Transfer learning with convolutional neural networks for advanced image classification and detection of Musa × paradisiaca leaf disease.

Universidad Tecnologica La Salle, Member number #95659087 A. Trujillo, Estudiante de ingeniería cibernética electrónica, IEEE, arnoldo_trujillo@ieee.org, Member number #95666642, D. Centeno, Estudiante de ingeniería cibernética electrónica, IEEE, dcentenogonza@gmail.com.

***Abstract* - In this research was created a Convolution Neural Networks called "MusaNet" to detect banana issues using Transfer learning, the creation of a little Dataset with 478 images and the transfer learning method used to train the network, as well as the accuracy percentages when predicting. The network "MusaNet" had an accuracy of 91.66% giving a quick response when querying one of the 3 defined categories such as good leaves or in good condition, leaf spot and Sigatoka. The transfer learning through fine-tuning a pre-trained neural network with an extremely large dataset, such as MobileNet V2, can significantly accelerate training while the accuracy is frequently bottlenecked by the limited dataset size of the new target task, were used to solved the new tasks, using Python and TensorFlow.**

Keywords – Deep Learning, Convolutional Neural Networks, Transfer learning, MobileNet, Python, TensorFlow, MusaNet.

I. Introducción

La Musa paradisiaca o también conocido como plátano, del cual existen en el país dos especies de esta familia para su comercialización los cuales por su nombre científico es Musa cavendishii (plátanos comestibles cuando están crudos) y Musa Paradisiaca (plátanos machos o para cocer), fruto que cuenta con distintos nombres alrededor del mundo así como también diferente tamaño, color y firmeza el cual es nativo de la región indomalaya que fueron domesticadas por primera vez en Papúa Nueva Guinea y llegaron a América en 1516 [1].

Mediante las nuevas tecnologías que han surgido a lo largo de los años, se encuentra la inteligencia artificial y sus diferentes aplicaciones, dentro de las cuales encontramos las redes neuronales convolucionales, estas se han caracterizado por tener campos receptivos muy similares a las neuronas encontradas en la corteza visual primaria de nuestro cerebro [2], esto abre el camino a infinidad de aplicaciones.

La visión artificial es una rama de la inteligencia artificial cuyo propósito es diseñar sistemas informáticos capaces de "entender" los elementos y características de una escena o imagen del mundo real [3]. En el documento se presentará todo el proceso de creación del Dataset paso a paso hasta el funcionamiento completo de la red neuronal convolucional para la clasificación de las imágenes dependiendo la enfermedad de la hoja del plátano. Se entrenó un modelo de CNN con ayuda de una red pre-entrenada para la detección de dos enfermedades en la hoja del plátano y la detección del estado saludable de la hoja, los nombres de las enfermedades en plátanos que es capaz de detectar son: La "mancha foliar de Cordana" y la "sigatoka del plátano". Esto con el fin de realizar una detección de las mismas de una manera eficiente, todo esto basado en redes neuronales convolucionales utilizando Python para el entrenamiento y ajuste de la red y TensorFlow para la creación del Dataset.

II. Objetivos

Objetivo general

- Desarrollar una red neuronal convolucional para la detección de enfermedades en los plátanos utilizando transferencia de aprendizaje.

Objetivos específicos.

- Creación y clasificación del dataset con aumento de datos.
- Definir método de transferencia de aprendizaje utilizado.
- Demostrar la eficiencia de la red "MusaNet".

III. Metodología

El estudio se realizó en una pequeña finca en la localidad de León, Nicaragua, con el fin de detectar

enfermedades en las hojas de los plátanos. En el presente documento se exponen las diferentes fases desde la creación del dataset, aumento de datos, transferencia de aprendizaje, entrenamiento, hasta llegar al modelo final "MusaNet". La investigación tiene un enfoque observacional cuantitativo ya que mediante la observación del entrenamiento se obtuvo un porcentaje de precisión satisfactorio en las predicciones de las enfermedades.

IV. Creación y clasificación del dataset con aumento de datos.

A. Creación y clasificación del data set.

Se combinaron imágenes de la web y fotografías tomadas "in situ" en sus diferentes estados para ser utilizado en el entrenamiento del modelo. El dataset tiene que ser lo suficientemente diverso por lo cual se realizó la captura de imágenes de cientos de plátanos desde varios ángulos, posiciones, iluminaciones y con un mismo color de fondo, para un mejor resultado al entrenar el modelo CNN , se reemplazó el fondo por un color blanco con el fin de estandarizar las imágenes y de esta manera optimizar la eficiencia del clasificador. El Dataset fue conformado por 478 imágenes de la hoja del plátano, en las imágenes encontramos 3 diferentes tipos de clasificación las cuales son: Hoja buena, Sigatoka y mancha foliar. Con ayuda de la librería "datagen.flow_from_directory" [4] dividimos el Dataset en dos, con el fin de dejar un 80% del dataset para entrenamiento y el 20% restante para los test de predicciones. Cuando se realizó la división del Dataset con ayuda de la librería, automáticamente nos asigna las etiquetas de cada clasificación, las cuales son: "0 = Hoja buena", "1 = Mancha foliar", "2 = Sigatoka". De igual forma se redimensiona el tamaño de las imágenes a un tamaño de 224x224 pixeles ya que el input de la red solicita las imágenes con esas dimensiones.

```
#Generator of training set with a size of 224x224
train_generator=datagen.flow_from_directory('/content/drive/MyDrive/Paper_V2/DataSet2/Train_set',
                                            target_size=(224,224),
                                            color_mode='rgb',
                                            batch_size=32,
                                            class_mode='categorical',
                                            shuffle=True)
```

Fig. 1: Creación del set de entrenamiento - (Autoría propia).

```
#Generator of training set with a size of 224x224
train_generator=datagen.flow_from_directory('/content/drive/MyDrive/Paper_V2/DataSet2/Train_set',
                                            target_size=(224,224),
                                            color_mode='rgb',
                                            batch_size=32,
                                            class_mode='categorical',
                                            shuffle=True)
```

Fig. 2: Creación de set de pruebas - (Autoría propia).

B. Aumento de datos

Debido a las pocas imágenes que contiene el Dataset se creó un generador de aumento de datos con la librería "ImageDataGenerator" [4] de TenserFlow, el generador tomo cada imagen del Dataset y les aplicó diversas transformaciones, como un escalado de 1/255, un rango de rotación de un 30°, una gama de desplazamiento en anchura de un 25%, una gama de desplazamiento de altura de un 25%, rango de corte o cizallamiento de 15° y un rango de zoom entre 0.5 hasta 1.5. Esto nos brinda un Dataset más robusto y variado y de esta manera el modelo puede clasificar las imágenes a cómo las neuronas de la corteza visual primaria lo hacen.

```
#Data augmentation, create a dataset generador
datagen = ImageDataGenerator(rescale=1. / 255, rotation_range = 30,
    width_shift_range = 0.25, height_shift_range = 0.25,
    shear_range = 15, zoom_range = [0.5, 1.5]
)
```

Fig. 3: Generador para el aumento de datos - (Autoría propia).

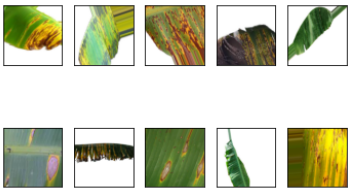

Fig. 4: Visualización de 10 imágenes del Dataset con las transformaciones aplicadas por el generador de aumento de datos - (Autoría propia).

V. Método de transferencia de aprendizaje utilizado.

A. Modelo de la red

MobileNet y MobileNet V2 son redes neuronales convolucionales creadas por Google, está conformada por una gran cantidad de capas y millones de parámetros que fueron entrenados por un set de datos gigantesco llamado ImageNet, este set de datos tiene más de diez millones de imágenes clasificadas, por lo tanto MobileNet es una red convolucional bastante robusta entrenada para detectar con gran precisión el mapa de características que le pase (Líneas, curvas, círculos, etc), es una red muy eficiente para trabajar con pocos recursos. Por lo tanto se tomó esta red como base para adaptarla a nuestro proyecto. La red fue diseñada para recibir imágenes de 224x224 pixeles a color con sus 3 canales, la salida está constituida por una capa de clasificación softmax con 1001 neuronas para clasificar imágenes en una de esas categorías [5].

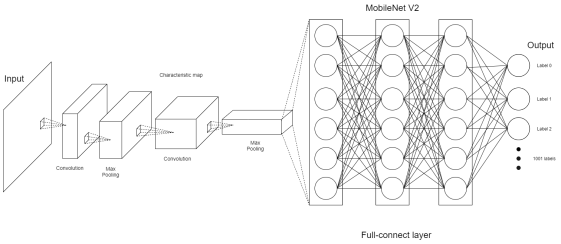

Fig. 5: Modelo de la red MobileNet V2 - Autoría propia - (Autoría propia).

B. Aplicación de la transferencia de aprendizaje y entrenamiento

Se descargo el modelo que se utilizó como base para nuestra red "MobileNet V2", pero sin su última capa ya que en la última capa es donde se adaptó la red para la clasificación de las imágenes que deseamos.

```
#Download Mobilenet V2 model
mobilenet_v2 = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"
```

Fig. 6: Importación del modelo Mobile Net V2 sin su última capa.

El objetivo de esto es congelar todo el modelo MobileNet para no perder todo el conocimiento previo, se le agregó una nueva capa densa con softmax que clasifica en las 3 categorías que se deseó [6]. De esta manera solo fue necesario entrenar la última capa que agregamos.

```
#Frozen the download model
musa_net_layers.trainable = False
```

Fig. 7: Congelamiento del modelo MobileNet V2 - (Autoría propia).

```
#Create a sequential model with 3 out dense neurons with softmax activation
musa_net = tf.keras.Sequential([
  musa_net_layers,
  tf.keras.layers.Dropout(0.3),
  tf.keras.layers.Dense(3,activation='softmax')
])
```

Fig. 8: Creación de la última capa con softmax de un modelo secuencial - (Autoría propia).

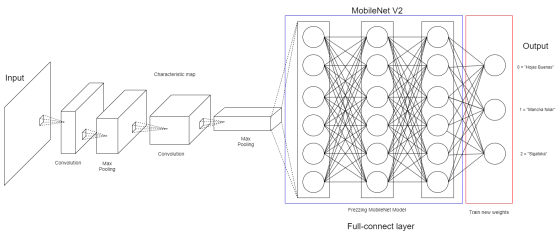

Fig. 9: Nueva estructura del modelo MobileNet V2 con su nueva salida - (Autoría propia).

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer (KerasLayer)    (None, 1280)              2257984

 dropout (Dropout)           (None, 1280)              0

 dense (Dense)               (None, 3)                 3843

=================================================================
Total params: 2,261,827
Trainable params: 3,843
Non-trainable params: 2,257,984
_________________________________________________________________
```

Fig. 10: Resumen del nuevo modelo - (Autoría propia).

Se creó un modelo secuencial con 3 neuronas densas de salida y un total de 2,261,827 millones de parámetros[7].

VI. Entrenamiento de la red "MusaNet".

Una vez se creó el nuevo modelo se compiló, con el optimizador "SGD" con una taza de aprendizaje del 0.01 este se encarga de mejorar el rendimiento del entrenamiento utilizando el método del descenso del gradiente [8], se designó una variable de pérdidas con clase "Categorical Crossentropy" el cual se encarga de calcular el porcentaje de pérdidas entre las etiquetas y las predicciones [9], también se designó una variable con la clase "accuracy" para calcular el porcentaje de aciertos o precisión entre las etiquetas y las predicciones[10].

```
#Train model
# SGD optimizer
# loss function will be categorical cross entropy
# evaluation metric will be accuracy
musa_net.compile(optimizer=opt,loss='categorical_crossentropy',metrics=['accuracy'])
```

Fig. 11: Compilación del modelo - (Autoría propia).

Teniendo el modelo compilado se entrenó durante 10 épocas con la función "fit" de la librería Keras [11].

```
History = musa_net.fit(train_generator, epochs=10, validation_data=test_generator)

Epoch 1/10
12/12 [==============================] - 552s 46s/step - loss: 1.2065 - accuracy: 0.4607 - val_loss: 0.8994 - val_accuracy: 0.5521
Epoch 2/10
12/12 [==============================] - 159s 13s/step - loss: 0.9768 - accuracy: 0.5628 - val_loss: 0.8491 - val_accuracy: 0.6042
Epoch 3/10
12/12 [==============================] - 155s 13s/step - loss: 0.7958 - accuracy: 0.6518 - val_loss: 0.6792 - val_accuracy: 0.7188
Epoch 4/10
12/12 [==============================] - 153s 13s/step - loss: 0.7726 - accuracy: 0.6675 - val_loss: 0.6770 - val_accuracy: 0.7500
Epoch 5/10
12/12 [==============================] - 153s 13s/step - loss: 0.7447 - accuracy: 0.6885 - val_loss: 0.6985 - val_accuracy: 0.6979
Epoch 6/10
12/12 [==============================] - 153s 13s/step - loss: 0.6525 - accuracy: 0.7199 - val_loss: 0.8013 - val_accuracy: 0.6562
Epoch 7/10
12/12 [==============================] - 152s 13s/step - loss: 0.6571 - accuracy: 0.7592 - val_loss: 0.7145 - val_accuracy: 0.6458
Epoch 8/10
12/12 [==============================] - 153s 13s/step - loss: 0.6166 - accuracy: 0.7723 - val_loss: 0.7195 - val_accuracy: 0.6771
Epoch 9/10
12/12 [==============================] - 152s 13s/step - loss: 0.5664 - accuracy: 0.7435 - val_loss: 0.6354 - val_accuracy: 0.7292
Epoch 10/10
12/12 [==============================] - 152s 13s/step - loss: 0.6000 - accuracy: 0.7539 - val_loss: 0.7535 - val_accuracy: 0.6875
```

Fig. 12: Entrenamiento del modelo durante 10 épocas - (Autoría propia).

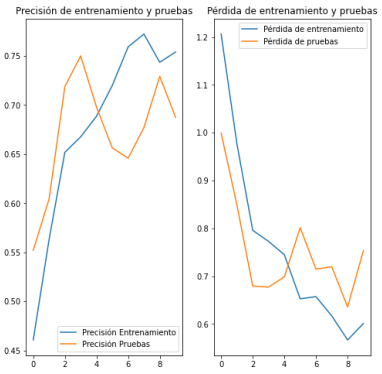

Fig. 13: "Gráfica de precisión de entrenamiento y pruebas" y "Gráfica de pérdida de entrenamiento y pruebas" - (Autoría propia).

VII. Resultados

Se observa en la figura 13 el resultado de las 10 épocas de entrenamiento las cuales resultaron satisfactorias. El modelo nos arrojó un valor de precisión del 67.20% de precisión.

Se realizarón una serie de predicciones pasandole al modelo “MusaNet” ya entrenado con imágenes de las enfermedades, fueron un total de 60 pruebas realizadas, 20 para la enfermedad “Mancha foliar de cordana” de las cuales 18 de las pruebas fueron acertadas obteniendo una precisión del 90%, para la enfermedad de “Sigatoka” de igual manera se realizaron 20 pruebas de las cuales 18 de las pruebas fueron acertadas obteniendo un 90% de precisión y finalmente para el estado sano de la hoja “Hoja buena” se realizaron 20 pruebas de las cuales 19 de las pruebas fueron acertadas obteniendo un 95% de precisión.

Por lo tanto, sacando la media de las predicciones acertadas se obtuvo una precisión del 91.66% [12].

VIII. Conclusiones.

En conclusión se creó con éxito la red “MusaNet” utilizando la transferencia de aprendizaje. Entrenando la red con un pequeño DataSet de imágenes de las enfermedades, el modelo se entrenó eficientemente y se obtuvieron resultados satisfactorios de la red.

Sin embargo la red funciona eficazmente sólo cuando las imágenes que recibe la red son de la hoja con la enfermedad en un primer plano, al recibir una imagen amplia en la cual se muestra más partes del cultivo nos da resultados incoherentes y poco precisos.

Referencias

[1]. V. Anders. "BANANA". Etimologías de Chile - Diccionario que explica el origen de las palabras. [Online] http://etimologias.dechile.net/?banana (accedido el 5 de julio de 2022).
[2]. F. A. Moreno Urrea, "Estudio de detección de emociones a través de reconocimiento de características faciales con técnicas AI", Universidad Politécnica de Cartagena, Cartagena, 2021. Accedido el 12 de julio de 2022. [En línea]. Disponible: https://repositorio.upct.es/bitstream/handle/10317/9371/tfg-mor-est.pdf?sequence=1&isAllowed=y
[3]. F. J. Nuñez, "Diseño de un sistema de reconocimiento automático de matrículas de vehículos mediante una red neuronal convolucional", 2016, jun. 01
[4]. Google Brain Team. "Tf.keras.preprocessing.image.ImageDataGenerator | tensorflow core v2.9.1". TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator (accedido el 12 de julio de 2022).
[5]. M. Sandler and A. Howard, "MobileNetV2: The Next Generation of On-Device Computer Vision Networks", Google Research, (2018, April 3).
[6]. Google Brain Team. "hub.KerasLayer tensorflow core v2.9.1". TensorFlow. https://www.tensorflow.org/hub/api_docs/python/hub/KerasLayer (accedido el 12 de julio de 2022).
[7]. Google Brain Team. "Keras documentation: The Model class | tensorflow core v2.9.1". TensorFlow. https://keras.io/api/models/model/ (accedido el 12 de julio de 2022).
[8]. Google Brain Team. "SGD | tensorflow core v2.9.1". TensorFlow. https://keras.io/api/optimizers/sgd/ (accedido el 12 de julio de 2022).
[9]. Google Brain Team. "tf.keras.losses.CategoricalCrossentropy | tensorflow core v2.9.1". TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy (accedido el 12 de julio de 2022).
[10]. Google Brain Team. "Accuracy metrics | tensorflow core v2.9.1". TensorFlow. https://keras.io/api/metrics/accuracy_metrics/ (accedido el 12 de julio de 2022).
[11]. Google Brain Team. "Model training APIs | tensorflow core v2.9.1". TensorFlow. https://keras.io/api/models/model_training_apis/ (accedido el 12 de julio de 2022).
[12]. D. Centeno and A. Trujillo (2022, month, day), Transfer learning to create “MusaNet” CNN model. [Online]. Available: https://github.com/Hunnter7/Transfer-learning-to-create-MusaNet-CNN

Daniel J. Centeno González nació en León, Nicaragua el 18 de abril de 2001.
Se recibió de bachiller en ciencias y letras del instituto La Asunción en León en el 2018 y actualmente cursando su cuarto año de la carrera de ingeniería cibernética electrónica en la Universidad Tecnológica La Salle. Actual miembro de la rama IEEE ULSA 2022 y Tesorero de la misma.

Arnoldo J. Trujillo Robelo nació en León, Nicaragua el 7 abril de 2002.
Se recibió de bachiller en ciencias y letras del instituto La Asunción en León en el 2018 y actualmente cursando su cuarto año de la carrera de ingeniería cibernética electrónica en la Universidad Tecnológica La Salle. Actual miembro de la rama IEEE ULSA 2022 y presidente de la misma.