ISSN 1870-4069

Clasificador de plantas medicinales por medio de Deep Learning

Juan Monroy-de-Jesús, Adriana Reyes-Nava, Fernando Olmos

Tecnológico de Estudios Superiores de Jocotitlán, Jocotitlán, México
{juan.monroy, adriana.reyes}@tesjo.edu.mx, fol9602@gmail.com

Resumen. El presente trabajo plantea la implementación de un algoritmo basado en Deep Learning para la identificación de plantas medicinales. El propósito es que el usuario le pueda indicar al sistema que planta buscar e inmediatamente se active la búsqueda de esta o bien que cuando se le muestre una planta devuelva una respuesta donde se haga una descripción de la planta con sus características. Esto se logró mediante el desarrollo de una red neuronal de aprendizaje profundo, a través del entrenamiento con una colección de corpus (base de datos) propia. Se busca que el sistema de reconocimiento sea implementado a futuro en un robot recolector de basura, donde sus dos funciones principales sean recolectar desechos e identificar plantas medicinales.

Palabras clave: red neuronal convolucional, CNN, aprendizaje profundo, inteligencia artificial.

Plant Sorter Using Deep Learning

Abstract. The present work propose the implementation of an algorithm based on Deep Learning for identification of medicinal plants.The porpose is that user can tell the system which plant look for and inmediately it active the search for it or that when plant is shown it return a response where will show a description about the plant with its characteristics.This achieved through the development of a neural network of deep learning , through a training with a collection of corpus (database) own it. The aim is that the recognition system to be implemented in a future in a garbage collection robot, where its two functions are wastecollection and identify medicinal plants.

Keywords: convolutional neural network, CNN, deep learning, artificial intelligence.

1. Introducción

Actualmente, con el uso de técnicas de aprendizaje profundo, se busca automatizar tareas en diversas áreas de estudio; campos como la medicina, industria automotriz o alimenticia se han visto beneficiados [1]; la botánica no es la excepción, en los últimos años se han implementado técnicas de clasificación con enfoques de estudio [2], de producción agrícola [3] e identificación de enfermedades que puedan padecer los cultivos [4, 5].

Partiendo de lo anterior, las Redes Neuronales Convolucionales, son un modelo donde las neuronas corresponden a campos receptivos que buscan simular la composición de las neuronas de la corteza visual primaria de un cerebro biológico.

La red se compone de múltiples capas; en el principio se encuentra la fase de extracción de características compuesta de neuronas convolucionales y de reducción; conforme avanzan las iteraciones del algoritmo se disminuyen sus dimensiones activando características cada vez más complejas; para llegados al final se utilicen neuronas sencillas las cuales realizan la clasificación.

En este tipo de arquitecturas se han logrado avances importantes en el campo de la Inteligencia Artificial, al proveer diferentes estructuras y mejorar los algoritmos de aprendizaje permitiendo mayor flexibilidad en los modelos de clasificación [6].

Sin embargo, este no es el único modelo que se emplea actualmente para que las máquinas sean capaces de aprender. Otros algoritmos son los árboles de decisión, las reglas de asociación, los algoritmos genéticos, las redes bayesianas o el aprendizaje por refuerzo [7].

Por otra parte, actualmente el uso de fármacos se ha utilizado para tratar cualquier enfermedad en las personas, sin embargo, estos fármacos se ha demostrado que pueden generar efectos secundarios en las personas, por ejemplo, problemas en los riñones, páncreas, acides, entre otras. Sin embargo, la comunidad en general ha olvidado el tratamiento tradicional de antaño, estos tratamientos se basan en remedios con plantas medicinales, las cuales se ha olvidado sobre sus propiedades curativas y aun mejor que no casusa problemas secundarios en las personas.

El presente trabajo muestra la implementación de un algoritmo convolucional clasificador de plantas medicinales con ayuda de Deep Learning.

2. Contexto general para la clasificación de imágenes

La clasificación de imágenes consiste en asignar a una imagen una etiqueta de un conjunto definido de categorías en función de sus características.

Factores como la escala, condiciones de iluminación, deformaciones u ocultamiento parcial de objetos hacen de la clasificación de imágenes una tarea compleja, a la que se ha dedicado un gran esfuerzo para desarrollar sofisticadas técnicas de reconocimiento de patrones, que no siempre producen los resultados esperados.

Desde el punto de vista del aprendizaje automático, la clasificación de imágenes es un problema de aprendizaje supervisado, en el que los algoritmos clasificadores generan un modelo a partir de un Dataset o conjunto de imágenes previamente categorizadas. El modelo obtenido se utiliza posteriormente para clasificar nuevas imágenes.

Para estos algoritmos las imágenes son matrices tridimensionales cuyas dimensiones son el ancho, alto y la profundidad de color, siendo el contenido de cada posición de la matriz un valor numérico que representa la intensidad de color de cada píxel de la imagen digital [8].

2.1. Redes neuronales

Las redes neuronales artificiales son redes interconectadas masivamente en paralelo de elementos simples, inspirados en el modelo biológico del ser humano [9],

actualmente existen diferentes modelos de redes neuronales, por ejemplo, las redes bayesianas, el perceptrón multicapa, redes recurrentes, convolucionales, entre otras. siendo esta última, tema de interés en este trabajo

2.2. Redes neuronales convolucionales

Son un tipo particular de red neuronal inspirada en el funcionamiento de la corteza visual del cerebro. Estas redes están diseñadas para resolver problemas de visión artificial como el reconocimiento de patrones, aunque pueden tener otros usos como la clasificación de textos o el procesamiento de lenguaje natural.

Al igual que las redes neuronales "convencionales", reciben una entrada que transforman a través de una serie de capas de neuronas, pero en este caso la entrada es una imagen representada en forma de matriz tridimensional dadas por ancho, alto y profundidad de color, siendo esta ultima la que contiene los valores numéricos de los píxeles respecto al tono que maneja. Las capas de neuronas también se organizan de manera tridimensional, y el resultado de las distintas transformaciones llevadas a cabo en la red es la clase o categoría a la que pertenece dicha imagen.

Una CNN (Red Neuronal Convolucional por sus siglas en inglés *Convolutional Neural Network)* es un conjunto de capas de procesamiento, de modo que puede verse como un diagrama secuencial de bloques [10], se obtienen apilando múltiples capas de características. Una capa está formada por *K* filtros lineales seguida de una función de respuesta no lineal.

Se caracterizan por el procesamiento de imágenes, sin embargo, a menudo es ventajoso explotar la estructura de la imagen. Por ejemplo, los píxeles que están muy juntos en la imagen (píxeles adyacentes) tienden a estar fuertemente correlacionados mientras que píxeles que están muy separados en la imagen tienden a ser débilmente correlacionados o no correlacionados [9].

Existen cuatro operaciones principales que conforman las redes convolucionales, estas se describen a continuación.

2.3. Capa convolucional

Las capas convolucionales se entienden como un conjunto de filtros comúnmente llamados campos receptivos [10]. Su principal propósito es extraer características de una imagen. Dicho conjunto de filtros son entrenables y realizan producto punto con los valores de la capa precedente. En la práctica, los valores de los filtros son aprendidos para su activación al encontrar ciertas características. Cuando son colocados en cascada se obtienen diferentes niveles de abstracción [11].

Permiten reducir el número de elementos que conforman la red y detectar características que poseen las imágenes. Considerando lo anterior, se sabe que las capas convolucionales son de gran utilidad en el análisis de imágenes permitiendo reducir la complejidad del sistema y al mismo tiempo, extraer rasgos útiles de las mismas [12].

En la convolución se realizan operaciones de productos y sumas entre la capa de partida y los filtros con lo que se genera un mapa de características. Las características extraídas corresponden a cada posible ubicación del filtro en la imagen original. Este proceso tiene la ventaja de que el mismo filtro puede extraer la misma característica en cualquier parte de la entrada, reduciendo el número de conexiones y de parámetros a entrenar en comparación con una red multicapa de conexión total [13].

Cada capa convolucional consta de 4 parámetros principales:

- Número de filtros N.
- Dimensión M X M: Campos receptivos que recorren toda la imagen de entrada a la capa.
- Stride: Representa el salto de cada filtro, si su valor es 1 saltara de pixel en pixel hasta recorrer toda la imagen.
- Pad: Representa el espacio a partir del cual comienza a recorrer cada campo, este determina el volumen de salida de cada capa [10].

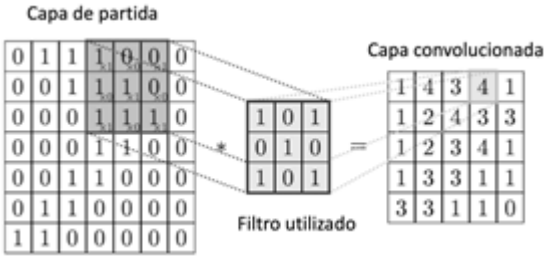


Fig. 1. Muestra el proceso de convolución en una imagen.

De manera general, una capa Convolucional recorre una matriz (pixeles contenidos en la imagen) aplicando un filtro que es una matriz más pequeña, obteniendo las características de la imagen aplicando multiplicaciones, generando una nueva matriz más pequeña que conforme las iteraciones avanzan, se profundiza en las características de la imagen como se muestra en la figura 1.

2.4. Rectificador lineal de unidad

Son utilizados después de cada convolución actuando como de función de activación en términos de redes neuronales tradicionales. Son una operación que reemplaza los valores negativos por cero y su propósito es agregar no linealidad al modelo, eliminando la relación proporcional entre la entrada y salida [11].

Su función matemática se expresa de la siguiente manera:

$$h = max(0, x), \tag{1}$$

donde todos los identificadores inferiores a 0 serán convertidos en 0 mientras los positivos se conservan [10].

El comportamiento por defecto y más habitual es que mientras la entrada tenga un valor por debajo de cero, la salida será cero, pero cuando la entrada se eleva por encima, la salida es una relación lineal con la variable de entrada de la forma $f(x) = x$. La función de activación ReLU ha demostrado funcionar en muchas situaciones diferentes, y actualmente es muy usada, gráficamente se puede observar en la figura 2.

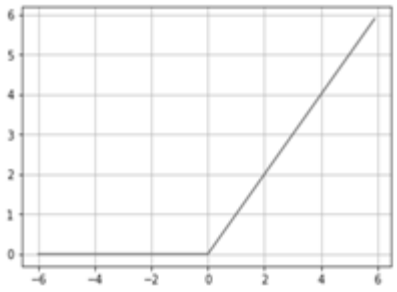


Fig. 2. Muestra el comportamiento de la función de activación gráficamente.

2.5. *Pooling*

Es la capa utilizada para reducir las dimensiones, con el objetivo de disminuir los tiempos de procesado reteniendo la información más importante [11], a su vez consigue reducir el sobreajuste (*overfitting*) en la red y algo de invarianza a la traslación. Como se muestra en la siguiente imagen la capa *pooling* se comporta conservando el valor mayor tras aplicar un filtro con una matriz, con lo que se crea una nueva matriz.

En la figura 3 se muestra una matriz de 4 X 4 que ejemplifica lo que se conoce como capa de partida misma que es un estado posterior a pasar por la capa convolucional, el filtro con el que se evaluó es de un tamaño 2 X 2, es decir la imagen se reducirá en una de 2 X 2 pixeles al ya tener la capa de reducción siendo evaluadas en total 4 secciones en la capa previa.

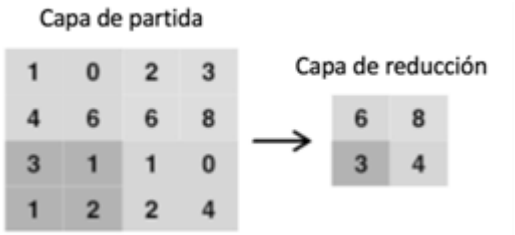


Fig. 3. Muestra el funcionamiento de la capa de reducción (*Pooling*).

2.6. Capa totalmente conectada (*Dense Layer*)

Capa totalmente conectada realiza la clasificación basada en las características extraídas por las capas de convolución y las reducidas por pooling. En esta capa todos los nodos están conectados con la capa precedente [11].

En la figura 4 se muestra el funcionamiento de esta capa, la primera capa de entrada es el resultante de la última reducción por parte de la capa de *pooling*, la capa oculta funciona como clasificador, a partir de la cual se encuentran las incidencias que determinen cual es el resultado de la clasificación, por último, se encuentran la capa de salida.

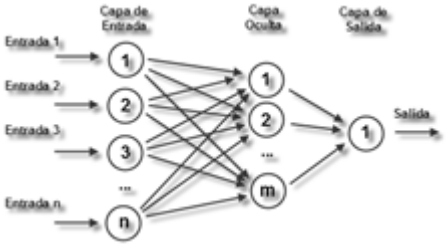


Fig. 4. Representación gráfica de una capa clasificatoria.

2.7. Funcionamiento de la red convolucional

Las capas de convolución y las de *pooling* se encargan de extraer características mientras que la capa totalmente conectada actúa como clasificador. Para el funcionamiento de este modelo debemos proceder al entrenamiento.

Esto implica:

1) Inicializar todos los parámetros o pesos con valores aleatorios.
2) Utilizar una imagen de entrenamiento y utilizarla en el modelo.
3) Calcular el error total de las probabilidades resultantes del modelo.
4) Propagar hacia atrás para calculare el error de gradiente de todos los pesos en la red y utilizar gradiente descendiente para actualizar estos valores y minimizar el error de salida [11].

2.8. Evaluación del nivel de clasificación en una red neuronal

Métrica Accuracy. Esta métrica trabaja en relación con la función que se utiliza para juzgar el rendimiento del modelo, matemáticamente se define de la siguiente manera:

$$\frac{TP + TN}{TP + TN + FP + FN} (\hat{P}), \tag{2}$$

donde TP son los Positivos Verdaderos (Clasificaciones esperadas durante el entrenamiento), FP son Falsos Positivos (Cuando se clasifica como perteneciente a una categoría a la que no pertenece), TN Negativos Verdaderos (Cuando un elemento perteneciente no es clasificado donde se esperaba) y FN Falsos Negativos (Cuando un elemento no perteneciente no se clasifica en alguna clase a la que no pertenece) y $\hat{P}$ proporción de aciertos definida de la siguiente manera: s/n, donde s es el número de aciertos obtenidos y n se refiere al número de pruebas realizadas (de iteración del algoritmo) [14].

Función de perdida Categorical Crossentropy. Una función de pérdida (función objetivo, o función de puntuación de optimización) es uno de los dos parámetros necesarios para compilar un modelo; cuando usa la función `categorical_crossentropy`, sus objetivos deben estar en formato categórico (por ejemplo, si tiene 10 clases, el objetivo para cada muestra debe ser un vector de 10 dimensiones que sea todo ceros, excepto un 1 en el índice correspondiente a la clase de la muestra). Para convertir *objetivos enteros en objetivos categóricos.*

Matemáticamente, la función `categorical_crossentropy` está definida entre una distribución aproximada y una distribución verdadera; esto quiere decir que mide el número promedio de bits necesarios para identificar un evento de un conjunto de posibilidades, si se usa un esquema de codificación basado en una distribución de probabilidad dada por q, en lugar de la distribución "verdadera" p expresada en la siguiente ecuación:

$$H(p,q) = -\sum_{x} p(x)\log\big(q(x)\big), \tag{3}$$

donde p está definido por una condición OR que contiene un vector dado de 1 a N donde en cada elemento representa la posición dada al valor "1", q representa las filas para la distribución y x es la entrada recibida [15].

3. Marco metodológico

Previo a la implementación del algoritmo de red se pensó en cómo debería estar estructurada la CNN a implementar por lo que se optó por el modelo que se muestra en la Fig. 5.

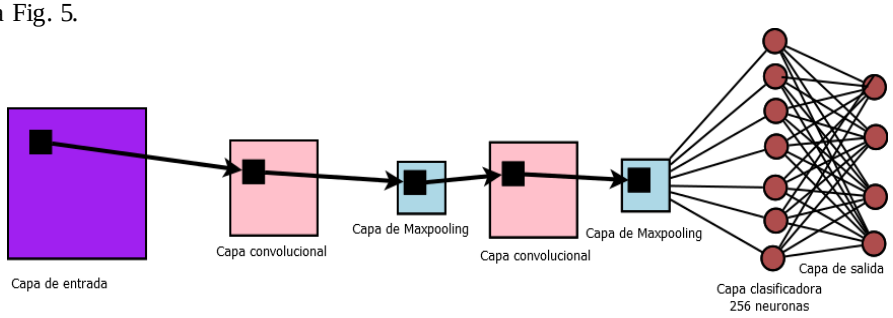


Fig. 5. Muestra la arquitectura de la red neuronal a utilizar.

Basándonos en la imagen anterior, se describirá a detalle el algoritmo a seguir para conseguir una correcta clasificación.

Capa de entrada. Esta capa mostrada en la figura anterior identificada con color morado es equivalente a las imágenes que se contendrán en el Corpus, además se describen algunos pasos de procesamiento de estas a fin de obtener una mejor clasificación, esto con ayuda de Keras.

Procesamiento de imagen con Keras

Tamaño estandarizado. A fin de reducir la carga de trabajo al procesador se redujo el tamaño de todas las imágenes que entraran al algoritmo a un tamaño de 150 X 150 pixeles.

Función rescale. Considerando que los valores posibles en un formato de color RGB son de 0 a 255, se implementó esta función con la finalidad de reducir el amplio rango que se puede obtener de tonalidad entre un píxel y otro, dándoles valores que van de 0 a 1 según el tono que adquiera cada pixel. Esto se muestra en la figura 6.

Fig. 6. Ejemplo de cambio de tono con la función Rescale.

Función shear_range. Esta ayuda a que aleatoriamente se seleccionen imágenes para inclinar su posición, esto para tener un mayor rango de clasificación y a su vez tener en cuenta que es posible que exista variación dada por el medio donde se tome o por la posición de la cámara. Esto se muestra en la figura 7.

Fig. 7. Ejemplos de la función shear_range.

Función zoom_range. selecciona aleatoriamente algunas entradas a fin de hacer un acercamiento que permita tener más muestras distintas a partir de una imagen. Como se muestra en la figura 8.

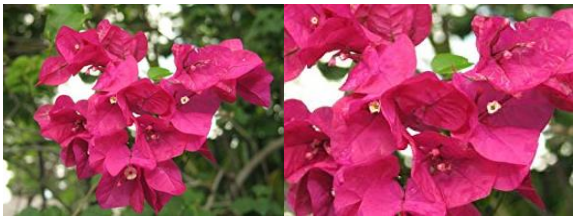


Fig. 8. Ejemplo en el que se muestra la aplicación de la función de zoom_range en la buganvilia.

Función horizontal_flip y vertical_flip. Estas ayudan a invertir la posición de la imagen para que cuando al algoritmo se le muestre incluso una imagen de cabeza pueda identificar la especie de planta a la que pertenece como se muestra en la figura 9.

Fig. 9. Muestra el reposicionamiento que sufren las imágenes previo a ser introducidas a la red neuronal para su entrenamiento.

Función flatten. Sirve para aplanar la imagen, es decir darle a la imagen el formato necesario para trabajar como una capa totalmente conectada que se pueda trabajar con

la capa Convolucional que la recibirá, es decir, se le indica al algoritmo que se está trabajando con imágenes 2D.

Capas convolucionales. En la figura 5. estas capas se identifican con color rosa, la primera de ellas cuenta con un tamaño de 3 X 3 pixeles mismos que irán recorriendo la matriz de la capa de entrada; este proceso se realizara en 32 iteraciones por entrada recibida.

La segunda capa consta de un tamaño de 2 X 2 pixeles, en este caso se convolucionara la matriz obtenida por la capa de reducción; esta realizará su función durante 64 iteraciones por entrada recibida.

Como producto de cada capa convolucional, se generará una nueva matriz por cada una, donde en base a la matemática expuesta en el marco teórico, se irán resaltando las características de cada imagen.

Capas de pooling (Maxpooling). Se propuso utilizar dos capas de este tipo, cada una después de una capa convolucional, se busca con ello evitar el sobre ajuste de la red y a su vez reducir la carga computacional que aumenta a medida que se avanza en el algoritmo.

Para esto se hizo uso de la función MaxPooling con la que en base a una matriz de tamaño 2 X 2 pixeles, se toma el valor más relevante que contenga ese espacio de pixeles y se envía a una nueva matriz.

La primera de estas capas se ubica después de la primera convolución y la otra después de la segunda convolución; esta última se toma como entrada para la capa clasificadora que se describirá a continuación.

Capa de clasificación. Una vez que ya se han obtenido las características más sobresalientes en las capas anteriores, se procede a ir separando categóricamente cada una de las imágenes con las que se ha de entrenar el algoritmo. Para esto se implementó esta capa que consta de 256 neuronas.

A fin de evitar el sobre entrenamiento de la red, se desactivan aleatoriamente el 50% de las neuronas utilizadas; con ello se busca que la red no se acostumbre a seguir un solo camino para la clasificación obligándola a alternar rutas para llegar al mismo resultado.

Además de esto con Keras se le agrega la función Softmax, misma que le dice al algoritmo que tiene que buscar la probabilidad más alta de que la imagen analizada pertenezca a una categoría en específico.

Capa de salida. Esta última capa está definida como el número de clases con la que se está realizando la clasificación, es decir, se tendrán tantas capas de salida como especies de plantas que se deseen clasificar.

Entrenamiento de la red neuronal convolucional

La arquitectura definida anteriormente se implementó en el lenguaje de programación Python. A fin de obtener una clasificación optima se incluyeron los siguientes parámetros:

- Número de épocas: En total el algoritmo recorrerá 20 veces el corpus de imágenes.

- Número de Iteraciones: Se definió un estándar de 1000 iteraciones por épocas y por cada iteración se analizan 32 imágenes.
- Validación del algoritmo: Al final de cada época se realizan 300 iteraciones donde se obtienen del directorio de validación las imágenes que contiene a fin de determinar qué tan eficiente ha sido la clasificación.

Ahora bien, se utiliza la función *categorical_crossentropy* a fin de minimizar la función de perdida y se registra el porcentaje de aciertos en la clasificación aplicando la métrica *accuracy* para saber cómo está aprendiendo la CNN. Para la actualización de pesos se utiliza el optimizador Adam con una tasa de aprendizaje de 0.0005.

Solo resta utilizar la función *save* para guardar el modelo y la función *sabe_weigths* para guardar los pesos, estos se guardaron con extensión .h5 para poder consumirlos posteriormente en una aplicación.

Con el modelo de la Red Neuronal queda ejecutar el script en Python y esperar el modelo entrenado como se muestra en la figura 10. donde se ve la época en que está entrenando y el número de iteraciones transcurridas, además de mostrar el valor de la función de perdida (variable loss) y el porcentaje de aciertos (variable acc) dado por la métrica accuracy, como podemos notar en la primera iteración el valor es bastante elevado para la función de perdida y el porcentaje de aciertos bastante bajo-mismos que mejoraran sus números con el avanzar de las épocas.

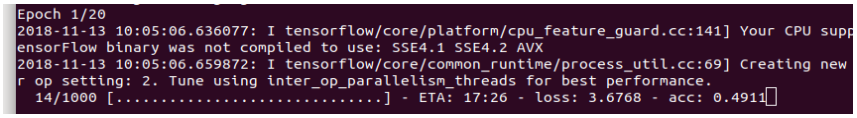


Fig. 10. Entrenamiento de la red.

4. Pruebas y resultados

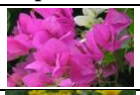
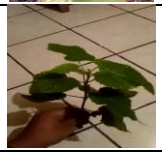
Para probar el modelo se desarrolló una aplicación de consola con el lenguaje de programación Python y con ayuda de Keras, en la que se consumieron los archivos de modelo.h5 y pesos.h5.

- Las predicciones se guardan en un arreglo cuya dimensión es equivalente al número de especies a clasificar (En este caso 11).
- La clasificación obtenida se expresa con un "1" y los demás espacios del vector se rellenan con "0", donde la posición del vector es equivalente al orden establecido por los directorios que contienen las imágenes.
- Se evaluaron el número de clasificaciones correctas obtenidas y se modificó el algoritmo hasta obtener la mejor clasificación posible.

A fin de determinar que la clasificación realizada por el algoritmo era aceptable, se realizó la siguiente prueba cargándole imágenes diferentes a las utilizadas durante el entrenamiento por lo que se obtuvieron los siguientes resultados (ver Tabla 1).

Se desarrolló una interfaz web a partir de la cual se pueden visualizar los resultados obtenidos por el clasificador de manera clara como se muestra en la Fig. 11. donde se aprecia una sencilla interfaz con la que iniciaremos la búsqueda al pulsar el botón de búsqueda intensiva ubicado en la parte superior.

Tabla 1. Resultados de las pruebas a las que se sometió el algoritmo.

Planta	Imagen de prueba	Resultado obtenido
Buganvilla		pred: Bugambilia [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Árnica		pred: Arnica [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
Lengua de suegra		pred: Lengua [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
Manzanilla		pred: Manzanilla [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
Mastuerzo		pred: Mastuerzo [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
Ruda		pred: Ruda [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
Yerba de sapo		pred: Y. de sapo [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
Sauco		pred: Sauco [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
Savile		pred: Savila [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
Siempreviva		pred: Siempreviva [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
Yerba de burro		pred: Y. de burro [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]

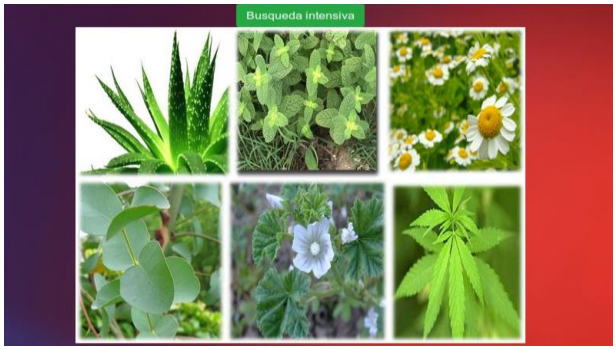


Fig. 11. Muestra la pantalla de bienvenida al clasificador.

Posteriormente se empiezan a tomar fotos y se van almacenando las incidencias hasta que finalmente se nos envía un resultado de la búsqueda (imagen izquierda) contrastada con una imagen precargada (derecha) donde además se describe de manera sencilla la planta identificada como se muestra en la figura 12.

Fig. 12. Se muestra la interfaz con resultado posterior a haber determinado que se ha encontrado una sábila.

En la parte del backend el programa ejecuta dos acciones, la primera es tomar fotos del entorno haciendo uso de la librería OpenCV y la segunda es identificar la planta ejecutando el algoritmo de la red neuronal explicado en la sección del procedimiento; lo anterior se hace mediante la ejecución de dos hilos que trabajan simultáneamente, mientras con uno se toman muestras del entorno el otro las toma para su análisis (ver Fig. 13.).

Fig. 13. Acciones realizadas por el software.

En cuanto al grado de acierto del clasificador se han monitorizado 2 valores durante el entrenamiento de la red neuronal que son la función de perdida y la métrica accuracy donde nuestra función de perdida obtuvo un valor del 0.4196 siendo el mejor valor obtenido debido a que en los primeros experimentos superaba las 3 unidades según la métrica implementada, nuestro clasificador identifica con un 84% de eficiencia como se muestra en la figura 14.

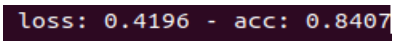


Fig. 14. Resultados obtenidos de la función de perdida y la métrica implementada.

5. Conclusión y trabajos futuros

A continuación, se muestra la Tabla 2. En ella se realiza la comparativa de los resultados obtenidos con trabajos analizados en el estado del arte.

El primero es el resultado de esta investigación con 84.07% de clasificación, con 11 plantas analizadas de formas completa, el segundo tiene una clasificación de 79.60 % con 27 especies analizando únicamente las hojas.

El tercer proyecto muestra un rango de efectividad de 93.08%, sin embargo, este proyecto se enfoca únicamente en la identificación mediante hojas.

El último proyecto muestra un porcentaje de clasificación del 92.64 % trabajando únicamente con la planta de café y su madurez.

Tabla 2. Comparación del trabajo realizado y similares.

Clasificador	Porcentaje de acierto
Clasificador de plantas medicinales en la región de Jocotitlán	84.07%
Leaves recognition system using a neural network	79.60%
Neural Network Application on Foliage Plant Identification	93.08%

En conclusión, el trabajo propuesto evidencia una mejor clasificación, no obstante, como trabajo futuro se pretende mejorar haciendo uso de más plantas medicinales y posteriormente implementarlo en un robot recolector de basura.

Referencias

1. Calvo, D.: http://www.diegocalvo.es/red-neuronal-convolucional-cnn/, último acceso: 2018/10/18
2. Luna González, J.A., Paris, S., Nakano Miyatake M., Robles Camarillo D.: Comparación de Arquitecturas de Redes Neuronales Convolucionales para la Clasificación de Imágenes de Ojos. En: Simposio Iberoamericano Multidiciplinario de Ciencias e Ingenieria., vol. 1, no. 1, pp. 94 –101 (2016)
3. Antea Herrera, A.: Detección de texto utilizando Redes Neuronales Convolucionales. Cataluña: Universidad Politécnica de Cataluña (2015)

4. Quintero, C., Merchán, F., Cornejo, A.: Uso de Redes Neuronales Convolucionales para el Reconocimiento Automático de Imágenes de Macro invertebrados para el Biomonitoreo Participativo. Conference Paper, vol. 2018, nº 1, pp. 585–596 (2017)
5. Durán Suárez, J.: Reconocimiento de caracteres escritos a mano. Sevilla: Universidad de Sevilla (2017)
6. Puy, P.: Facultad de Matemática, Astronomía y Física, Universidad Nacional de Cordoba, http://www.famaf.proed.unc.edu.ar/pluginfile.php/19400/mod_resource/content/1/02.evaluacion.pdf, último acceso: 2018/12/10
7. Hernandez Avila, R.: Deep Learning. Una revisión. Innovación y Gestión del Conocimiento , vol. 1, no. 1, pp. 2–7 (2018)
8. Núñez Sánchez, A.: Diseño de un sistema de reconocimiento automático de matrículas de vehículos mediante una Red Neuronal Convolucional. Cataluña: Universidad Oberta de Cataluña (2016)
9. Matich, D.J.: Redes Neuronales: Conceptos Básicos y Aplicaciones. Universidad Tecnológica Nacional, vol. 1, nº 1, pp. 10–32 (2001)
10. Swain, M., Dash, S. K., Dash, S., Mohapatra, A.: An approach for iris plant classification using neural network. International Journal on Soft Computing, vol. 3, no. 1, pp. 79–89 (2012)
11. Pitarque, A., Ruiz, J.C., Roy, J.F.: Las redes neuronales como herramientas estadísticas no paramétricas de clasificación. Psicothema, vol. 12, no. 2, pp. 459–463 (2000)
12. Gavarini, S., Trípode, M., Tosini, M., Ceccatto, A.: Clasificación de semillas de malezas utilizando Redes Neuronales Artificiales. Instituto Computación aplicada (INCA), vol. 1, no. 1, pp. 1–5 (2010)
13. Sekeroglu, B., Inan, Y.: Leaves recognition system using a neural network. In: 12th International Conference on Application of Fuzzy Systems and Soft Computing, ICAFS, Vols. 1877-0509, no. 102, pp. 579–592 (2016)
14. Gang Wu, S., Sheng Bao, F., You Xu, E., Wang, Y. X., Chang, Y. F., Xiang, Q. L.: A Leaf Recognition Algorithm for Plant Classification Using Probabilistic Neural Network. Dept. of Computer Science, Texas Tech University, vol. 1, no. 0707-4289, pp. 1–5 (2007)
15. Grajales-Múnera, J.E., Restrepo-Martínez A.: Clasificación de Mariposas por Modelos de Color HSI y RGB Usando Redes Neuronales. Tecnológicas, vol. 1, no. 0123-7799, pp. 669–679 (2013)