```
import os
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
import shutil
import matplotlib.pyplot as plt
%matplotlib inline
from tensorflow.keras.preprocessing.image import load_img, img_to_array, array_to_img, ImageDataGenerator
from keras.applications.vgg16 import VGG16
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, InputLayer
from keras.models import Sequential
from keras.layers import BatchNormalization
from keras import optimizers
```

```
# remove when running on laptop
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
main_path = '/content/gdrive/MyDrive/Groundnut'
groundnut_healthy = main_path + '/Gnut healthy/'
groundnut_leaf_spot = main_path + '/Gnut leaf spot/'

groundnut_healthy_data = [main_path+'/Gnut healthy/'+f for f in os.listdir(groundnut_healthy)]
groundnut_leaf_spot_data = [main_path+'/Gnut leaf spot/'+f for f in os.listdir(groundnut_leaf_spot)]
len(groundnut_healthy_data),len(groundnut_leaf_spot_data)
```

```
(503, 500)
```

```
print(groundnut_healthy_data[0:5])
print(groundnut_leaf_spot_data[0:5])
```

```
['/content/gdrive/MyDrive/Groundnut/Gnut healthy/IMG_20221014_083419.jpg', '/content/gdrive/MyDrive/Groundnut/Gnut healthy/IMG_20221
['/content/gdrive/MyDrive/Groundnut/Gnut leaf spot/IMG_20210818_111250.jpg', '/content/gdrive/MyDrive/Groundnut/Gnut leaf spot/IMG_2
```

```
images = np.array(groundnut_healthy_data + groundnut_leaf_spot_data)
labels = np.array([0]*len(groundnut_healthy_data)+[1]*len(groundnut_leaf_spot_data)).astype('float32')
```

```
validation_ratio = 0.2
X_train, X_test, y_train, y_test = train_test_split(images,labels,test_size=0.33)
X_train, X_validation, y_train, y_validation = train_test_split(X_train,y_train,test_size=validation_ratio)
```

```
X_train.shape,X_test.shape,y_train.shape, y_test.shape
```

```
((537,), (331,), (537,), (331,))
```

```
len(X_train),len(X_test),len(y_train),len(y_test)
```

```
(537, 331, 537, 331)
```

```
from tensorflow.keras.preprocessing.image import load_img, img_to_array, array_to_img, ImageDataGenerator
```

```
IMG_WIDTH=224
IMG_HEIGHT=224
IMG_DIM = (IMG_WIDTH, IMG_HEIGHT)
```

```
X_train = np.array([img_to_array(load_img(img, target_size=IMG_DIM)).astype('float32')/255 for img in X_train])
X_test = np.array([img_to_array(load_img(img, target_size=IMG_DIM)).astype('float32')/255  for img in X_test])
X_validation = np.array([img_to_array(load_img(img,target_size=IMG_DIM)).astype('float32')/255 for img in X_validation])
```

```
dataGen = ImageDataGenerator(width_shift_range=0.2,
                             # 0.1 = 10%     IF MORE THAN 1 E.G 10 THEN IT REFFERS TO NO. OF  PIXELS EG 10 PIXELS
                             height_shift_range=0.6,
                             zoom_range=0.3,  # 0.2 MEANS CAN GO FROM 0.8 TO 1.2
                             shear_range=0.4,  # MAGNITUDE OF SHEAR ANGLE
                             rotation_range=10)  # DEGREES
dataGen.fit(X_train)


batches = dataGen.flow(X_train, y_train,batch_size=20)  # REQUESTING DATA GENRATOR TO GENERATE IMAGES  BATCH SIZE = NO. OF IMAGES CREAEI
X_batch, y_batch = next(batches)


array_to_img(X_train[0])
```

```
print(X_train[:10],y_train[-10:])
```

```
[0.45882353 0.50980395 0.44313726]]]] [0. 0. 1. 0. 0. 0. 1. 0. 0. 1.]
```

```
from keras.models import Model
vgg = VGG16(include_top=False, weights='imagenet', input_shape = (IMG_HEIGHT,IMG_WIDTH,3))
output1 = vgg.layers[-1].output
output2 = Flatten()(output1)
vgg = Model(vgg.input, output2)
for layer in vgg.layers:
    layer.trainable = False
vgg.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_no
58889256/58889256 [==============================] - 0s 0us/step
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
_________________________________________________________________
```

```
model = Sequential()
model.add(vgg)
input_shape = (IMG_HEIGHT,IMG_WIDTH,3)
model.add(Flatten())
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))
```

```
flatten_layer = Flatten()
dense_layer_1 = Dense(500, activation='relu')
dense_layer_2 = Dense(500, activation='relu')
dropout_layer = Dropout(0.2)
prediction_layer = Dense(1, activation='sigmoid')
```

```
model = Sequential([
    vgg,
    flatten_layer,
    dense_layer_1,
    dense_layer_2,
    dropout_layer,
```

```
    prediction_layer
])

model.compile(loss='binary_crossentropy', optimizer=optimizers.Adam(learning_rate=0.001), metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 model (Functional)          (None, 25088)             14714688

 flatten_1 (Flatten)         (None, 25088)             0

 dense (Dense)               (None, 500)               12544500

 dense_1 (Dense)             (None, 500)               250500

 dropout (Dropout)           (None, 500)               0

 dense_2 (Dense)             (None, 1)                 501

=================================================================
Total params: 27,510,189
Trainable params: 12,795,501
Non-trainable params: 14,714,688
_________________________________________________________________
```

```
history = model.fit(dataGen.flow(X_train, y_train.reshape((-1,1))),
                    epochs=100,
                    validation_data=(X_validation,y_validation),
                    shuffle=True,
                    verbose=1)
```

```
Epoch 1/100
17/17 [==============================] - 7s 396ms/step - loss: 0.1559 - accuracy: 0.9404 - val_loss: 0.2197 - val_accuracy: 0.92
Epoch 2/100
17/17 [==============================] - 8s 486ms/step - loss: 0.1166 - accuracy: 0.9516 - val_loss: 0.3733 - val_accuracy: 0.87
Epoch 3/100
17/17 [==============================] - 8s 443ms/step - loss: 0.1395 - accuracy: 0.9423 - val_loss: 0.1850 - val_accuracy: 0.93
Epoch 4/100
17/17 [==============================] - 7s 383ms/step - loss: 0.1173 - accuracy: 0.9590 - val_loss: 0.2035 - val_accuracy: 0.91
Epoch 5/100
17/17 [==============================] - 8s 469ms/step - loss: 0.1405 - accuracy: 0.9534 - val_loss: 0.2050 - val_accuracy: 0.94
Epoch 6/100
17/17 [==============================] - 7s 394ms/step - loss: 0.1312 - accuracy: 0.9553 - val_loss: 0.2587 - val_accuracy: 0.91
Epoch 7/100
17/17 [==============================] - 8s 459ms/step - loss: 0.1556 - accuracy: 0.9367 - val_loss: 0.4128 - val_accuracy: 0.85
Epoch 8/100
17/17 [==============================] - 7s 381ms/step - loss: 0.1399 - accuracy: 0.9441 - val_loss: 0.2322 - val_accuracy: 0.91
Epoch 9/100
17/17 [==============================] - 8s 450ms/step - loss: 0.0991 - accuracy: 0.9665 - val_loss: 0.1810 - val_accuracy: 0.95
Epoch 10/100
17/17 [==============================] - 7s 390ms/step - loss: 0.1142 - accuracy: 0.9553 - val_loss: 0.2541 - val_accuracy: 0.89
Epoch 11/100
17/17 [==============================] - 8s 454ms/step - loss: 0.1058 - accuracy: 0.9572 - val_loss: 0.3270 - val_accuracy: 0.89
Epoch 12/100
17/17 [==============================] - 7s 381ms/step - loss: 0.1358 - accuracy: 0.9460 - val_loss: 0.1922 - val_accuracy: 0.94
Epoch 13/100
17/17 [==============================] - 8s 474ms/step - loss: 0.1178 - accuracy: 0.9460 - val_loss: 0.4225 - val_accuracy: 0.88
Epoch 14/100
17/17 [==============================] - 7s 404ms/step - loss: 0.1216 - accuracy: 0.9479 - val_loss: 0.1696 - val_accuracy: 0.94
Epoch 15/100
17/17 [==============================] - 7s 390ms/step - loss: 0.1315 - accuracy: 0.9590 - val_loss: 0.2245 - val_accuracy: 0.92
Epoch 16/100
17/17 [==============================] - 8s 463ms/step - loss: 0.1035 - accuracy: 0.9665 - val_loss: 0.1850 - val_accuracy: 0.95
Epoch 17/100
17/17 [==============================] - 7s 382ms/step - loss: 0.0846 - accuracy: 0.9739 - val_loss: 0.1387 - val_accuracy: 0.94
Epoch 18/100
17/17 [==============================] - 8s 454ms/step - loss: 0.0996 - accuracy: 0.9572 - val_loss: 0.2337 - val_accuracy: 0.93
Epoch 19/100
17/17 [==============================] - 6s 376ms/step - loss: 0.0866 - accuracy: 0.9646 - val_loss: 0.1470 - val_accuracy: 0.94
Epoch 20/100
17/17 [==============================] - 8s 454ms/step - loss: 0.0901 - accuracy: 0.9665 - val_loss: 0.2041 - val_accuracy: 0.94
Epoch 21/100
17/17 [==============================] - 7s 385ms/step - loss: 0.0919 - accuracy: 0.9609 - val_loss: 0.1738 - val_accuracy: 0.93
Epoch 22/100
17/17 [==============================] - 7s 388ms/step - loss: 0.0951 - accuracy: 0.9665 - val_loss: 0.3838 - val_accuracy: 0.89
Epoch 23/100
17/17 [==============================] - 8s 461ms/step - loss: 0.1787 - accuracy: 0.9423 - val_loss: 0.1875 - val_accuracy: 0.92
Epoch 24/100
17/17 [==============================] - 7s 383ms/step - loss: 0.1262 - accuracy: 0.9516 - val_loss: 0.2977 - val_accuracy: 0.90
Epoch 25/100
17/17 [==============================] - 8s 454ms/step - loss: 0.0805 - accuracy: 0.9683 - val_loss: 0.2917 - val_accuracy: 0.91
Epoch 26/100
17/17 [==============================] - 8s 464ms/step - loss: 0.1208 - accuracy: 0.9516 - val_loss: 0.3856 - val_accuracy: 0.88
Epoch 27/100
```

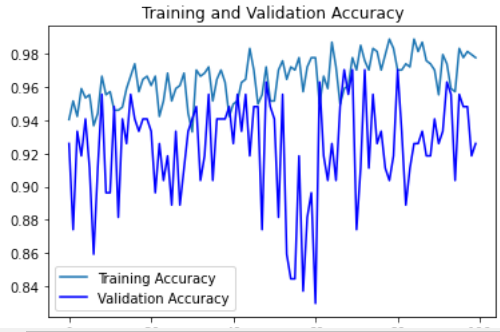
```
17/17 [==============================] - 7s 427ms/step - loss: 0.0963 - accuracy: 0.9590 - val_loss: 0.2167 - val_accuracy: 0.93
Epoch 28/100
17/17 [==============================] - 7s 408ms/step - loss: 0.1050 - accuracy: 0.9609 - val_loss: 0.4372 - val_accuracy: 0.88
```

```
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
epochs = range(len(accuracy))

plt.plot(epochs, accuracy, label='Training Accuracy')
plt.plot(epochs, val_accuracy, 'b', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
```

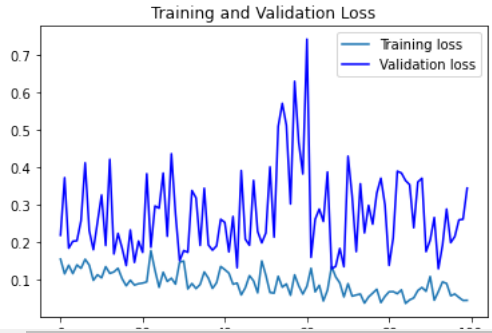
```
<matplotlib.legend.Legend at 0x7fed1820cdc0>
```




```
epochs = range(len(loss))

plt.plot(epochs, loss, label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and Validation Loss')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fed18a43400>
```




```
score = model.evaluate(X_test,y_test)
print('Test Accuracy: {}'.format(score[1]))
```

```
11/11 [==============================] - 1s 121ms/step - loss: 0.4112 - accuracy: 0.9245
Test Accuracy: 0.9244713187217712
```

```
predict = model.predict(X_test)
predict
```

```
       [1.27108957e-04],
       [1.00000000e+00],
       [1.78698465e-04],
       [3.07445767e-08],
       [6.70844456e-13],
       [1.00000000e+00],
       [3.11666710e-08],
       [9.99976993e-01],
       [9.99976993e-01],
       [4.33685466e-12],
       [6.61135313e-11],
       [9.82998013e-01],
       [9.03754413e-01],
       [9.99993682e-01],
       [1.00000000e+00],
       [4.53243149e-04],
       [6.05646349e-12],
       [1.29514275e-08],
       [1.09303989e-11],
       [3.39909420e-11],
       [1.83572120e-04],
       [1.00000000e+00],
       [5.80272972e-13],
       [5.09401737e-03],
       [9.99981284e-01],
       [1.00000000e+00],
       [9.99999166e-01],
       [1.00000000e+00],
       [9.99999642e-01],
       [9.83431280e-01],
       [1.56052843e-01],
       [1.00000000e+00],
       [9.98243093e-01],
       [1.00000000e+00],
       [1.00000000e+00],
       [9.60044563e-01],
       [9.99999762e-01],
       [9.99941111e-01],
       [1.00000000e+00],
       [9.99056995e-01],
       [9.99954700e-01]], dtype=float32)
```

```
# remove when running on laptop
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
import tensorflow as tf
#model.save('/content/gdrive/MyDrive/CNN/groundnutmodel.h5')
model = tf.keras.models.load_model('/content/gdrive/MyDrive/CNN/groundnutmodel.h5')
```

```
x = []
for i in predict:
  if i<=0.5:
    x.append(0)
  else:
    x.append(1)
x = np.array(x).astype('float32')
```

```
x[0:10]
```

```
array([0., 0., 1., 0., 0., 0., 1., 0., 0., 1.], dtype=float32)
```

```
X_test[0:10]
```

```
from sklearn.metrics import accuracy_score
```

```
acc = accuracy_score(x,y_test)
print('Accuracy Score : ',acc*100)
```

```
Accuracy Score :  92.44712990936556
```

```
!pip install gradio
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting gradio
  Downloading gradio-3.27.0-py3-none-any.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 83.8 MB/s eta 0:00:00
Requirement already satisfied: markdown-it-py[linkify]>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from gradio) (2.2.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packages (from gradio) (3.1.2)
```

```
Collecting ffmpy
  Downloading ffmpy-0.3.0.tar.gz (4.8 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from gradio) (1.22.4)
Collecting aiohttp
  Downloading aiohttp-3.8.4-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 64.2 MB/s eta 0:00:00
Collecting orjson
  Downloading orjson-3.8.10-cp39-cp39-manylinux_2_28_x86_64.whl (140 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 140.5/140.5 kB 18.1 MB/s eta 0:00:00
Collecting gradio-client>=0.1.3
  Downloading gradio_client-0.1.3-py3-none-any.whl (286 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 286.2/286.2 kB 31.2 MB/s eta 0:00:00
Collecting aiofiles
  Downloading aiofiles-23.1.0-py3-none-any.whl (14 kB)
Requirement already satisfied: pillow in /usr/local/lib/python3.9/dist-packages (from gradio) (8.4.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.9/dist-packages (from gradio) (3.7.1)
Collecting semantic-version
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Collecting httpx
  Downloading httpx-0.24.0-py3-none-any.whl (75 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.3/75.3 kB 10.1 MB/s eta 0:00:00
Collecting pydub
  Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Collecting websockets>=10.0
  Downloading websockets-11.0.2-cp39-cp39-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.7/129.7 kB 16.8 MB/s eta 0:00:00
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from gradio) (2.27.1)
Collecting uvicorn
  Downloading uvicorn-0.21.1-py3-none-any.whl (57 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.8/57.8 kB 6.6 MB/s eta 0:00:00
Collecting mdit-py-plugins<=0.3.3
  Downloading mdit_py_plugins-0.3.3-py3-none-any.whl (50 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.5/50.5 kB 6.8 MB/s eta 0:00:00
Collecting python-multipart
  Downloading python_multipart-0.0.6-py3-none-any.whl (45 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45.7/45.7 kB 4.1 MB/s eta 0:00:00
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (from gradio) (1.5.3)
Collecting fastapi
  Downloading fastapi-0.95.1-py3-none-any.whl (56 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.0/57.0 kB 7.2 MB/s eta 0:00:00
Requirement already satisfied: pyyaml in /usr/local/lib/python3.9/dist-packages (from gradio) (6.0)
Collecting huggingface-hub>=0.13.0
  Downloading huggingface_hub-0.13.4-py3-none-any.whl (200 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 200.1/200.1 kB 25.1 MB/s eta 0:00:00
Requirement already satisfied: pydantic in /usr/local/lib/python3.9/dist-packages (from gradio) (1.10.7)
Requirement already satisfied: altair>=4.2.0 in /usr/local/lib/python3.9/dist-packages (from gradio) (4.2.2)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.9/dist-packages (from gradio) (4.5.0)
Requirement already satisfied: markupsafe in /usr/local/lib/python3.9/dist-packages (from gradio) (2.1.2)
Requirement already satisfied: entrypoints in /usr/local/lib/python3.9/dist-packages (from altair>=4.2.0->gradio) (0.4)
Requirement already satisfied: toolz in /usr/local/lib/python3.9/dist-packages (from altair>=4.2.0->gradio) (0.12.0)
```

```
import gradio as gr
import cv2
import numpy as np
```

```
def groundnut_disease(img):
  img = np.asarray(img)
  img = cv2.resize(img,(224,224))
  img = img.reshape(1,224,224,3)
  while True:
    predict = model.predict(img)
    if predict<=0.5:
      return 'Healthy crop'
    elif predict>=0.5:
      return 'Leaf spot'
```

```
#outputs = gr.output.Textbox()
app = gr.Interface(fn=groundnut_disease,inputs="image",outputs="text",description='This is Ground disease classification model')
```

```
app.launch(debug=True)
```

```
Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set debug=False in launch
Note: opening Chrome Inspector may crash demo inside Colab notebooks.

To create a public link, set `share=True` in `launch()`.
1/1 [==============================] - 9s 9s/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 19ms/step
```