Aplicación del modelo mobilenetv3 para la reducción de residuos orgánicos

Mullisaca Condori Carlos Augusto

aEP. Ingeniería de Sistemas, Facultad de Ingeniería y Arquitectura, Universidad Peruana Unión

Resumen

El objetivo de esta investigación es desarrollar y entrenar una red neuronal, específicamente MobileNetV3, para identificar en qué fase del proceso se encuentra el compost. Se recopiló un conjunto de datos de 1311 imágenes, divididas equitativamente entre las fases de mesofilia, termofilia y maduración. Los resultados indicaron que la red neuronal puede predecir eficazmente la fase del compost, facilitando así la fabricación de compostaje en los hogares y promoviendo una gestión más eficiente de los residuos orgánicos. Las conclusiones sugieren que la implementación de redes neuronales puede mejorar significativamente la calidad y eficiencia del compostaje.

*Palabras clave:* Compostaje; Redes Neuronales; MobileNetV3; Gestión de Residuos; Inteligencia Artificial

# Introducción

La producción de residuos sólidos es un desafío global, exacerbado por la falta de conciencia y educación sobre su adecuada gestión. Muchos desconocen los beneficios del compostaje y tratan los residuos orgánicos como basura, resultando en un desperdicio de materiales valiosos que podrían enriquecer el suelo y reducir la dependencia de fertilizantes químicos. Además, la falta de infraestructura y políticas efectivas para el compostaje limita su adopción. Este proyecto propone utilizar inteligencia artificial para optimizar la producción de compostaje, entrenando un modelo de clasificación de imágenes MobileNetV3 para desarrollar un prototipo inicial. Los objetivos incluyen desarrollar una red neuronal para identificar las fases del compost, facilitar su fabricación en hogares, reducir residuos y promover la educación y conciencia ambiental.

# Materiales y Métodos

## Participantes

El método mantiene cohesión y coherencia.

## Instrumentos

Computadora portátil: para realizar todo el procesamiento informático (en caso de usar jupyter notebook, se recomienda que tenga tarjeta gráfica para reducir el tiempo de entrenamiento)

Google colab: Utilizado para el entrenamiento, refinamiento y ajuste del modelo de red neuronal

Librerías de software: TensorFlow, pandas y Keras para la construcción, entrenamiento y evaluación de la red neuronal

MobileNetV3: modelo de red neuronal

DataSet1:

Cámara digital de teléfono móvil: Las imágenes fueron clasificados de forma manual. Los principales instrumentos y herramientas utilizados fueron:

DataSet2:

Extensión de Chrome download all images: para descargar grandes cantidades de imágenes de internet para su posterior selección manual

## Análisis de datos

El análisis de datos se centró en evaluar la precisión y eficacia del modelo MobileNetV3 para clasificar las fases del compostaje. El proceso de análisis incluyó las siguientes etapas:

Preparación de los Datos Dataset:

Se comprimio las imágenes en un archivo zip para pasarlo al colab y que se pueda

Preparación de los Datos Dataset2:

instalamos la extensión dowload all images para nuestro set de datos

Realizamos búsquedas sobre el compostage de forma masiva y los descargamos con la extencion instalada

Clasificamos las imágenes de forma manual en tres carpetas diferentes que se llamaran mesofilia termofilia y maduración, comprimimos un archivo zip para cada una.

ejecutamos una interface de Google colab; creamos una carpeta para cada archivo zip con el mismo nombre, movemos nuestros archivos zip a cada carpeta con sus respectivos nombres y los descomprimimos

creamos un carpeta llamada datset

##Copiar imagenes que subimos a carpetas del dataset

#Limitar para que todos tengan la misma cantidad de imagenes

#maximo 419 (el num. menor de imagenes que subi)

import shutil

carpeta\_fuente = '/content/mesofilia'

carpeta\_destino = '/content/dataset/mesofilia'

imagenes = os.listdir(carpeta\_fuente)

for i, nombreimg in enumerate(imagenes):

  if i < 419:

    #Copia de la carpeta fuente a la destino

    shutil.copy(carpeta\_fuente + '/' + nombreimg, carpeta\_destino + '/' + nombreimg)

#Mostrar cuantas imagenes tengo de cada categoria en el dataset para asaegurarnos que sea igual en todos

!ls /content/dataset/cuchara | wc -l

!ls /content/dataset/cuchillo | wc -l

!ls /content/dataset/tenedor | wc -l

#Copiar imagenes que subimos a carpetas del dataset

#Limitar para que todos tengan la misma cantidad de imagenes

#maximo 419 (el num. menor de imagenes que subi)

import shutil

carpeta\_fuente = '/content/mesofilia'

carpeta\_destino = '/content/dataset/mesofilia'

imagenes = os.listdir(carpeta\_fuente)

for i, nombreimg in enumerate(imagenes):

  if i < 419:

    #Copia de la carpeta fuente a la destino

    shutil.copy(carpeta\_fuente + '/' + nombreimg, carpeta\_destino + '/' + nombreimg)

#Aumento de datos con ImageDataGenerator

from tensorflow.keras.preprocessing.image import ImageDataGenerator

import numpy as np

#Crear el 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],

    validation\_split=0.2 #20% para pruebas

)

#Generadores para sets de entrenamiento y pruebas

data\_gen\_entrenamiento = datagen.flow\_from\_directory('/content/dataset', target\_size=(224,224),

                                                     batch\_size=32, shuffle=True, subset='training')

data\_gen\_pruebas = datagen.flow\_from\_directory('/content/dataset', target\_size=(224,224),

                                                     batch\_size=32, shuffle=True, subset='validation')

#Imprimir 10 imagenes del generador de entrenamiento

for imagen, etiqueta in data\_gen\_entrenamiento:

  for i in range(10):

    plt.subplot(2,5,i+1)

    plt.xticks([])

    plt.yticks([])

    plt.imshow(imagen[i])

  break

plt.show()

#aqui termina el proceso de creacion de data set

Entrenamiento:

Para el entrenamiento se crea tres neuronas que serán la que se entrenan y el resto de las neuronas del modelo mobile net se congelan

import tensorflow as tf

import tensorflow\_hub as hub

url = "https://tfhub.dev/google/tf2-preview/mobilenet\_v2/feature\_vector/4"

mobilenetv2 = hub.KerasLayer(url, input\_shape=(224,224,3))

#Congelar el modelo descargado

mobilenetv2.trainable = False

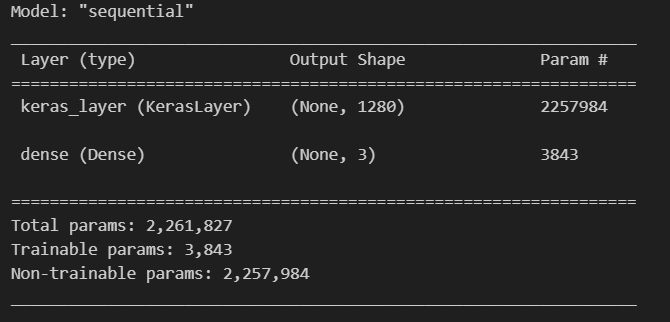
modelo = tf.keras.Sequential([

    mobilenetv2,

    tf.keras.layers.Dense(3, activation='softmax')

])

modelo.summary()



#Compilar como siempre

modelo.compile(

    optimizer='adam',

    loss='categorical\_crossentropy',

    metrics=['accuracy']

)

#Entrenar el modelo

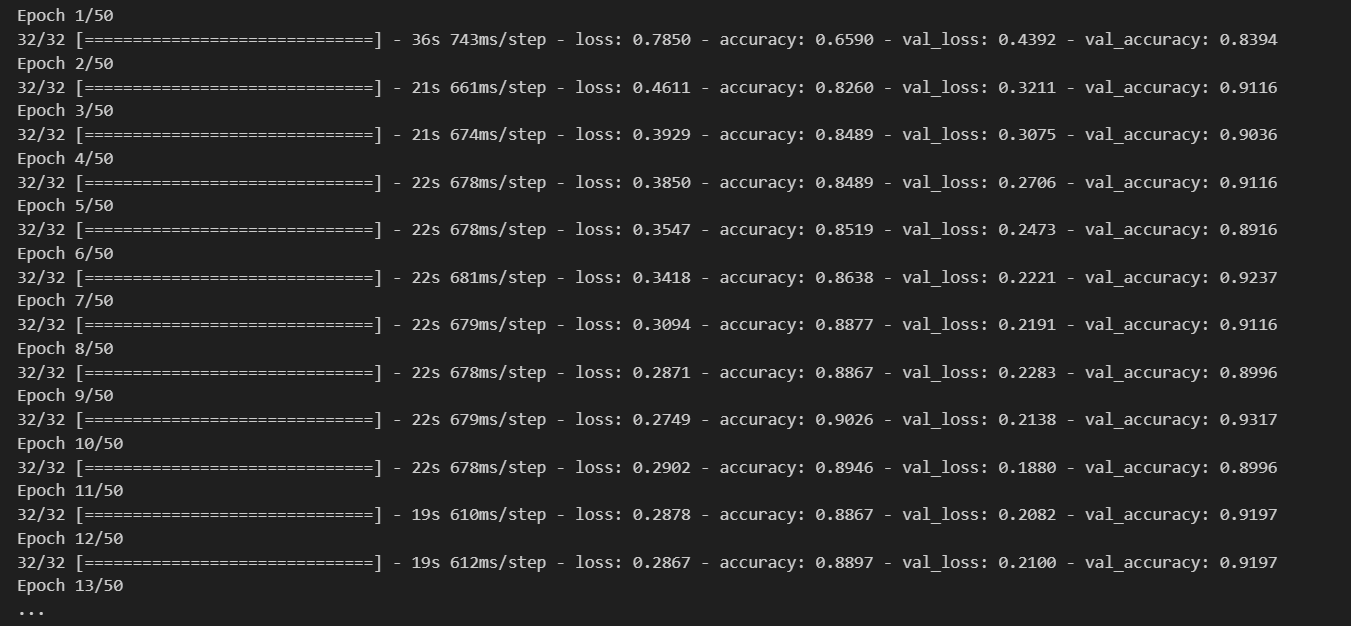
EPOCAS = 50

historial = modelo.fit(

    data\_gen\_entrenamiento, epochs=EPOCAS, batch\_size=32,

    validation\_data=data\_gen\_pruebas

)



Resultados:

Para mostrar los resultados ejecutamos las graficas de precisión

#Graficas de precisión

acc = historial.history['accuracy']

val\_acc = historial.history['val\_accuracy']

loss = historial.history['loss']

val\_loss = historial.history['val\_loss']

rango\_epocas = range(50)

plt.figure(figsize=(8,8))

plt.subplot(1,2,1)

plt.plot(rango\_epocas, acc, label='Precisión Entrenamiento')

plt.plot(rango\_epocas, val\_acc, label='Precisión Pruebas')

plt.legend(loc='lower right')

plt.title('Precisión de entrenamiento y pruebas')

plt.subplot(1,2,2)

plt.plot(rango\_epocas, loss, label='Pérdida de entrenamiento')

plt.plot(rango\_epocas, val\_loss, label='Pérdida de pruebas')

plt.legend(loc='upper right')

plt.title('Pérdida de entrenamiento y pruebas')

plt.show()

#Crear una carpeta para poder para exportarla a TensorFlow Serving

!mkdir -p carpeta\_salida/compost/1

#Guardar el modelo en formato SavedModel

modelo.save('carpeta\_salida/compost /1')

#Hacerlo un zip para bajarlo y usarlo en otro lado

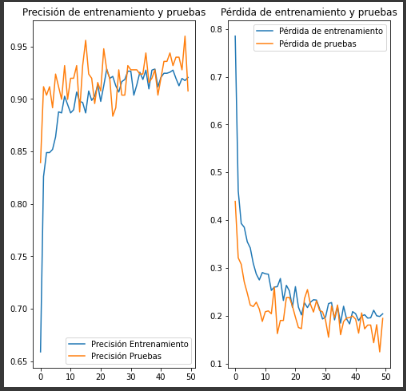
!zip -r modelo\_compost.zip /content/carpeta\_salida/compost/

# Resultados y Discusión

En los resultados se resume los datos compilados y el análisis de los datos que sean relevantes el discurso, presente con detalle los datos a fin de que pueda justificar las conclusiones.

## Resultados 1

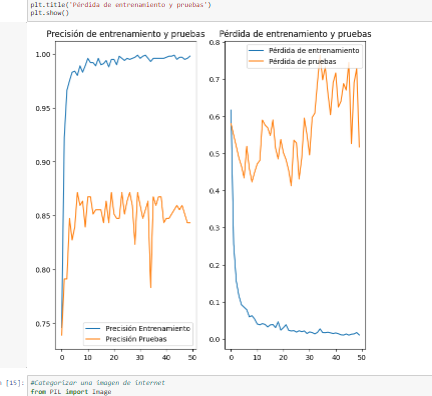
Se realizó un entrenamiento inicial de 50 épocas con el dataset1, monitoreando la pérdida y la precisión en el conjunto de validación



## Resultados 2

Se realizó un entrenamiento de 50 épocas, monitoreando la pérdida y la precisión en el conjunto de validación.

Donde los resultados no fueron muy satifactorios ya que se ve una perdida de entrenamiento bastante considerable lo cual reduce significa que nuestros datos fueron repetitivos para red neuronal



# Conclusiones

El análisis realizado ha demostrado que la implementación de redes neuronales como MobileNetV3 en la producción de compostaje puede mejorar significativamente la eficiencia y calidad del proceso. La red neuronal entrenada demostró una alta precisión en la clasificación de las fases del compostaje, lo cual facilita una mejor gestión de los residuos orgánicos y promueve prácticas sostenibles en los hogares. Las conclusiones del estudio sugieren que esta tecnología no solo es viable, sino que también puede ser escalada para aplicaciones más amplias en diferentes entornos de compostaje.

Recomendaciones

Agradecimientos

Referencias

Google. "MobileNetV3: A Neural Architecture for Efficient Neural Network". TensorFlow Blog. 8 de mayo de 2019. URL: https://www.tensorflow.org/api\_docs/python/tf/keras/applications/MobileNetV3Small

Howard, A., & Gupta, S. "Presentamos la próxima generación de modelos de visión en el dispositivo: MobileNetV3 y MobileNetEdgeTPU". Google Research Blog. 13 de noviembre de 2019. URL: https://research.google/blog/introducing-the-next-generation-of-on-device-vision-models-mobilenetv3-and-mobilenetedgetpu/