UNIVERSIDAD PERUANA UNIÓN

FACULTAD DE INGENIERÍA Y ARQUITECTURA

Escuela Profesional de Ingeniería de Sistemas

Sistema Experto Basado en Visión Artificial para el Diagnóstico de Plagas de Cacao en la Región San Martín.

Tesis para obtener el Título Profesional de Ingeniero de Sistemas

Autor:

Oswaldo Chuquipoma Fermín
Jared Elim Arapa Mejia

Asesor:

Dr. Miguel Ángel Valles Coral

Morales, Julio de 2023

DECLARACIÓN JURADA DE AUTORÍA DE TESIS

Yo, *Dr. Miguel Ángel Valles Coral*, de la Facultad de Ingeniería y Arquitectura , Escuela Profesional de Ingeniería de Sistemas, de la Universidad Peruana Unión.

DECLARO:

Que la presente investigación titulada: **“Expert System Based on Artificial Vision for the Diagnosis of Cocoa Pests in the San Martin Region.”** constituye la memoria que presenta el (la) / los Bachiller(es) Oswaldo Chuquipoma Fermín, Jared Elim Arapa Mejia para obtener el título de Profesional de Ingeniero de Sistemas cuya tesis ha sido realizada en la Universidad Peruana Unión bajo mi dirección.

Las opiniones y declaraciones en este informe son de entera responsabilidad del autor, sin comprometer a la institución.

Y estando de acuerdo, firmo la presente declaración en la ciudad de Morales, a los 20 días del mes de Julio del año 2023

Miguel Ángel Valles Coral

ACTA DE SUSTENTACIÓN DE TESIS

En San Martín, Tarapoto, Morales, a 20 día(s) del mes de julio del año 2023 siendo las 11:00 horas, se reunieron los miembros del jurado en la Universidad Peruana Unión Campus Tarapoto, bajo la dirección del (de la) presidente(a): Mg. Danny Lévano Rodríguez, el (la) secretario(a): Mg. Marco Antonio Ruiz Grandez y los demás miembros: Mg. Yngue Elizabeth Ramirez Pezo y el (la) asesor(a) Dr. Miguel Angel Valles Coral con el propósito de administrar el acto académico de sustentación de la tesis titulado: **Sistema experto basado en visión artificial para el diagnóstico de plagas de cacao en la región San Martin** del(los) bachiller/es: a) Oswaldo Chuquipoma Fermín b) Jared Elim Arapa Mejía c) conducente a la obtención del título profesional de: Ingeniero de Sistemas

(Denominación del Título Profesional)

El Presidente inició el acto académico de sustentación invitando al (a la) / a (los) (las) candidato(a)/s hacer uso del tiempo determinado para su exposición. Concluida la exposición, el Presidente invitó a los demás miembros del jurado a efectuar las preguntas, y aclaraciones pertinentes, las cuales fueron absueltas por al (a la) / a (los) (las) candidato(a)/s. Luego, se produjo un receso para las deliberaciones y la emisión del dictamen del jurado.

Posteriormente, el jurado procedió a dejar constancia escrita sobre la evaluación en la presente acta, con el dictamen siguiente:

Bachiller-(a): Oswaldo Chuquipoma Fermín

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	
Aprobado	18	A-	Muy Bueno	Sobresaliente

Bachiller -(b): Jared Elim Arapa Mejía

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	
Aprobado	18	A-	Muy Bueno	Sobresaliente

Bachiller -(c):

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	

(*) Ver parte posterior

Finalmente, el Presidente del jurado invitó al (a la) / a (los) (las) candidato(a)/s a ponerse de pie, para recibir la evaluación final y concluir el acto académico de sustentación procediéndose a registrar las firmas respectivas.

Presidente/a

Secretario/a

Asesor/a

Miembro

Miembro

Bachiller (a)

Bachiller (b)

Bachiller (c)

Resumen. En San Martin el cultivo de cacao (Theobroma cacao L.) se ve afectado por enfermedades y plagas que pueden causar una disminución en su producción e incluso la pérdida total de la cosecha. Por esta razón en esta investigación se desarrollaron modelos de redes neuronales convolucionales para identificar de manera temprana las plagas en frutos de cacao (Teobroma cacao L.), para este proyecto nos centramos en la plaga Moniliasis (Moniliophthora roreri) utilizando imágenes simples de frutos sanos y enfermos, mediante técnicas de aprendizaje profundo. Los modelos fueron entrenados con un conjunto de 4602 imágenes obtenidas de Kaggle, una comunidad en línea compuesta por científicos de datos y profesionales del aprendizaje automático, que incluyen imágenes de frutos enfermos y sanos. Se entrenaron dos arquitecturas modelos: CacaoCustom, un modelo personalizado que alcanzo una precisión de 82%, e InceptionV3 un modelo preentrenado, siendo este último con el mejor rendimiento, alcanzando con una tasa de éxito de 96.77% en la identificación del fruto con plaga y sin plaga. La tasa de éxito significativamente alta hace que el modelo sea una herramienta de alerta temprana y muy útil en la agricultura, identificando la plaga Moniliasis en los frutos del cultivo de cacao (Teobroma cacao L.).

Palabras clave. Agricultura, Inteligencia Artificial, CNN, Aprendizaje Profundo.

Abstract. In San Martin the cocoa crop (Theobroma cacao L.) is affected by diseases and pests that can cause a decrease in production and even the total loss of the crop. For this reason, in this research, convolutional neural network models were developed to identify early the pests in cocoa fruits (Theobroma cacao L.), for this project we focused on the pest Moniliasis (Moniliophthora roreri) using simple images of healthy and diseased fruits, using deep learning techniques. Models were trained with a set of 4602 images obtained from Kaggle, an online community composed of data scientists and machine learning practitioners, including images of diseased and healthy fruits. Two model architectures were trained: CacaoCustom, a customized model that achieved an accuracy of 82%, and InceptionV3, a pre-trained model, the latter being the best performing, achieving a success rate of 96.77% in identifying fruit with and without pests. The significantly high success rate makes the model an early warning tool and very useful in agriculture, identifying the pest Moniliasis in the fruits of the cocoa crop (Teobroma cacao L.).

Keywords. Agriculture, artificial intelligence, CNN, Deep Learning..

1 Introducción

Las plagas agrícolas suelen ser desencadenantes de daños y pérdidas en las plantaciones. Por eso, la prevención y el diagnóstico oportuno de cualquier enfermedad traerá ventaja; pues al identificar la plaga que sufre el cultivo, incrementamos las posibilidades de éxito en el tratamiento. A nivel mundial, las pérdidas de producción agrícola representan el 95% de las pérdidas totales, siendo las más comunes las que ocurren antes de la cosecha debido a plagas y enfermedades. Estas pérdidas se producen durante la etapa de producción[9].

Existen diversos métodos para el diagnóstico de enfermedades en plantas, ya sea con una muestra, o con un ingeniero experto en el campo; en cualquiera de los dos métodos, la desventaja es el tiempo para obtener resultados. A nivel mundial se implementa agricultura de precisión que, junto a la inteligencia artificial, mejora la calidad de la producción agrícola, y reduce costos. El principal beneficio es que pronostican basándose en entrenamiento, llegando a ser muy precisos y rápidos [3]

En Palestina analizaron 9,000 imágenes de hojas de tomate, para producir un modelo que se use en teléfonos inteligentes, el cual identifica 5 tipos de enfermedades; basado en una red convolucional profunda, compuesta por dos partes: funciones de extracción para enfoque a todo color y de escala de grises, con 4 capas convolucionales y función de activación ReLU, seguida por una capa Max Pooling, la segunda: con dos capas densas de contención para los enfoques. El estudio dio resultados a color de 99.84% respecto a escala de grises con un 95.54% [1].

En Lambayeque, Perú, se desarrolló un aplicativo móvil con un sistema de visión artificial para el diagnóstico de plagas en cultivos de sandía, que identifica patrones en las imágenes de forma rápida y confiable. El software busca en repositorios registrados, una vez capturada la imagen, se procesa en busca de patrones identificando qué tipo de plaga ataca al cultivo, apoyando en el diagnóstico, con intención de suministrar el agroquímico correcto [10]

En la región San Martín la producción de diversos productos agrícolas como arroz, café, maíz, plátano ha dado impulso económico a la región, traduciéndose esto en una mayor

comercialización, un aumento de tasa de empleo, productividad de las MYPES [17] entre otras, que han ubicado a esta región como una de las más productivas que se ha denominado como “El milagro San Martín”. De todos los productos agrícolas producidos y comercializados en esta región, el cacao es el segundo que más impulso ha traído a la región[17]y que tanto su producción como comercialización implica una importante fuente de ingresos y desarrollo para la región.

Sin embargo, muchos de los agricultores involucrados en esta tarea han encontrado que la deficiencia en la identificación de las características visuales presentes para el diagnóstico de plagas en el cultivo del cacao, les han ocasionado bajos rendimientos y pérdidas en la producción de este cultivo. Esto se debe al desconocimiento de las características visuales de las plagas en frutos de cacao, al no identificar exactamente qué plaga está afectando el cultivo, por lo que se aplican diversos agroquímicos, dañando la planta o, en su caso, no deteniendo la plaga (o siendo ineficientes en el control de la plaga) [6]

Existe un alto desconocimiento en los procesos al diagnosticar las plagas presentes en el cacao, afectando el rendimiento en calidad y producción. Las metodologías tradicionales que se aplican no son muy favorables para el agricultor, que solo se dedica a atacar la causa visible de los efectos provocados por las plagas visibles, perjudiciales, ignorando o minimizando el potencial que tienen los agentes de control biológico en el agro u otro tipo de herramientas amigables que resulten favorables para contrarrestar plagas [7].

El uso de la tecnología para la agricultura es un aspecto clave para una mejora de manera acelerada de la actividad; la aplicación de nuevas herramientas permite mejorar los problemas actuales en el manejo de los cultivos y procesos agrícolas, sin embargo, la adaptación de procesos tecnológicos está limitada por muchos agricultores de nuestro país; debido al conocimiento y aplicación de estos en cultivos de cacao (Theobroma cacao L.). Es importante que en los cultivos de cacao (Theobroma cacao L.), se logre la implementación de nuevas tecnologías, para enfrentar y prevenir las diferentes plagas que afectan la productividad de este cultivo [11]

El alcance de esta investigación solo abarca la identificación del fruto de cacao con plaga, más no brindará el detalle de la plaga, no dará consejos de cómo se debe tratar la plaga, ya que se pueden usar diversos métodos y/o agroquímicos para combatirlas, lo que se

quiere es el rápido y preciso diagnóstico para su pronta aplicación de tratamientos. (Servirá de prototipo para proyectos futuros en el cultivo cacao).

Es por ello qué, se desarrolló un algoritmo basado en inteligencia artificial, con redes neuronales artificiales, acompañada de patrones y algoritmos, para que de forma automática y precisa diagnostique la plaga en el fruto de cacao, ayudando así al agricultor a combatirlas de forma temprana.

1.1 Estado de arte

Teniendo en cuenta que la tecnología llegará a todos los ámbitos en el sector agrícola, ya sea desde el empleo de drones para el cultivo, hasta la utilización de IA en el monitoreo y control de plagas.

En la India, Khan S. ha propuesto la utilización de arquitecturas de redes neuronales convolucionales profundas, específicamente VGGNet [12] y AlexNetOWTBn [16], para automatizar el proceso de diagnóstico de enfermedades en las hojas de tomate. Las enfermedades que se abordaron en la clasificación incluyen el tizón temprano, oidio y mildiu. Las imágenes son sometidas a procesos de pre-procesamiento mediante técnicas de manipulación que contribuyen a la reducción de costos y tiempo de cálculo. Luego, se extrajeron características del conjunto de datos mediante mapas de convolución aplicados a las imágenes de hojas sanas e infectadas. Aunque esta arquitectura ha logrado resultados precisos en términos de datos, para este estudio en particular los porcentajes obtenidos fueron 32.23% utilizando la arquitectura AlexNetOWTBn y 33.27% para VGG [4].

En un estudio llevado a cabo en Ecuador por Yandún Velasteguí y su equipo de la Universidad Politécnica Estatal del Carchi, se investigó la detección de enfermedades en los cultivos de papa mediante el uso de procesamiento de imágenes. El enfoque adoptado involucra una combinación de distintas técnicas de procesamiento en un sistema de identificación y clasificación de enfermedades. Este sistema permite discernir las fases de la Alternariosis, una enfermedad que es común en los cultivos de papa. Sin embargo, debido a la amplia variedad de enfermedades que pueden afectar este cultivo, se optó por abordar los tres tipos de plagas más frecuentes en el mismo: Alternariosis (Alternaria solani), Tizón Tardío (Phytophthora infestans) y Virosis (PVS).

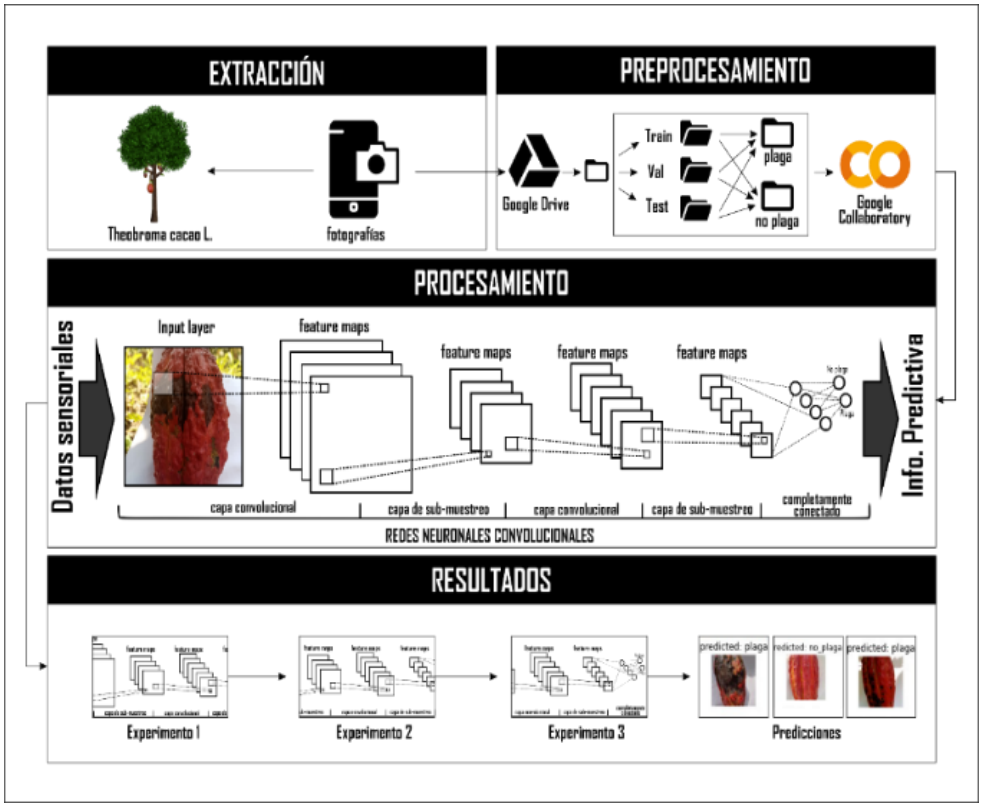
El proceso se detalla en cinco etapas: adquisición de la imagen, preprocesamiento de la imagen, segmentación de la imagen, extracción de características y clasificación. Para

llevar a cabo esta clasificación, se utiliza una red neuronal artificial (ANN, por sus siglas en inglés) que analiza la imagen capturada de la hoja de papa. A medida que la imagen pasa por cada una de las cinco etapas mencionadas, en la última etapa, gracias al uso de la red neuronal, se compara con miles de fotos de hojas enfermas almacenadas en la base de datos. Esto permite a la red neuronal clasificar a qué categoría pertenece y establecer una semejanza con el tipo de enfermedad, con el objetivo de proporcionar resultados precisos [15].

Singh y Misra llevaron a cabo experimentos con el propósito de lograr la detección de enfermedades o daños presentes en las hojas de diferentes cultivos, como plátano, frijol, limón y rosas. Una vez que las imágenes fueron procesadas, sugieren aplicar la segmentación a través de algoritmos genéticos, seguido por el proceso de agrupamiento. Para extraer las características necesarias, optaron por el enfoque de concurrencia de color, considerando más efectivo utilizar imágenes a color en lugar de la tradicional escala de grises.

Para la tarea de clasificación, emplearon el algoritmo MDC en combinación con K-Means, lo que arrojó un resultado del 86.54%. Además, introdujeron un algoritmo propio en el proceso de MDC, lo cual mejoró el rendimiento al alcanzar un 93.63%. Asimismo, experimentaron con el método de Máquinas de Soporte Vectorial (SVM), obteniendo un aumento notable en la precisión, llegando a un 95.71%. Estos porcentajes representan un promedio general de rendimiento para los cuatro tipos de cultivos estudiados [13].

En Perú, se llevó a cabo un estudio que se enfocó en detectar enfermedades y plagas en las hojas de arándanos. Para lograr esto, se emplearon dos enfoques convencionales de Aprendizaje Profundo: Random Forest (RF) y Support Vector Machines (SVMs). El proceso comenzó al utilizar fotografías como punto de partida para ingresar datos en una base de datos. Se recolectaron imágenes de diversas variedades de plantas de arándanos. Estas imágenes luego se sometieron a una fase inicial de preprocesamiento, en la cual se realizaron acciones como reducción de ruido y realce de detalles. Posteriormente, se procedió a extraer características clave, evaluando elementos como el tamaño, la forma y el color de las hojas. En la cuarta etapa, las imágenes se sometieron a un proceso de clasificación a través de algoritmos específicos. Luego, se avanzó a una fase de reconocimiento y, finalmente, se obtuvieron los resultados deseados. Estos resultados permitieron la identificación de las plagas presentes y la formulación de recomendaciones médicas. Los resultados del modelo mostraron una



precisión del 85.6%, lo que demuestra la idoneidad y eficacia del modelo utilizado [14].

Fig 1. modelo propuesto. La figura 1 ilustra el modelo propuesto que consta de 4 etapas.

2. Materiales y Métodos

La investigación fue aplicada, con un diseño no experimental, con nivel descriptivo en la que nos enfocamos en aplicar tecnologías de aprendizaje profundo para la detección de plagas en cultivos a través de imágenes.

Empleamos 4602 imágenes de frutos de Theobroma cacao L con plaga y sin plaga, utilizamos el conjunto de datos 'Moniliasis Detection Dataset' recopilado por Zaldy Jr. Pagaduan en Kaggle. Este conjunto de datos proporcionó imágenes de frutos afectados y no afectados, permitiéndonos entrenar nuestro algoritmo de aprendizaje automático para identificar patrones distintivos y lograr una detección precisa de la plaga.

Propusimos un modelo de reconocimiento de imágenes basado en Deep Learning que consta de los siguientes pasos: extracción, preprocesamiento, procesamiento y resultados, para clasificar imágenes según su contenido. Para ello, realizamos técnicas de extracción de características, preprocesamiento de imágenes y procesamiento de datos utilizando una red neuronal convolucional [2].

Para el entrenamiento del algoritmo, utilizamos el entorno de desarrollo Google Colab, aprovechando sus recursos de hardware y software. Las especificaciones del entorno que utilizamos incluyen 12.7 GB de memoria RAM, 15 GB de memoria RAM de GPU y un disco de 166.8 GB de almacenamiento. Además, utilizamos Google Drive como almacenamiento para el conjunto de datos, permitiendo un acceso remoto y seguro a los archivos. Utilizamos estas herramientas para garantizar la calidad y reproducibilidad del modelo de clasificación desarrollado mediante técnicas de aprendizaje automático.

Los datos experimentales se organizaron en dos carpetas y se subdividieron en tres subcarpetas: Train, Val y Test. La subcarpeta Train contiene el conjunto de datos de entrenamiento utilizado para ajustar el modelo. La subcarpeta Val incluye el conjunto de datos de validación, utilizado para evaluar de manera imparcial el ajuste del modelo durante el entrenamiento. La subcarpeta Test contiene los datos de prueba utilizados para evaluar de manera imparcial el ajuste final del modelo en los datos de entrenamiento, esta metodología es usada como estructura estándar en el entrenamiento de algoritmos con imágenes.

```
seed = 1
tf.random.set_seed(seed)
np.random.seed(seed)

train_data_dir = 'dataset/train'
validation_data_dir = 'dataset/val'
test_data_dir = 'dataset/test'
img_width, img_height = 256, 256
input_shape = (img_width, img_height, 3)

train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
validation_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_dataset =
train_datagen.flow_from_directory(train_data_dir,
                                  target_size=(img_width,
img_height),

batch_size=32,class_mode='categorical')
validation_dataset =
validation_datagen.flow_from_directory(validation_data_dir,
                                   target_size=(img_width,
img_height),

batch_size=32,class_mode='categorical')

test_dataset = test_datagen.flow_from_directory(test_data_dir,
                                  target_size=(img_width,
img_height),
                    batch_size=32,class_mode='categorical',
```

```
shuffle=False)
```

Preprocesamiento

El proceso de tratamiento de datos fue llevado a cabo utilizando un entorno de desarrollo integrado (IDE) de código abierto para programación en Python llamado Google Colab.

Procesamiento.

Para el procesamiento de los datos utilizamos redes neuronales convoluciones, modelo donde las neuronas tienen campos receptivos de manera similar a las neuronas de la corteza visual de un cerebro biológico, como también trabajamos con las librerías que nos ofrece Python y Tensorflow :

```
import tensorflow as tf
import numpy as np
import keras
import cv2
from sklearn import metrics
from keras.models import Sequential
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Dropout
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Input, Dense, Flatten
from keras.models import Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
import matplotlib.pyplot as plt
import scikitplot as skplt
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Input, Dense, Flatten
from keras.models import Model
```

Iniciamos configurando semillas aleatorias en TensorFlow y NumPy, asignando el valor 1 a la semilla. Luego, se definen los directorios de los conjuntos de datos de entrenamiento, validación y prueba. Las imágenes de entrada se establecen en un tamaño de 256x256 píxeles. A continuación, se crean generadores de datos para el conjunto de entrenamiento, validación y prueba utilizando la clase ImageDataGenerator de Keras. El generador de datos de entrenamiento aplica una rescalación de 1/255, un rango de corte de 0.2, un rango de zoom de 0.2 y una inversión horizontal. El generador de datos de validación realiza únicamente una rescalación de 1/255. El generador de datos de prueba también realiza una rescalación de 1/255, pero no mezcla los datos. Los generadores de datos de entrenamiento, validación y prueba cargan los datos de los respectivos directorios, redimensionan las imágenes al tamaño especificado, agrupan los

datos en lotes de 32 y codifican las etiquetas en formato categórico. En resumen, este código establece semillas aleatorias, define los directorios de datos y crea generadores

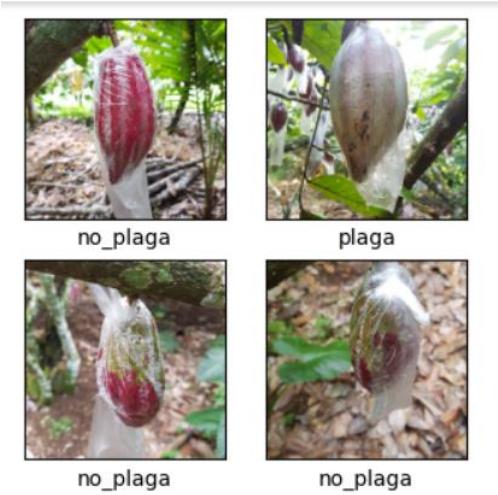


Fig2. Previsualización del Algoritmo de entrenamiento para la identificación de frutos con plaga y sin plaga.

de datos con transformaciones y aumentos para su uso en modelos de aprendizaje automático.

Modelo de entrenamiento val y test

Entrenamiento con Modelo CacaoCustom (Modelo Personalizado)

Construimos un modelo de red neuronal convolucional (CNN) secuencial utilizando la API Sequential de TensorFlow. El modelo se compone de varias capas convolucionales Conv2D seguidas de capas de agrupación MaxPooling2D, que ayudan a extraer y resumir características relevantes de las imágenes de entrada. Las capas convolucionales aplican operaciones de convolución a los mapas de características anteriores utilizando filtros de tamaño 3x3, mientras que las capas de agrupación reducen la dimensionalidad mediante submuestreo. Después de las capas de convolución y agrupación, se agrega una capa de aplanamiento Flatten para convertir los mapas de características en un vector unidimensional. A continuación, se agregan capas densas Dense que realizan operaciones de multiplicación matricial y aplican una función de activación ReLU para generar representaciones de mayor nivel. La última capa densa utiliza la función de activación Softmax para obtener una distribución de probabilidad sobre las clases de salida.

El modelo se compila utilizando el optimizador Adam, que se encarga de ajustar los pesos de la red durante el entrenamiento, minimizando la función de pérdida binary_crossentropy, ya que se trata de un problema de clasificación binaria. Además, se especifica que se desea evaluar la precisión (accuracy) como métrica de evaluación durante el entrenamiento.

Se definen dos callbacks: ModelCheckpoint para guardar el modelo con la mejor precisión en el conjunto de validación y EarlyStopping para detener el entrenamiento si no se observa una mejora en la precisión durante un número determinado de épocas.

Luego, el modelo se entrena utilizando el conjunto de datos de entrenamiento train_dataset durante un número determinado de épocas. Durante el entrenamiento, se utiliza el conjunto de datos de validación validation_dataset para evaluar el rendimiento del modelo en cada época y realizar el seguimiento de la precisión. Además, se utilizan las callbacks definidas anteriormente para guardar el mejor modelo y detener el entrenamiento según las condiciones especificadas.

Después de entrenar el modelo, se evalúa su rendimiento en el conjunto de prueba test_dataset utilizando el método evaluate(), obteniendo la pérdida y la precisión del modelo en ese conjunto.

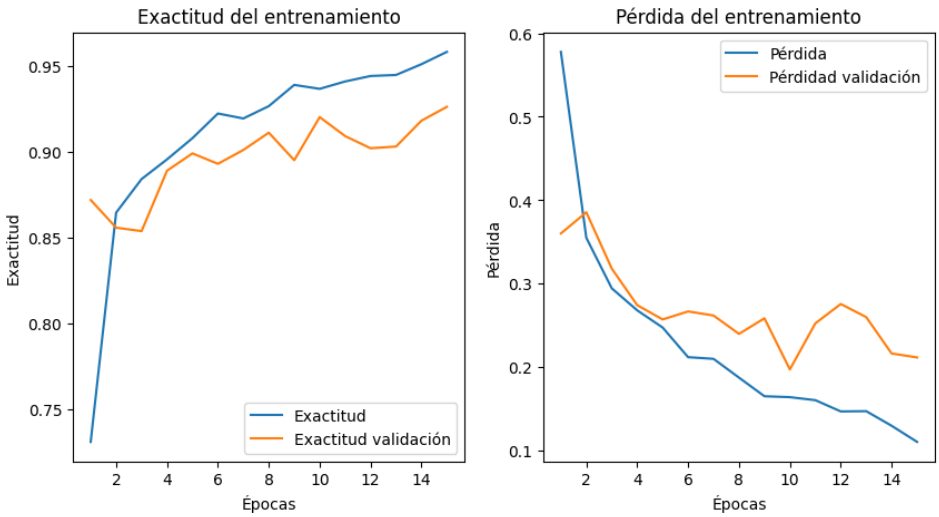


Fig 3. Train vs Val Loss y Accuracy.

Entrenamiento con InceptionV3 (Modelo Preentrenado)

Este código utiliza el modelo InceptionV3 preentrenado en la base de datos ImageNet como base para construir un nuevo modelo de clasificación de imágenes.

Primero, se instancia el modelo InceptionV3 utilizando la función InceptionV3 de Keras. Se especifica que se deben cargar los pesos preentrenados en ImageNet y se establece include_top en False para excluir las capas completamente conectadas superiores del modelo.

A continuación, se itera sobre todas las capas del modelo InceptionV3 y se establece el atributo trainable en False para congelar los pesos de todas las capas existentes. Esto impide que se actualicen durante el entrenamiento del nuevo modelo.

Luego, se toma la salida del modelo InceptionV3 y se pasa a través de una capa Flatten para convertir los mapas de características en un vector unidimensional.

Después de la capa Flatten, se agrega una capa densa (Dense) con 1024 unidades y la función de activación ReLU. Esta capa se encarga de realizar operaciones de multiplicación matricial y de introducir no linealidad en el modelo. Por último, se añade una capa densa de salida con 2 unidades (correspondientes a las clases de salida) y se utiliza la función de activación softmax para obtener una distribución de probabilidad sobre las clases.

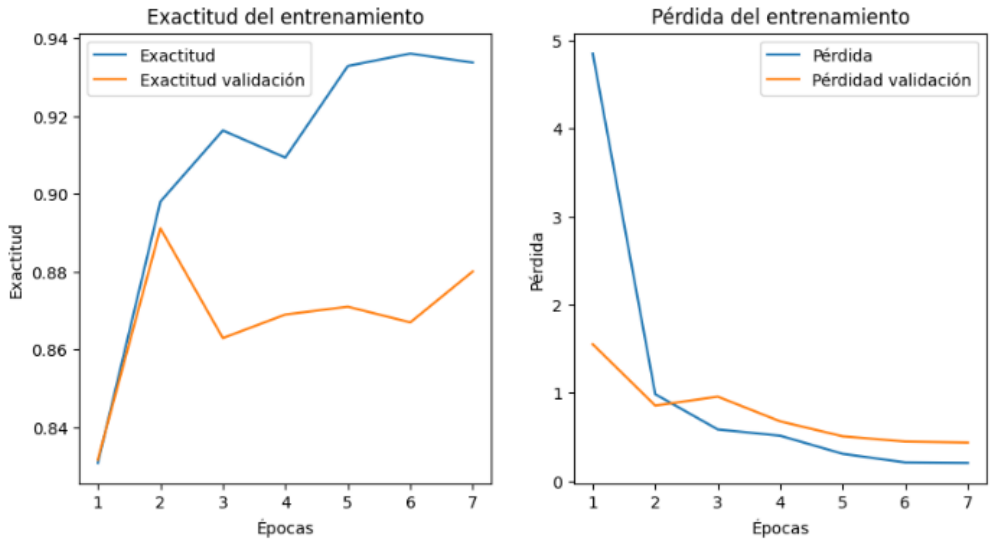


Fig 4. Train vs Val Loss y Accuracy.

Finalmente, se crea un nuevo modelo (Model) especificando las entradas como las entradas del modelo InceptionV3 y las salidas como las salidas obtenidas después de pasar a través de las capas personalizadas agregadas. Este modelo fusiona el modelo InceptionV3 preentrenado con las nuevas capas densas añadidas.

Compilamos el modelo utilizando el optimizador Adam con una tasa de aprendizaje predeterminada de 0.001, la función de pérdida de entropía cruzada binaria y la métrica de precisión. A continuación, se define un callback llamado ModelCheckpoint para guardar el mejor modelo durante el entrenamiento. Este callback monitorea la precisión en el conjunto de validación después de cada época y guarda el modelo con la mejor precisión en un archivo llamado "CacaoInceptionV3.h5". Además, se establece el modo de guardado en "max" para maximizar la precisión y se muestra un mensaje informativo durante el guardado del modelo.

Se define otro callback llamado EarlyStopping para detener el entrenamiento si no se observa una mejora en la precisión en el conjunto de validación durante 5 épocas consecutivas. Este callback monitorea la precisión en el conjunto de validación después de cada época y detiene el entrenamiento si no hay mejora en la precisión después de las 5 épocas.

Luego, se realiza el entrenamiento del modelo utilizando el conjunto de datos de entrenamiento y validación. Se establece un total de 30 épocas de entrenamiento. El número de pasos por época se calcula como el número total de muestras de entrenamiento dividido por el tamaño del lote de entrenamiento (32 en este caso). De manera similar, el número de pasos de validación se calcula como el número total de muestras de validación dividido por el tamaño del lote de validación (32 en este caso). Durante el entrenamiento, se utilizan las callbacks definidas anteriormente para guardar el mejor modelo y detener el entrenamiento si es necesario.

3. Resultados y discusión.

Evaluamos el desempeño de dos algoritmos de aprendizaje profundo en la predicción de plagas de Tehobroma Cacao. Estos algoritmos son: Modelo CacaoCustom y InceptionV3.

En la Tabla 1 mostramos las principales métricas que obtuvimos en la etapa de entrenamiento de los algoritmos. Podemos ver que el modelo InceptionV3 obtuvo los mejores resultados en todas las métricas evaluadas.

Tabla 1: Resumen estadístico de métricas.

Modelo	CacaoCustom	InceptionV3
Precisión	0.8200	0.9677
Exactitud	0.7156	0.7951
Especificidad	0.9554	0.9900
Sensibilidad	0.7156	0.7951
F1 Score	0.7156	0.7951
Kappa de Cohen	0.320045	0.5201
ROC	0.77	0.91

Obtuvimos el mejor valor respecto a sensibilidad con el modelo InceptionV3 de 0.7951. Esto indica la probabilidad de que el modelo clasifique correctamente a un fruto que está con plaga como plaga.

Obtuvimos un valor de 0.7951 en el F1 Score del modelo InceptionV3 y un valor cercano de 0.7156 en el modelo CacaoCustom.

Finalmente, en la Tabla 1 vemos que el modelo IceptionV3 obtuvo un valor de especificidad de 0.99, indicando que este algoritmo es adecuado para predecir la clase no plaga. Otra métrica que nos permite saber la probabilidad de que el fruto tenga plaga

dado que fue clasificado como tal, es la precisión. Este valor fue mayo en el modelo InceptionV3. El valor de precisión fue de 0.9677

En la Figura 6 mostramos la matriz de confusión de los modelos usados. La matriz de confusión del modelo InceptionV3 (b) muestra el valor más bajo de falsos negativos 65 (frutos con plaga que el modelo clasifico como frutos sin plaga) y el valor más alto de verdaderos positivos 60 (frutos con plaga que el modelo clasificó como plaga).

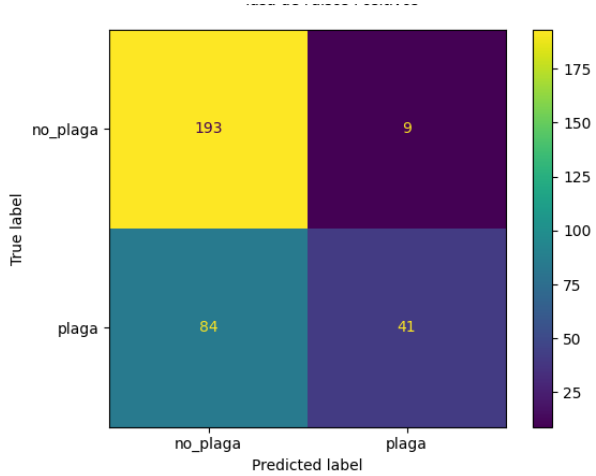


Fig 5. Matriz de Confusión del modelo CacaoCustom (Modelo personalizado).

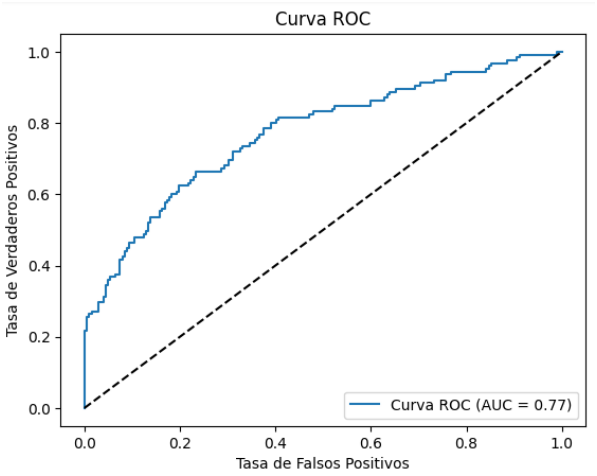


Fig 6. Área bajo la curva ROC. CacaoCustom (Modelo personalizado).

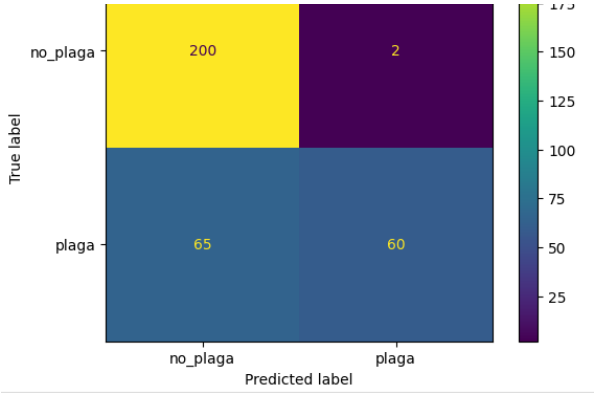


Fig7. Matriz de Confusión del modelo Incep (Modelo preentrenado).

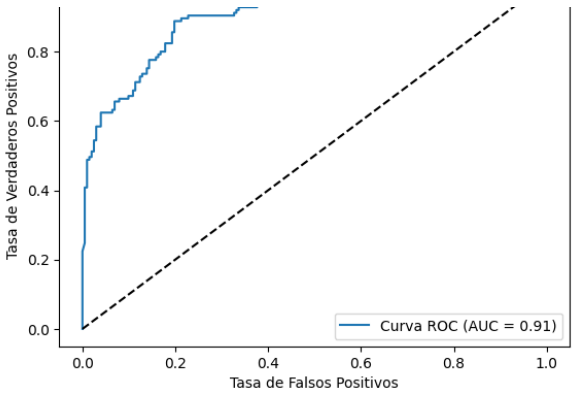


Fig 8. Área bajo la curva ROC de InceptionV3 ((Modelo preentrenado).

4. Conclusiones

Basándonos en los resultados mencionados, se puede concluir lo siguiente en el contexto agronómico, específicamente en la detección de la enfermedad de la Moniliasis en frutos:
En el estudio realizado, se evaluaron dos modelos de clasificación de imágenes, el modelo InceptionV3 y el modelo CacaoCustom, con el objetivo de detectar la presencia de la enfermedad de la Moniliasis en frutos en el campo agrícola. Los resultados obtenidos son altamente relevantes para la identificación precisa de frutos afectados por esta enfermedad, lo cual es de suma importancia en la gestión de la Moniliasis en cultivos específicos.

El modelo InceptionV3 demostró ser altamente sensible en la detección de frutos con Moniliasis, obteniendo un valor de sensibilidad de 0.7951. Esto indica que el modelo tiene una alta probabilidad de clasificar correctamente los frutos que están afectados por la enfermedad. Esta capacidad de detección precisa es fundamental en la toma de decisiones para implementar medidas de control específicas y oportunas, reduciendo así el impacto de la Moniliasis en los cultivos.

En cuanto al puntaje F1, el modelo InceptionV3 alcanzó un valor de 0.7951, mientras que el modelo Custom con un valor cercano de 0.7156. Este puntaje F1 refleja el equilibrio entre la precisión y la exhaustividad del modelo en la clasificación de frutos con Moniliasis. En este sentido, el modelo InceptionV3 se destaca por su capacidad de lograr un alto nivel de precisión y exhaustividad en la detección de frutos afectados por esta enfermedad.

Los resultados también revelan que el modelo InceptionV3 muestra una alta especificidad, con un valor de 0.99 en la tabla de resultados. Esto indica que el modelo es altamente preciso en la clasificación de frutos sanos, sin presencia de Moniliasis, lo cual es fundamental para evitar falsas alarmas y optimizar los recursos destinados a la gestión de la enfermedad.

En cuanto a la matriz de confusión, se observa que el modelo InceptionV3 tiene un bajo número de falsos negativos (65), lo que significa que clasificó correctamente una gran cantidad de frutos con Moniliasis. Además, el modelo InceptionV3 obtuvo el mayor número de verdaderos positivos (60), lo que indica su capacidad para identificar adecuadamente los frutos afectados por esta enfermedad.

En resumen, los resultados obtenidos en este estudio demuestran que el modelo InceptionV3 es altamente efectivo en la detección de la Moniliasis en frutos en el campo agronómico. Estos resultados proporcionan una herramienta valiosa para los profesionales agrónomos y productores, permitiéndoles tomar decisiones informadas y oportunas en la gestión de esta enfermedad, implementando medidas de control específicas en los cultivos afectados.

Referencias bibliográficas

1. **Ashqar, B. A. M., & Abu-Naser, S. S. (2018).** Image-Based Tomato Leaves Diseases Detection Using Deep Learning. International Journal of Academic Engineering Research, 2, 10–16.
2. **Ferentinos, K. P. (2018).** Deep learning models for plant disease detection and diagnosis. Computers and Electronics in Agriculture, 145, 311–318. https://doi.org/10.1016/J.COMPAG.2018.01.009
3. **Jha, K., Doshi, A., Patel, P., & Shah, M. (2019). A** comprehensive review on automation in agriculture using artificial intelligence. Artificial Intelligence in Agriculture, 2, 1–12. https://doi.org/10.1016/j.aiia.2019.05.004
4. **Khan, S., & Narvekar, M. (2020).** Disorder Detection in Tomato Plant Using Deep Learning. In Advanced Computing Technologies and Applications (pp. 187–197). Springer, Singapore. https://doi.org/10.1007/978-981-15-3242-9_19
5. **Krizhevsky, A., Sutskever, I., & Hinton, G. E.** (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84–90. https://doi.org/10.1145/3065386
6. **León, R., Díaz, M., & Rodríguez, L. (2020).** Gestión de un sistema de visión artificial para la detección de los daños causados por plagas en el cultivo de palto utilizando un drone. Cienc. Tecnol, 16(4), 135–141. https://doi.org/10.17268/rev.cyt.2020.04.14
7. **Leonardo, S., & Peralta, P. (2019).** Valoración económica del cambio de variedad de cacao en parcelas de productores de la provincia de Cotopaxi-Ecuador. REVISTA CIENTÍFICA ECOCIENCIA, 6(4), 1–20. https://doi.org/10.21855/ECOCIENCIA.64.199

8. **Papernot, N., & McDaniel, P. (2018).**
Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning. https://doi.org/https://doi.org/10.48550/arXiv.1803.04765
9. **Perú Velasco, E. C., Ordinola, M., & Devaux, A.** (2019). Una aproximación a la medición de pérdidas de alimento en la cadena de la papa en Ecuador y Perú. Revista Latinoamericana de La Papa, ISSN 1019-6609, ISSN-e 1853-4961, Vol. 23, No. 2, 2019, Págs. 46-65, 23(2), 46–65. https://dialnet.unirioja.es/servlet/articulo?codigo=7342638&info=resumen&idioma=ENG
10. **Piscoya Ferreñan, & Jesus Enrique. (2019).** Sistema de visión artificial para apoyar en la identificación de plagas y enfermedades del cultivo de sandía en el distrito de Ferreñafe. Universidad Católica Santo Toribio de Mogrovejo.
11. **Quintero-García, J. C., Murcia-Torrejano, V., Guzmán-Pacheco, K. Y., Saavedra-Mora, D., & Caviedes-Morales, J. D. (2019).**
DESAFÍOS TECNOLÓGICOS PARA EL MEJORAMIENTO DE LA TRAZABILIDAD DE CACAO (THEOBROMA CACAO L.): REVISIÓN LITERARIA. Revista Agropecuaria y Agroindustrial La Angostura, 6(1), 68–79. https://doi.org/10.23850/RAA.V6I1.3738
12. **Simonyan, K., & Zisserman, A. (2015).**
VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION.
13. **Singh, V., & Misra, A. K. (2017).**
Detection of plant leaf diseases using image segmentation and soft computing techniques. Information Processing in Agriculture, 4(1), 41–49. https://doi.org/10.1016/j.inpa.2016.10.005.
14. **Sullca Cecilia, Molina Carlos, Rodríguez Carlos, & Fernández Thais. (2019).**
Detección de enfermedades y plagas en las hojas de arándanos utilizando técnicas de visión artificial | Perspectiv@s. Perspectiv@s, 15, 32–39.
15. **Yandún Velasteguí, M. A. (2020).**
Detección de enfermedades en cultivos de Papa usando procesamiento de imágenes. Cumbres, 6(1), 43–52. https://doi.org/10.48190/cumbres.v6n1a4

16. **Krizhevsky, A., Sutskever, I. and Hinton, G.E.** (2017) Imagenet Classification with Deep Convolutional Neural Networks. Communications of the ACM, 60, 84-90.
https://doi.org/10.1145/3065386
17. Manrique, H. (2017). El largo camino hacia la economía lícita: Estado y estrategias de desarrollo alternativo en el «milagro de San Martín». Revista de Ciencia Política y Gobierno, 4(7), 161–189. https://doi.org/10.18800/rcpg.201701.007

Anexos

Código en python de las arquitecturas modelos CacaoCustom(modelo personalizado) e InceptionV3(modelo preentrenado).

Librerías utilizadas en los modelos.

```
import tensorflow as tf

import numpy as np

import keras

import cv2

from sklearn import metrics

from keras.models import Sequential

from keras.callbacks import ModelCheckpoint, EarlyStopping

from keras.layers import Dense, Conv2D, MaxPooling2D,
Flatten, Dropout

from keras.optimizers import Adam

from keras.preprocessing.image import ImageDataGenerator

from keras.layers import Input, Dense, Flatten

from keras.models import Model

from tensorflow.keras.applications.inception_v3 import
InceptionV3

import matplotlib.pyplot as plt

import scikitplot as skplt
```

Carga de dataset en las arquitecturas modelos.

```
from google.colab import drive

drive.mount('/gdrive')

%cd /gdrive/

%cd MyDrive/Cacao/
```

Modelo CacaoCustom(modelo personalizado)

```
train_data_dir = 'dataset/train'
validation_data_dir = 'dataset/val'
test_data_dir = 'dataset/test'
img_width, img_height = 256, 256
input_shape = (img_width, img_height, 3)
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
validation_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
train_dataset =
train_datagen.flow_from_directory(train_data_dir,
target_size=(img_width, img_height),

batch_size=32,

class_mode='categorical')
validation_dataset =
validation_datagen.flow_from_directory(validation_data_dir,
target_size=(img_width, img_height),

batch_size=32,
class_mode='categorical')
test_dataset =
test_datagen.flow_from_directory(test_data_dir,
target_size=(img_width, img_height),

batch_size=32, class_mode='categorical',

shuffle=False)

model = Sequential([
    # Capa de convolución con 32 filtros de 3x3 y función
de activación ReLU
```

```
    Conv2D(32, (3, 3), activation='relu',
input_shape=input_shape),
    # Capa de pooling para reducir la dimensionalidad
    MaxPooling2D((2, 2)),
    # Capa de convolución con 64 filtros de 3x3 y función
de activación ReLU
    Conv2D(64, (3, 3), activation='relu'),
    # Capa de pooling para reducir la dimensionalidad
    MaxPooling2D((2, 2)),
    # Capa de convolución con 128 filtros de 3x3 y función
de activación ReLU
    Conv2D(128, (3, 3), activation='relu'),
    # Capa de pooling para reducir la dimensionalidad
    MaxPooling2D((2, 2)),
    # Capa de convolución con 256 filtros de 3x3 y función
de activación ReLU
    Conv2D(256, (3, 3), activation='relu'),
    # Capa de pooling para reducir la dimensionalidad
    MaxPooling2D((2, 2)),
    # Capa de aplanamiento de la imagen
    Flatten(),
    # Capa dense con 512 unidades y función de activación
ReLU
    Dense(512, activation='relu'),
    # Capa de salida con 3 unidades y función de activación
Softmax
    Dense(2, activation='softmax')
])


model.summary()
# Compilamos el modelo
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

```
#Instanciamos las callbacks
checkpoint = ModelCheckpoint('CacaoCustom.h5',
                             monitor='val_accuracy',
                             save_best_only=True,
                             mode='max',
                             verbose=1)
earlyStop = EarlyStopping(monitor='val_accuracy',
patience=5)
# Entrenamos el modelo
history = model.fit(train_dataset, epochs=50,
validation_data=validation_dataset, callbacks=[checkpoint,
earlyStop])
# Evaluamos el modelo con el conjunto de prueba
test_loss, test_acc = model.evaluate(test_dataset)
# Imprimimos la precisión de la clasificación
print('Precisión de la clasificación:', test_acc)



import matplotlib.pyplot as plt
# Graficar la evolución de la precisión y pérdida durante
el entrenamiento
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))
ax1.plot(np.arange(1, 16), history.history['accuracy'],
label='Exactitud')
ax1.plot(np.arange(1, 16), history.history['val_accuracy'],
label='Exactitud validación')
ax1.set_title('Exactitud del entrenamiento')
ax1.set_xlabel('Épocas')
ax1.set_ylabel('Exactitud')
ax1.legend()
ax2.plot(np.arange(1,16), history.history['loss'],
label='Pérdida')
```

```
ax2.plot(np.arange(1,16), history.history['val_loss'],
label='Pérdidad validación')
ax2.set_title('Pérdida del entrenamiento')
ax2.set_xlabel('Épocas')
ax2.set_ylabel('Pérdida')
ax2.legend()
plt.show()

trained_model = keras.models.load_model('CacaoCustom.h5')
# prob = model.predict(test_dataset, verbose=0)
prob = trained_model.predict(test_dataset)
predicted_classes = np.argmax(prob, axis=1)



import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
labels = list(test_dataset.class_indices.keys())
for i in range(10):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(test_dataset[0][0][i])
    plt.xlabel(labels[predicted_classes[i]])
plt.show()


ref_classes = []
for index in range(len(test_dataset)):
  ref_classes.append(np.argmax(test_dataset[index][1],
axis=1))
ref_classes = np.hstack(ref_classes)
```

```
accuracy = metrics.accuracy_score(ref_classes,
predicted_classes)

print('Accuracy: %f' % accuracy)

f1 = metrics.f1_score(ref_classes, predicted_classes,
average='micro')

print('F1 score: %f' % f1)

precision = metrics.precision_score(ref_classes,
predicted_classes, average='micro')

print('Precision: %f' % precision)

recall = metrics.recall_score(ref_classes,
predicted_classes, average='micro')

print('Recall: %f' % recall)

cohen_kappa = metrics.cohen_kappa_score(ref_classes,
predicted_classes)

print('Cohen kappa: %f' % cohen_kappa)


auc_score = metrics.roc_auc_score(ref_classes, prob[:,1])

fpr, tpr, thresholds = metrics.roc_curve(ref_classes,
prob[:,1])

plt.plot(fpr, tpr, label='Curva ROC (AUC = %0.2f)' %
auc_score)

plt.plot([0, 1], [0, 1], 'k--')  # Línea de referencia:
clasificación aleatoria

plt.xlabel('Tasa de Falsos Positivos')

plt.ylabel('Tasa de Verdaderos Positivos')

plt.title('Curva ROC')

plt.legend(loc='lower right')


matrix = metrics.confusion_matrix(ref_classes,
predicted_classes)

matrix_disp = metrics.ConfusionMatrixDisplay(matrix,
display_labels=labels)

matrix_disp.plot()
```

```
plt.show()
```

Modelo CacaoInceptionV3 (modelo preentrenado)

```
seed = 1
tf.random.set_seed(seed)
np.random.seed(seed)
train_data_dir = 'dataset/train'
validation_data_dir = 'dataset/val'
test_data_dir = 'dataset/test'
img_width, img_height = 256, 256
input_shape = (img_width, img_height, 3)
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
validation_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
train_dataset =
train_datagen.flow_from_directory(train_data_dir,
target_size=(img_width, img_height),

batch_size=32,
class_mode='categorical')
validation_dataset =
validation_datagen.flow_from_directory(validation_data_dir,
target_size=(img_width, img_height),

batch_size=32, class_mode='categorical')
test_dataset =
test_datagen.flow_from_directory(test_data_dir,
target_size=(img_width, img_height),

batch_size=32, class_mode='categorical',

shuffle=False)
```

```
batch = train_dataset.next()
muestra = batch[0][:10]
labels = list(train_dataset.class_indices.keys())
etiquetas = np.argmax(batch[1], axis=1)
plt.figure(figsize=(14, 6))
for i in range(len(muestra)):
    plt.subplot(2, 5, i+1)
    # plt.axis('auto')
    plt.imshow(muestra[i])
    plt.xlabel(labels[etiquetas[i]])
plt.show()
```

```
iv3_model = InceptionV3(weights='imagenet',
include_top=False, input_shape=input_shape)
for layer in iv3_model.layers:
  layer.trainable = False
res = iv3_model.output
res = Flatten()(res)
res = Dense(1024, activation = 'relu')(res)
predictions = Dense(2, activation = 'softmax')(res)
model = Model(inputs = iv3_model.input, outputs =
predictions)

model.compile(optimizer = 'adam',
loss='binary_crossentropy', metrics = ['accuracy'])
checkpoint = ModelCheckpoint('CacaoInceptionV3.h5',
                             monitor='val_accuracy',
```

```
                                save_best_only=True,
                                mode='max',
                                verbose=1)
earlyStop = EarlyStopping(monitor='val_accuracy',
patience=5)


history = model.fit(train_dataset, steps_per_epoch =
train_dataset.n//train_dataset.batch_size,
                    epochs = 30, validation_data =
validation_dataset,
                    validation_steps =
validation_dataset.n//validation_dataset.batch_size,
                    callbacks=[checkpoint, earlyStop])


score = model.evaluate(test_dataset, steps =
test_dataset.n//test_dataset.batch_size)
print('Pérdida: ', score[0])
print('Precisión: ', score[1])


import matplotlib.pyplot as plt
# Graficar la evolución de la precisión y pérdida durante
el entrenamiento
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))
ax1.plot(np.arange(1, 8), history.history['accuracy'],
label='Exactitud')
ax1.plot(np.arange(1, 8), history.history['val_accuracy'],
label='Exactitud validación')
ax1.set_title('Exactitud del entrenamiento')
ax1.set_xlabel('Épocas')
ax1.set_ylabel('Exactitud')
ax1.legend()
ax2.plot(np.arange(1, 8), history.history['loss'],
label='Pérdida')
```

```
ax2.plot(np.arange(1, 8), history.history['val_loss'],
label='Pérdidad validación')

ax2.set_title('Pérdida del entrenamiento')

ax2.set_xlabel('Épocas')

ax2.set_ylabel('Pérdida')

ax2.legend()

plt.show()


labels = list(test_dataset.class_indices.keys())

from sklearn import metrics

ref_classes = []

for index in range(len(test_dataset)):

  ref_classes.extend(np.argmax(test_dataset[index][1],
axis=1))

accuracy = metrics.accuracy_score(ref_classes,
predicted_classes)

print('Exactitud: %f' % accuracy)

f1 = metrics.f1_score(ref_classes, predicted_classes,
average='micro')

print('Coeficiente F1: %f' % f1)

precision = metrics.precision_score(ref_classes,
predicted_classes, average='micro')

print('Precisión: %f' % precision)

recall = metrics.recall_score(ref_classes,
predicted_classes, average='micro')

print('Recall: %f' % recall)

cohen_kappa = metrics.cohen_kappa_score(ref_classes,
predicted_classes)

print('Kappa de Cohen: %f' % cohen_kappa)


auc_score = metrics.roc_auc_score(ref_classes, prob[:,1])

print('ROC: %f' % auc_score)
```

```
fpr, tpr, thresholds = metrics.roc_curve(ref_classes,
prob[:,1])

plt.plot(fpr, tpr, label='Curva ROC (AUC = %0.2f)' %
auc_score)

plt.plot([0, 1], [0, 1], 'k--')  # Línea de referencia:
clasificación aleatoria

plt.xlabel('Tasa de Falsos Positivos')

plt.ylabel('Tasa de Verdaderos Positivos')

plt.title('Curva ROC')

plt.legend(loc='lower right')

matrix = metrics.confusion_matrix(ref_classes,
predicted_classes)

matrix_disp = metrics.ConfusionMatrixDisplay(matrix,
display_labels=labels)

matrix_disp.plot()

plt.show()
```

Anexos Documentación

"AÑO DEL BICENTENARIO DEL PERÚ: 200 AÑOS DE INDEPENDENCIA"

RESOLUCIÓN N° 1408/A-2021/UPeU-FIA-CF-T

Lima, Ñaña 21 de diciembre de 2021

VISTO:

El expediente de **Oswaldo Chuquipoma Fermín**, identificado(a) con Código Universitario N° 201713197 y **Jared Elim Arapa Mejia**, identificado(a) con Código Universitario N° 201710177, de la Escuela Profesional de Ingeniería de Sistemas de la Facultad de Ingeniería y Arquitectura de la Universidad Peruana Unión;

CONSIDERANDO

Que la Universidad Peruana Unión tiene autonomía académica, administrativa y normativa, dentro del ámbito establecido por la Ley Universitaria N° 30220 y el Estatuto de la Universidad;

Que la Facultad de Ingeniería y Arquitectura de la Universidad Peruana Unión, mediante sus reglamentos académicos y administrativos, ha establecido las formas y procedimientos para la aprobación e inscripción del perfil de proyecto de tesis en formato artículo y la designación o nombramiento del asesor para la obtención del título profesional;

Que **Oswaldo Chuquipoma Fermín** y **Jared Elim Arapa Mejia**, han solicitado: la inscripción del perfil de proyecto de tesis titulado "Sistema experto basado en visión artificial para el diagnóstico de plagas de cacao en la región San Martín" y la designación del Asesor, encargado de orientar y asesorar la ejecución del perfil de proyecto de tesis en formato artículo;

Estando a lo acordado en la sesión del Consejo de la Facultad de Ingeniería y Arquitectura de la Universidad Peruana Unión, celebrada el 21 de diciembre de 2021, y en aplicación del Estatuto y el Reglamento General de Investigación de la Universidad;

SE RESUELVE:

Aprobar el perfil de proyecto de tesis en formato artículo titulado **"Sistema experto basado en visión artificial para el diagnóstico de plagas de cacao en la región San Martín"** y disponer su inscripción en el registro correspondiente, designar al (a la) **Dr. Miguel Angel Valles Coral** como ASESOR para que oriente y asesore la ejecución del perfil de proyecto de tesis en formato artículo el cual fue dictaminado por: **Mg. Danny Lévano Rodriguez** y **Mg. Marco Antonio Ruiz Grandez**, otorgándoles un plazo máximo de doce (12) meses para la ejecución.

Regístrese, comuníquese y archívese.

Dra. María Vallejos Atalaya de Cornejo
DECANA

Dra. Erika Inés Acuña Salinas
SECRETARIA ACADÉMICA

cc:
-Interesado
Asesor
Dirección General de Investigación
Archivo

Evidencias de Sumisión: