuación (3).

$$\Delta w = -\eta \frac{\partial J}{\partial w} \qquad \textbf{(3)}$$

Donde η es la taza de aprendizaje, ∂w representa a la derivada de la función de error con respecto a los pesos ∂w [18]. Entonces la regla de actualización en la t-ésima iteración es mostrada en la ecuación (4):

$$w(t+1) = w(t) + \Delta w(t). \qquad \textbf{(4)}$$

Donde w(t+1) representa al nuevo valor de actualización de pesos, w(t) es el valor anterior en la t-ésima iteración, Δw(t) es el valor resultante de la regla delta en la t-ésima iteración.

Para dar un panorama general de entrenamiento Backpropagation vamos a considerar un MLP con tres capas (capa de entrada, capa oculta y capa de salida).

1. Se actualizan los pesos de la capa oculta a la capa de salida con la ecuación (5).

$$\Delta w_{kj} = -\eta(z_k - t_k)y_j \qquad \textbf{(5)}$$

Donde, -η es la taza de aprendizaje, z_k es la respuesta de la capa de salida en la clase k, t_k es la salida deseada, y_j representa la respuesta del t-ésimo nodo oculto activado por la función de activación [18].

2. Se actualizan los pesos de la capa de entrada a la capa oculta, como se muestra en la ecuación (6).

$$\Delta w_{ji} = -\eta\left(\sum_{k=1}^{c}\delta_k w_{kj}\right)\left(\sigma(u_{ji})\left(1-\sigma(u_{ji})\right)\right)x_i \qquad (6)$$

Donde c, representa a la cantidad de nodos de salida. $\sigma(u_{ji})$ representa el resultado de la multiplicación de $(x_i\, w_{ji})$, es decir la i-ésima variable del patrón de entrada x_i multiplicada por el peso w_{ji}, que lo conecta con el j-ésimo nodo oculto. δ_k representa el resultado de $(z_k - t_k)$ y w_{kj} son los pesos de la capa oculta que lo conecta con él k-ésimo nodo de salida [18]. Este tipo de redes neuronales usan una función de activación que en el caso de esta investigación será sigmoide. La función sigmoide transforma los valores introducidos a una escala (0, 1), donde los valores máximos tienden a 1 y los valores mínimos tienden a 0 [16], [17].

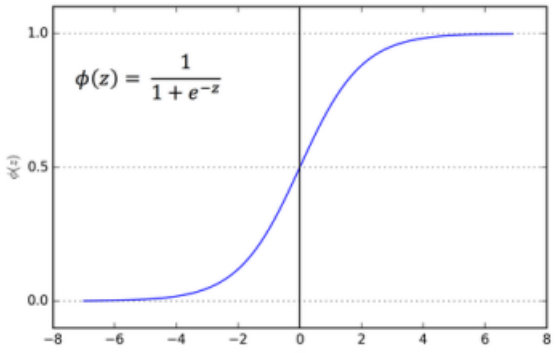

Figura 11. Gráfica de la función sigmoide.[16].

i. Métricas de clasificación

De los algoritmos de clasificación se pueden extraer métricas, las cuales ayudan a saber que tan bien funciona nuestro clasificador. Estas métricas son extraídas de una matriz de confusión la cual muestra en un cuadro los valores de: falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos. Al realizar nuestra fase de entrenamiento y fase de prueba o validación, se puede aplicar a calcular la matriz de confusión teniendo los datos esperados y los datos devueltos del clasificador [19].

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos(FP)	Verdaderos Negativos(VN)

Figura 12. Matriz de confusion para una clasificacion binaria [19]

- Verdaderos positivos (VP): datos clasificados como pertenecientes a su clase que corresponden.
- Falsos positivos (FP): datos que no pertenecían a la clase, pero fueron considerados como pertenecientes a la clase.
- Falsos negativos (FN): datos que pertenecían a una clase, pero no se asignaron a la clase que corresponden.
- Verdaderos Negativos (VN): datos que no pertenecen a una clase y los clasificó correctamente [19].

Estos valores mostrados en la Figura 12 nos permiten calcular nuevas métricas de rendimiento para extraer más información. Estas métricas son calculadas por medio de fórmulas que se especifican en las ecuaciones (7) a (9) [19] y son usadas para mostrar sus respectivos resultados en los 11 experimentos realizados en este articulo.

Exactitud: porcentaje de observaciones clasificadas correctamente.

$$Exactitud = \frac{VP + VN}{VP + VN + FP + FN} \quad (7)$$

Precisión: mide la habilidad de un clasificador de no etiquetar como positivo una observación siendo negativa. Entre mayor sea este valor mejor será.

$$Precision = \frac{VP}{VP + FP} \quad (8)$$

Sensibilidad o recuperación: mide la capacidad del clasificador de encontrar todos los datos positivos.

$$Recuperación = \frac{VP}{(VP + FN)} \quad (9)$$

3. METODOLOGÍA

La metodología que se realizo fue la que se muestra en la Figura 13, donde se describen los pasos a seguir de manera general.

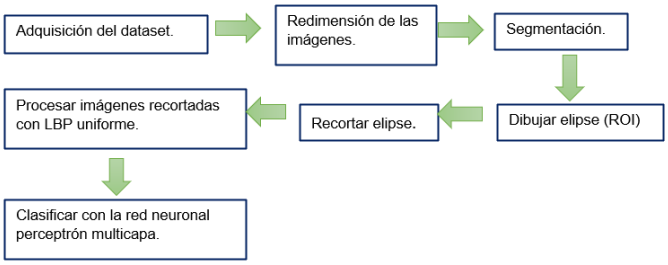

Figura 13. Diagrama de la metodología que se usó en la investigación.[fuente propia].

El lenguaje de programación Python fue utilizado, por ser una herramienta que brinda gran ayuda en manejo de grandes cantidades de datos, procesamiento de imágenes y operaciones matriciales. Para el preprocesamiento de la imagen se cambiaron las dimensiones de las imágenes uniformemente ya que el dataset que se utilizó contiene gran variación en dimensiones. Las medidas usadas fueron: 236 en altura y 348 en anchura, se eligieron estas cantidades basándonos en la menor altura y la menor anchura encontrada en el conjunto de imágenes. El preprocesamiento de cada imagen se convirtió a escala de grises, después se realizó una segmentación binaria que se realizó utilizando un umbral devuelto por la función de OpenCV THRESH_OTSU en combinación con THRESH_BINARY [6], [7]. Se binariza la imagen a continuación, se rellenan los agujeros pequeños con un área menor o igual a 1000, se remueven los objetos chicos de un área menor o igual a 1000, lo anterior con la ayuda de Skimage-morphology [8], [9], [10]. Se dibuja una elipse encerrando la región de interés (zona pulmonar) con la función OpenCV-ellipse [20].

Una vez realizado lo anterior se recortan las imágenes con matplotlib-patches [21] . Para procesar las imágenes con LBP uniforme y el vector de características o patrones resultantes dárselos de entrada a la red perceptrón multicapa.

a. Adquisición del dataset

En este proyecto de investigación se usaron 2 datasets, donde uno es de imágenes de TC torácicas de pacientes con Covid-19 y el otro dataset es de pacientes sin Covid-19 [4]. Este conjunto de datos (dataset) se encuentra en un repositorio de GitHub, donde se van recopilando imágenes de pacientes con y sin Covid-19. En la Figura 14 se muestra algunos ejemplos de los datasets usados. Se creó un archivo con extensión csv, para poder organizar la combinación de cada dataset indicando a que clase pertenece y su vector de características correspondiente a cada imagen y de esta manera poder llamar el archivo con Python.

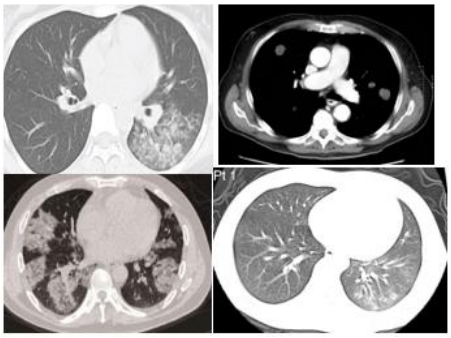

Figura 14. Ejemplo de las imágenes de los 2 datasets.[4]

b. Arquitectura de Red neuronal Perceptrón Multicapa

La arquitectura de la red neuronal es la siguiente:

- Número de neuronas de la capa de entrada: 58 neuronas.
- Número de neuronas en la primera capa intermedia: 58 neuronas.
- Número de neuronas en la segunda capa intermedia: 57 neuronas.
- Número de neuronas en la capa de salida: 2 neuronas.

En la capa de entrada se seleccionaron 58 neuronas porque el vector de características que nos proporciona LBP uniforme es de los 58 patrones encontrados en cada imagen.

En la capa intermedia u oculta se aplicó la ecuación (10):

$$Total\ neuronas = \big((N(2)\big) - 1) \qquad \textbf{(10)}$$

Donde N es el número de neuronas que contiene la capa de entrada.

Sustituyendo los valores en esta ecuación (10), nos da un resultado de 115 neuronas en la capa intermedia. Pero se decidió dividir en 2 esa capa oculta de 115 neuronas para dividir la carga de trabajo en 2 capas ocultas, dándonos como resultado la primera capa oculta de 58 neuronas y la segunda capa oculta de 57 neuronas. La función de activación usada es la función sigmoidal mencionada anteriormente [16]. Para los resultados se usaron 11 experimentos independientes con 1000 generaciones cada uno y una división de 30% para los datos de prueba, los datos de entrenamiento son 70% del total de las 746 imágenes de la combinación de los 2 datasets. El optimizador para ajustar los pesos en la red neuronal es el SGD (Descenso de Gradiente Estocástico) con una taza de aprendizaje de 0.001 y función de perdida el error cuadrático medio. En la Figura 15 se muestra gráficamente la arquitectura de la red neuronal.

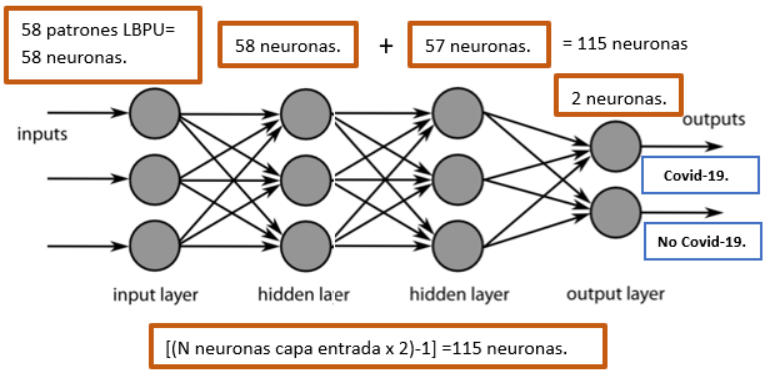

Figura 15. Representación de arquitectura de la red neuronal perceptrón multicapa [fuente propia]

c. Dibujar elipse

El centro de la elipse es el centro de la imagen total con sus respectivas dimensiones, esto se decidió porque en el caso de los datasets que se usan, el centro de nuestra zona de interés coincide en la mayoría de los casos con el centro de la imagen total. Para decidir que eje mayor (eje horizontal) de la elipse se tomará, es con base en la fila de pixeles del eje Y de la coordenada del centro de la elipse, donde se encuentren valores 0 (fondo de la imagen binaria) se mide la distancia desde esa posición a la posición del centro de la elipse. Lo anterior para cada pulmón (tanto izquierdo como derecho) se comparan los dos ejes mayores a partir de la coordenada central y el que sea mayor en cuanto a su medida (eje mayor izquierdo o eje mayor derecho) se toma como el eje mayor definitivo para la elipse. Para decidir que eje menor (eje vertical) de la elipse se tomará, se hace de la misma manera que el eje mayor, solo que en este caso se toma la columna de eje X de la coordenada del centro de la elipse, otra de sus diferencias en el proceso en comparación de eje mayor es que aquí se toma la altura del pulmón (pulmón izquierdo y pulmón derecho) con base en la binarización realizada y se toma el eje con mayor medida para tomarlo como el eje menor definitivo para la elipse.

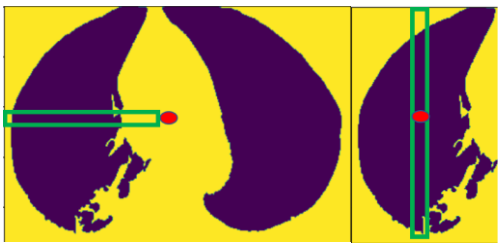

Figura 16. Ejemplo de como se tomó el eje menor y el eje mayor [fuente propia]

d. Recortar elipse

La elipse se recortó con matplotlib-patches [21], que nos permite recortar y guardar la imagen en base a la medida de la elipse que se le proporcione.

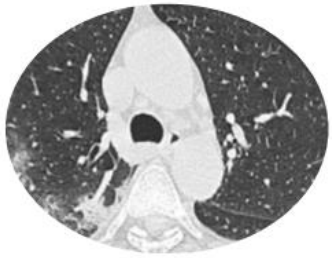

Figura 17. Imagen recortada basándonos en el contorno de la elipse generada [fuente propia]

e. Procesar imágenes recortadas con LBP uniforme

Las imágenes son procesadas mediante el algoritmo LBP uniforme [14] mencionado anteriormente, que nos dará de salida un vector de 58 patrones el cual será la entrada de la red neuronal. Tendremos un vector de patrones por cada imagen procesada para finalmente tener 746 vectores para ser clasificados.

f. Clasificar con red neuronal perceptrón multicapa

Una vez teniendo los vectores de características por cada imagen, se introduce cada vector a la red neuronal. Previamente se realizó la separación de vectores de entrenamiento y vectores de prueba (mencionado en la sección de Método en este documento), en el proceso de entrenamiento se ajustan los pesos hasta terminar con la cantidad de iteraciones (1000 iteraciones en este caso) y llegar a la fase de prueba para evaluar los datos seleccionados como prueba y obtener los resultados correspondientes a la clasificación.

4. RESULTADOS

Los resultados de los 11 experimentos de clasificación binaria en cuanto a exactitud, precisión y sensibilidad son los que se muestran en la Tabla II.

Tabla 2. Resultados de las métricas en cada uno de los 11 experimentos [fuente propia]

Experimentos.	Exactitud.	Precisión.	Sensibilidad.
1	0.451	0.458	0.924
2	0.558	0.875	0.067
3	0.455	0.460	0.933
4	0.536	0.556	0.048
5	0.464	0.455	0.724
6	0.429	0.436	0.743
7	0.406	0.425	0.752
8	0.580	0.704	0.181
9	0.522	0.527	0.983
10	0.420	0.432	0.752
11	0.429	0.425	0.619

5. DISCUSIÓN

Los resultados obtenidos en cuanto a exactitud de la clasificación no llegaron al objetivo, el cual consistía en tener una exactitud mayor o igual a 70%, en cada uno de los experimentos no fue superior a 60%. Se realizaron 11 experimentos donde el promedio de exactitud fue 47.72 %, la mayor exactitud 58.0% y la menor exactitud 42.0% del total de experimentos realizados. Estos resultados fueron afectados por el método de segmentación empleado para la extracción de las regiones de interés porque no fue exactamente preciso para todas las imágenes de los dos datasets, en algunos casos se toma más allá de la región de interés necesaria. Existieron otros factores que influyeron negativamente a los resultados como lo es la variación de contraste en las imágenes de los datasets y anotaciones sobre dichas imágenes que afectaban al área de interés. Con los resultados obtenidos se pueden realizar cambios necesarios para probar otros métodos a seguir y comparar resultados con los de este proyecto.

6. CONCLUSIONES

En este trabajo de investigación se obtuvieron resultados diferentes a los esperados en el objetivo, pero nos hizo darnos cuenta de la importancia del preprocesamiento de imágenes, el funcionamiento del algoritmo LBP

y poder clasificar imágenes médicas con una metodología propuesta. Un buen preprocesamiento de datos junto con una buena obtención del área de interés e imágenes con una mayor calidad, pueden influir en la obtención de resultados más altos en cuanto a las métricas que en este trabajo se utilizaron para evaluar los resultados del clasificador. También se utilizó herramientas que se desconocían en el lenguaje de programación Python y que gracias a la realización de este trabajo se pudieron implementar y obtener los resultados necesarios para seguir con el proceso de metodología.

REFERENCIAS

[1] Kanter, I. "Muertes por Covid-19 en México". Instituto Belisario Domínguez, 2020. http://www.bibliodigitalibd.senado.gob.mx/handle/123456789/4927

[2] H. Patricia Bitar, "Evaluación cardiaca con tomografía computada y resonancia magnética", vol. 24, Revista Medica Clinica Las Condes, 2013,pp.54-62, https://www.sciencedirect.com/science/article/pii/S0716864013701299

[3] L. Prieto, E. Garcia, D. Perez and Y. Gonzalez, "Utilidad de la tomografía computarizada torácica en el diagnostico a pacientes con COVID-19", Primera Jornada Científica Virtual de COVID-19 en Cienfuegos, 2022, https://covidcien2022.sld.cu/index.php/covidcien/2022/paper/viewPaper/314

[4] Z. Jinyu, Z. Yichen, H. Xuehai and X. Pengtao, "COVID-CT-Dataset: a CT scan dataset about COVID-19", arXiv preprint arXiv:2003.13865, 2020, https://github.com/UCSD-AI4H/COVID-CT

[5] E. Dorronsoro, "Segmentacion de imágenes mediante logica difusa", Tesis, escuela tecnica superior de ingenieros industriales y de telecomunicacion, 2010, https://academica-e.unavarra.es/handle/2454/1951

[6] R. Acosta y L. Toruño, "Reconocimiento de placas vehiculares aplicando procesamiento de imágenes digitales en Python-OpenCV", tesis, Universidad Nacional Autonoma de Nicaragua, Loen, 2020, http://riul.unanleon.edu.ni:8080/jspui/bitstream/123456789/8177/1/245048.pdf

[7] OpenCV, «Image Thresholding,» OpenCV 4.6.0, 2021. [En línea]. Available: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html.

[8] Scikit-image, Scikit-image image processing in python, 2022. [En linea]. Avalailable: https://scikit-image.org/docs/stable/api/skimage.morphology.html

[9] Scikit-image_remove_small_holes, remove_small_holes, 2022. [En linea]. Available: https://scikit-image.org/docs/stable/api/skimage.morphology.html#skimage.morphology.remove_small_holes

[10] Scikit-image_remove_small_holes, remove_small_objects, 2022. [En linea]. https://scikit-image.org/docs/stable/api/skimage.morphology.html#skimage.morphology.remove_small_objects

[11] P. Jangbari and D. Patel, "Review on region of interest coding techniques for medical image compression," Tech. Rep., 2016.

[12] A. Rincon, "la elipse", Noche de las estrellas articulo de divulgacion de la ciencia, 2021.

[13] O. Garcia y E. Alegre, «Descripción de textura en imágenes utilizando Local Binary Pattern (LPB),» Conceptos y métodos en visión por computador, vol. 7, pp. 115-130, 2016.

[14] J. Chen y et al, «RLBP: Robust Local Binary Pattern,» British Machine Vision Conference, pp. 122.1-122.11, 2013.

[15] E. J. De Ramon Balmaseda, «Transformaciones basadas en el algoritmo Local Binary Pattern de imágenes capturadas con la Kinect para clasificación facial,» Tesis de Maestria, Oviedo, España, 2011

[16] I. Quereda Llopis, «Clasificación de imágenes TC de tórax afectadas por tuberculosis,» Obtencion de grado en ingenieria, San Vicente del Raspeig, España, 2018.

[17] M. Desai y M. Shah, «An anatomization on breast cancer detection and diagnosis employing multi-layer perceptron neural network (MLP) and Convolutional neural network (CNN),» Clinical eHealth, vol. 4, pp. 1-11, 2021

[18] E. Caicedo y J. Lopéz, «Perceptrón Multicapa y Algoritmo Backpropagation.,» Una aproximación práctica a las Redes Neuronales Artificiales, vol. 1, pp. 75-136, 2009.

[19] J. Zamorano Ruiz, «Comparación y análisis de métodos de clasificación con las bibliotecas scikit-learn y TensorFlow en Python,» Repositorio Institucional de la Universidad de Málaga, Málaga, España., 2019.

[20] OpenCV, «Drawing Functions,» 2021. [En línea]. Available: https://docs.opencv.org/4.x/d6/d6e/group__imgproc__draw.html#ga57be400d8eff22fb946ae90c8e7441f9.

[21] Matplotlib, «Matplotlib,» 2021. [En línea]. Available: https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.patches.Ellipse.html.

Correo de autor de correspondencia: ***david.gutierrez@leon.tecnm.mx***