```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & I
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session


from numpy.random import seed
seed(101)

import pandas as pd
import numpy as np

import tensorflow

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Conv2D, MaxPooling2D, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint

import os
import cv2

import imageio
import skimage
import skimage.io
import skimage.transform

from sklearn.utils import shuffle
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
import itertools
import shutil
import matplotlib.pyplot as plt
%matplotlib inline


# Number of samples we will have in each class.
SAMPLE_SIZE = 224

# The images will all be resized to this size.
IMAGE_SIZE = 224


## se creo el modelo
```

```
# Source: https://www.kaggle.com/fmarazzi/baseline-keras-cnn-roc-fast-5min-0-8253-lb

kernel_size = (3,3)
pool_size= (2,2)
first_filters = 32
second_filters = 64
third_filters = 128

dropout_conv = 0.3
dropout_dense = 0.3


model = Sequential()
model.add(Conv2D(first_filters, kernel_size, activation = 'relu',
                 input_shape = (IMAGE_SIZE, IMAGE_SIZE, 3)))
model.add(Conv2D(first_filters, kernel_size, activation = 'relu'))
model.add(Conv2D(first_filters, kernel_size, activation = 'relu'))
model.add(MaxPooling2D(pool_size = pool_size))
model.add(Dropout(dropout_conv))

model.add(Conv2D(second_filters, kernel_size, activation ='relu'))
model.add(Conv2D(second_filters, kernel_size, activation ='relu'))
model.add(Conv2D(second_filters, kernel_size, activation ='relu'))
model.add(MaxPooling2D(pool_size = pool_size))
model.add(Dropout(dropout_conv))

model.add(Conv2D(third_filters, kernel_size, activation ='relu'))
model.add(Conv2D(third_filters, kernel_size, activation ='relu'))
model.add(Conv2D(third_filters, kernel_size, activation ='relu'))
model.add(MaxPooling2D(pool_size = pool_size))
model.add(Dropout(dropout_conv))

model.add(Flatten())
model.add(Dense(1024, activation = "relu"))
model.add(Dense(1024, activation = "relu"))
model.add(Dense(512, activation = "relu"))
model.add(Dropout(dropout_dense))
model.add(Dense(11, activation = "softmax"))

model.summary()
```

Mostrar salida oculta

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os

# Directorio que contiene todas las imágenes
data_dir = '/content/drive/MyDrive/estadistica/uco-plant-seedlings-classification-01-2024/train'

# Proporciones para la división de los datos
train_ratio = 0.7
val_ratio = 0.15
test_ratio = 0.15

# Crear generadores de datos de imagen para cargar imágenes del directorio
datagen = ImageDataGenerator(rescale=1./255, validation_split=val_ratio)

# Tamaño del lote (batch size) de imágenes a cargar en cada iteración
batch_size = 32
color_mode = 'rgb'

# Cargar imágenes del directorio de entrenamiento y validación usando el generador de datos
train_generator = datagen.flow_from_directory(
    data_dir,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    color_mode=color_mode,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training',
    shuffle=True)

val_generator = datagen.flow_from_directory(
    data_dir,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation')
```

```
Found 3619 images belonging to 11 classes.
Found 633 images belonging to 11 classes.
```

Haz doble clic (o pulsa Intro) para editar

```
mobile = keras.applications.mobilenet.MobileNet()
base_model=MobileNet(weights='imagenet',include_top=False) #imports the mobilenet model and discards the last 1000 neuron layer.

x=base_model.output
x=GlobalAveragePooling2D()(x)
x=Dense(1024,activation='relu')(x) #we add dense layers so that the model can learn more complex functions and classify for better results.
x=Dense(1024,activation='relu')(x) #dense layer 2
x=Dense(512,activation='relu')(x) #dense layer 3
preds=Dense(11,activation='softmax')(x) #final layer with softmax activation
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [128, 160, 192, 224]. Weights for input shape (224, 224
```

```
model.compile(Adam(learning_rate=0.0001), loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
filepath = "model.h5"
checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1,
                             save_best_only=True, mode='max')

reduce_lr = ReduceLROnPlateau(monitor='val_acc', factor=0.5, patience=3,
                                  verbose=1, mode='max', min_lr=0.00001)


callbacks_list = [checkpoint, reduce_lr]

# Calcular los pasos por época
#train_steps = len(train_generator)// 15
train_steps = 32
#val_steps = len(val_generator)// 15
val_steps = 32

# Establecer estos valores en fit_generator
history = model.fit_generator(train_generator, steps_per_epoch=train_steps,
                    validation_data=val_generator,
                    validation_steps=val_steps,
                    epochs=50, verbose=1,
                   callbacks=callbacks_list)
```

Mostrar salida oculta

```
from keras.preprocessing import image
import os
import numpy as np
import keras
from keras.models import load_model
from keras.preprocessing import image
import os


test_data_dir = '/content/drive/MyDrive/estadistica/uco-plant-seedlings-classification-01-2024/test'
def preprocess_image(img_path):
    img = image.load_img(img_path, target_size=(IMAGE_SIZE, IMAGE_SIZE))
#load_img(img_path, target_size=(img_width, img_height))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    return img_array

# Hacer predicciones sobre las imágenes de prueba
def predict_images(model, test_data_dir):
    predictions = []
    image_paths = [os.path.join(test_data_dir, f) for f in os.listdir(test_data_dir) if f.endswith('.jpg') or f.endswith('.png')]
    for img_path in image_paths:
        img = preprocess_image(img_path)
        prediction = model.predict(img).argmax()
        predictions.append(prediction)
    return predictions

# Obtener las predicciones
predictions = predict_images(model, test_data_dir)

# Imprimir las predicciones
for i, prediction in enumerate(predictions):
    print(f"Predicción para imagen {i+1}: {prediction}")
```

Mostrar salida oculta

```
def preprocess_image(img_path):
    img = image.load_img(img_path, target_size=(IMAGE_SIZE, IMAGE_SIZE))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    return img_array

# Hacer predicciones sobre las imágenes de prueba
def predict_images(model, test_data_dir):
    predictions = []
    image_paths = [os.path.join(test_data_dir, f) for f in os.listdir(test_data_dir) if f.endswith('.jpg') or f.endswith('.png')]
    for img_path in image_paths:
        img = preprocess_image(img_path)
        prediction = np.argmax(model.predict(img))
        predictions.append(prediction)
    return predictions, image_paths

# Obtener las predicciones
predictions, image_paths = predict_images(model, test_data_dir)
```

Mostrar salida oculta

```
df = pd.DataFrame({'Image_Name': image_paths, 'Predicted_Category': predictions})
df
```

	Image_Name	Predicted_Category
0	/content/drive/MyDrive/estadistica/uco-plant-s...	3
1	/content/drive/MyDrive/estadistica/uco-plant-s...	1
2	/content/drive/MyDrive/estadistica/uco-plant-s...	7
3	/content/drive/MyDrive/estadistica/uco-plant-s...	10
4	/content/drive/MyDrive/estadistica/uco-plant-s...	5
...	...	...
1034	/content/drive/MyDrive/estadistica/uco-plant-s...	3
1035	/content/drive/MyDrive/estadistica/uco-plant-s...	5
1036	/content/drive/MyDrive/estadistica/uco-plant-s...	7
1037	/content/drive/MyDrive/estadistica/uco-plant-s...	1
1038	/content/drive/MyDrive/estadistica/uco-plant-s...	5

1039 rows × 2 columns

```
import os
import pandas as pd

# Supongamos que 'df' es tu DataFrame de pandas y 'Ruta' es el nombre de la columna que contiene las rutas completas de los archivos
df['Image_Name'] = df['Image_Name'].apply(lambda x: os.path.basename(x))

# Ahora 'Nombre_archivo' contendrá solo los nombres de los archivos
df
```

	Image_Name	Predicted_Category

```
clases = os.listdir('/content/drive/MyDrive/estadistica/uco-plant-seedlings-classification-01-2024/train')
```

```
df['Predicted_Category'] = df['Predicted_Category'].apply(lambda x: clases[x])
```

```
df.head()
```

	Image_Name	Predicted_Category
0	0b53d3421f971bf9e30b2decae7842dd.png	Common Chickweed
1	0e7dea5901b218d61837cddc85c54459.png	Charlock
2	04aebf9632278c3ab55424709aa458dc.png	Scentless Mayweed
3	09d0cd1027584e266892bf340b6918cd.png	Sugar beet
4	0c8098935a70243273e2d4c3593acb33.png	Loose Silky-bent

```
import pandas as pd

# Supongamos que tienes un DataFrame llamado 'df' con columnas 'A', 'B' y 'C'

# Cambiar los nombres de las columnas
nuevos_nombres = ['file', 'species']
df.columns = nuevos_nombres

# Ahora las columnas del DataFrame se llamarán 'Columna1', 'Columna2' y 'Columna3'
df
```

	file	species
0	0b53d3421f971bf9e30b2decae7842dd.png	Common Chickweed
1	0e7dea5901b218d61837cddc85c54459.png	Charlock
2	04aebf9632278c3ab55424709aa458dc.png	Scentless Mayweed
3	09d0cd1027584e266892bf340b6918cd.png	Sugar beet
4	0c8098935a70243273e2d4c3593acb33.png	Loose Silky-bent
...	...	...
1034	0696130b21de4398b1f8ffa728351f61.png	Common Chickweed
1035	084d46562156eb60ff6547d7a9a5d00d.png	Loose Silky-bent
1036	07e6d5be4c9c864c4196b64fe72b4a31.png	Scentless Mayweed
1037	06a17690fe90e18ee3f62b82d8cd3ae8.png	Charlock
1038	0e3abd413ad0e37a7721568c72fcdf2a.png	Loose Silky-bent

1039 rows × 2 columns

```
df.to_csv('results.csv', index=False)
```

Empieza a programar o a crear código con IA.

```
import os
import pandas as pd
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Cargar el modelo entrenado
# Suponiendo que tienes el modelo ya cargado en una variable llamada 'model'
```