▾ SOIL CLASSIFICATION

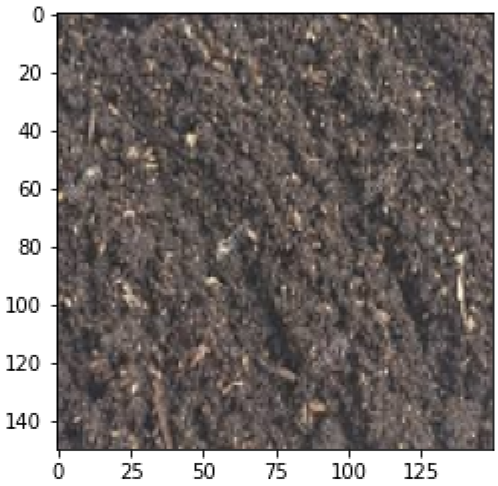
+ Code \+ Text

```
import os
from warnings import filterwarnings
filterwarnings('ignore')

from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report


import tensorflow as tf
import keras

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Activation,Dropout,MaxPool2D,Conv2D,MaxPooling2D,Flatten
from tensorflow.keras.optimizers import Adam,RMSprop,SGD,Adagrad
from tensorflow.keras.preprocessing.image import ImageDataGenerator

import cv2
import numpy as np
import matplotlib.pyplot as plt


img_width , img_height = 150,150
batch_size = 32
epochs = 10


path = "/content/sample_data/Soil_Dataset"


train_data_dir ="/content/sample_data/Soil_Dataset/train"
test_data_dir ="/content/sample_data/Soil_Dataset/train"


from glob import glob
glob("/content/sample_data/Soil_Dataset/test/*/")
```

```
['/content/sample_data/Soil_Dataset/test/Cinder soil/',
 '/content/sample_data/Soil_Dataset/test/Black soil/',
 '/content/sample_data/Soil_Dataset/test/Yellow soil/',
 '/content/sample_data/Soil_Dataset/test/Peat soil/',
 '/content/sample_data/Soil_Dataset/test/Laterite soil/']
```

```
labels = ['Cinder soil','Black soil','Laterite soil']


def show_soil(path):
  im = cv2.imread(path)
  im_resized = cv2.resize(im, (img_height,img_width), interpolation=cv2.INTER_LINEAR)

  plt.imshow(cv2.cvtColor(im_resized, cv2.COLOR_BGR2RGB))
  plt.show()
```

```
show_soil("/content/sample_data/Soil_Dataset/train/Black soil/10.jpg")
```



```
trainGen = ImageDataGenerator(rescale=1./255,shear_range=0.2,horizontal_flip=True,zoom_range=0.2)
testGen = ImageDataGenerator(rescale=1./255)
```

```
train = trainGen.flow_from_directory(train_data_dir,target_size=(img_height,img_width),classes=labels,c
test = testGen.flow_from_directory(test_data_dir,target_size=(img_height,img_width),classes=labels,clas

     Found 58 images belonging to 3 classes.
     Found 58 images belonging to 3 classes.
```

```
model = Sequential()
model.add(Conv2D(128, (3,3),input_shape=(img_height,img_width,3),padding='same',activation='relu'))
model.add(Activation('relu'))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(64,3,3,padding='same',activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.2))

model.add(Conv2D(32,3,3,padding='same',activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.2))

model.add(Dense(32,activation='relu'))

model.add(Flatten())
model.add(Dense(32,activation='relu'))

model.add(Dense(3,activation='softmax'))

model.summary()

     Model: "sequential_1"
     _________________________________________________________________
     Layer (type)                 Output Shape              Param #
     =================================================================
     conv2d_3 (Conv2D)            (None, 150, 150, 128)     3584
```

```
_________________________________________________________________
activation_1 (Activation)    (None, 150, 150, 128)     0         
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 75, 75, 128)       0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 25, 25, 64)        73792     
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 12, 12, 64)        0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 12, 12, 64)        0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 4, 4, 32)          18464     
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 2, 2, 32)          0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 2, 2, 32)          0         
_________________________________________________________________
dense_3 (Dense)              (None, 2, 2, 32)          1056      
_________________________________________________________________
flatten_1 (Flatten)          (None, 128)               0         
_________________________________________________________________
dense_4 (Dense)              (None, 32)                4128      
_________________________________________________________________
dense_5 (Dense)              (None, 3)                 99        
=================================================================
Total params: 101,123
Trainable params: 101,123
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

```
model.fit(train,epochs=15,validation_data=test)
```

```
Epoch 1/15
2/2 [==============================] - 6s 3s/step - loss: 1.0985 - acc: 0.3103 - val_loss: 1.0797
Epoch 2/15
2/2 [==============================] - 4s 2s/step - loss: 1.0764 - acc: 0.3448 - val_loss: 1.0559
Epoch 3/15
2/2 [==============================] - 4s 2s/step - loss: 1.0621 - acc: 0.4483 - val_loss: 1.0083
Epoch 4/15
2/2 [==============================] - 4s 2s/step - loss: 1.0194 - acc: 0.5172 - val_loss: 0.9475
Epoch 5/15
2/2 [==============================] - 4s 2s/step - loss: 0.9609 - acc: 0.5517 - val_loss: 0.8556
Epoch 6/15
2/2 [==============================] - 5s 2s/step - loss: 0.8762 - acc: 0.5862 - val_loss: 0.7625
Epoch 7/15
2/2 [==============================] - 7s 3s/step - loss: 0.8031 - acc: 0.5862 - val_loss: 0.6902
Epoch 8/15
2/2 [==============================] - 4s 2s/step - loss: 0.6849 - acc: 0.6207 - val_loss: 0.6161
Epoch 9/15
2/2 [==============================] - 4s 2s/step - loss: 0.6250 - acc: 0.6034 - val_loss: 0.5689
Epoch 10/15
2/2 [==============================] - 4s 2s/step - loss: 0.6507 - acc: 0.6034 - val_loss: 0.5556
Epoch 11/15
2/2 [==============================] - 4s 2s/step - loss: 0.5582 - acc: 0.6207 - val_loss: 0.5661
Epoch 12/15
2/2 [==============================] - 4s 2s/step - loss: 0.5145 - acc: 0.6207 - val_loss: 0.5131
```

```
Epoch 13/15
2/2 [==============================] - 4s 2s/step - loss: 0.5146 - acc: 0.6379 - val_loss: 0.5492
Epoch 14/15
2/2 [==============================] - 4s 2s/step - loss: 0.5133 - acc: 0.6379 - val_loss: 0.5080
Epoch 15/15
2/2 [==============================] - 4s 2s/step - loss: 0.4911 - acc: 0.6379 - val_loss: 0.4797
<tensorflow.python.keras.callbacks.History at 0x7f12d285ffd0>
```

```
model.history.history.keys()
```

```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

```
train_loss = model.history.history['loss']
train_loss
```

```
[1.098689214936618,
 1.0770509530758035,
 1.0613567459172215,
 1.0161742259716164,
 0.959354739764641,
 0.8772686247167916,
 0.801072174105151,
 0.6895739682789507,
 0.6222904920578003,
 0.6492152152390316,
 0.56047673677576,
 0.5127051369897251,
 0.5077928982931992,
 0.5156958164839909,
 0.4852605749820841]
```

```
train_acc = model.history.history['acc']
train_acc
```

```
[0.31034482,
 0.3448276,
 0.44827586,
 0.51724136,
 0.55172414,
 0.5862069,
 0.5862069,
 0.62068963,
 0.6034483,
 0.6034483,
 0.62068963,
 0.62068963,
 0.63793105,
 0.63793105,
 0.63793105]
```

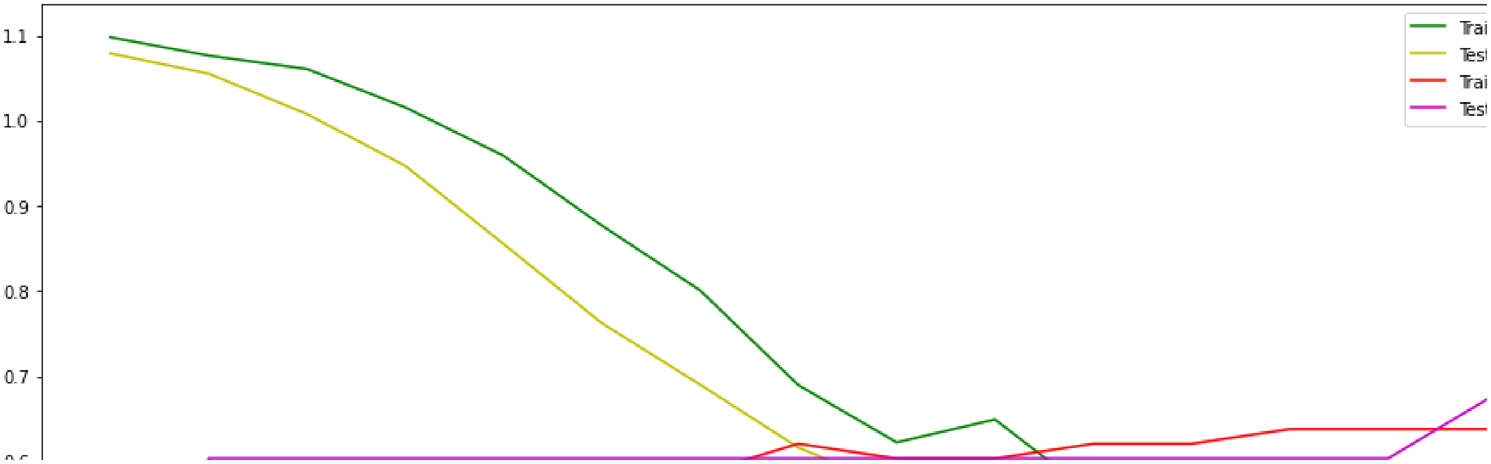
```
test_loss = model.history.history['val_loss']
test_loss
```

```
[1.0796663761138916,
 1.0558983087539673,
      1.0083120465278625,
      0.9474844932556152,
      0.8556050360202789,
      0.7625182867050171,
      0.690228283405304,
      0.616146057844162,
      0.5689255148172379,
      0.5556299537420273,
      0.5660660564899445,
      0.5130860507488251,
      0.5491515398025513,
      0.5079878270626068,
      0.4796639382839203]
```

```
test_acc = model.history.history['val_acc']
test_acc
```

```
     [0.36206895,
      0.6034483,
      0.6034483,
      0.6034483,
      0.6034483,
      0.6034483,
      0.6034483,
      0.6034483,
      0.6034483,
      0.6034483,
      0.6034483,
      0.6034483,
      0.6034483,
      0.6034483,
      0.67241377]
```

```
plt.figure(figsize=(16,8))
epochs=15
plt.plot(np.arange(epochs),train_loss,color='g',label='Train_loss')
plt.plot(np.arange(epochs),test_loss,color='y',label='Test_loss')
plt.plot(np.arange(epochs),train_acc,color='r',label='Train_acc')
plt.plot(np.arange(epochs),test_acc,color='m',label='Test_acc')
plt.legend()
plt.xlabel('Epochs')
plt.show()
```



```
loss,acc = model.evaluate(test)
print('Test Loss:' ,loss)
print('Test Acc:' ,acc)
```

```
2/2 [==============================] - 1s 455ms/step - loss: 0.4859 - acc: 0.6724
Test Loss: 0.48594602942466736
Test Acc: 0.67241377
```

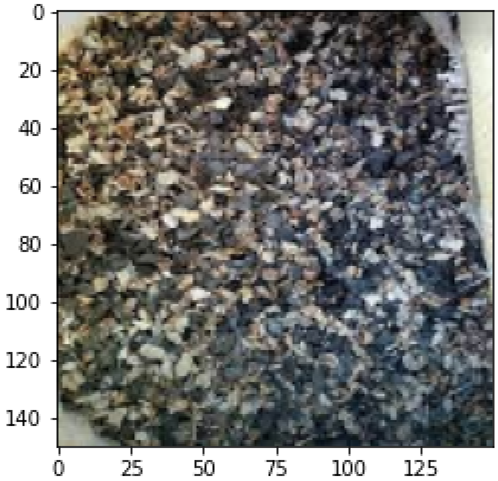
```
labels
```

```
['Cinder soil', 'Black soil', 'Laterite soil']
```

```
def predict_soil(path):
  im = cv2.imread(path)
  im_resized = cv2.resize(im, (img_height,img_width), interpolation=cv2.INTER_LINEAR)

  plt.imshow(cv2.cvtColor(im_resized, cv2.COLOR_BGR2RGB))
  plt.show()

  img_pred = image.load_img(path,target_size=(img_height,img_width))
  img_pred = image.img_to_array(img_pred)
  img = np.expand_dims(img_pred,axis=0)
  result = model.predict_classes(img)
  prob = model.predict_proba(img)
  print('Predicted class: ',result)
  print('Probability:{}'.format(prob[0]))
  if result[0]==0:
    prediction ="Cinder soil"
  elif result[0]==1:
     prediction ="Black soil"
  else:
     prediction ="Laterite soil "

  print('Predicted Class:',prediction)
```

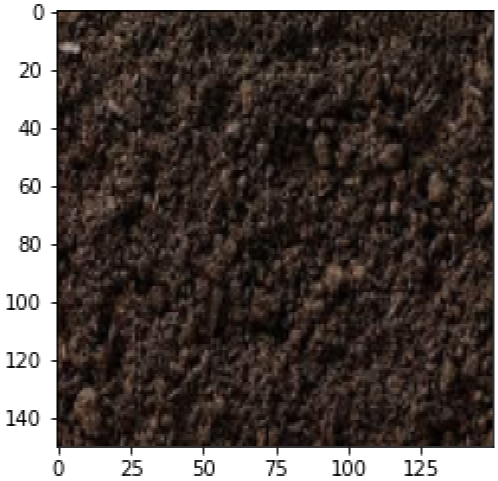
```
from keras.preprocessing import image
```

```
predict_soil("/content/sample_data/Soil_Dataset/train/Cinder soil/1.jpg")
```



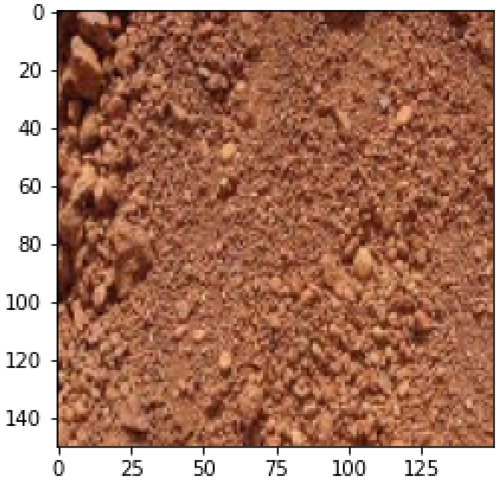
```
    Predicted class:  [0]
    Probability:[1. 0. 0.]
    Predicted Class: Cinder soil
```

```
predict_soil("/content/sample_data/Soil_Dataset/test/Black soil/26.jpg")
```



```
    Predicted class:  [1]
    Probability:[4.7452536e-06 9.9999523e-01 0.0000000e+00]
    Predicted Class: Black soil
```

```
predict_soil("/content/sample_data/Soil_Dataset/test/Laterite soil/23.jpg")
```



```
Predicted class:  [2]
Probability:[0. 0. 1.]
Predicted Class: Laterite soil
```