```
import cv2
import os
import numpy as np
```

```
l = os.listdir("/content/covid")
```

```
l
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
dataset = []
for i in l:
  d = {}
  img = cv2.imread(f"/content/covid/{i}",0)
  img = cv2.resize(img, (256, 256))
  d["img"] = img
  d["label"] = 1
  dataset.append(d)
```

```
dataset
```

```
l1 = os.listdir("/content/normal")
print(l1)
```

```
for i in l1:
  d = {}
  img = cv2.imread(f"/content/normal/{i}",0)
  img = cv2.resize(img, (256, 256))
  d["img"] = img
  d["label"] = 0
```

```
  dataset.append(d)

dataset

import pandas as pd

df = pd.DataFrame(dataset)

df.to_csv('data.csv')

df.head()
```

	img	label
0	[[1, 2, 3, 3, 4, 6, 6, 9, 11, 14, 16, 18, 18, ...	1
1	[[182, 180, 181, 181, 188, 184, 135, 71, 33, 1...	1
2	[[6, 6, 7, 7, 7, 6, 7, 7, 7, 7, 7, 7, 7, 6,...	1
3	[[191, 191, 192, 191, 193, 193, 195, 197, 197,...	1
4	[[144, 143, 148, 146, 120, 52, 21, 12, 5, 4, 2...	1

```
len(df)
y = df['label'].values
x = df['img'].values

dataset = np.array(dataset) / 255.0
img_labels = np.array(y)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

```
import tensorflow as tf
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D,MaxPool2D Flatten, Dense, Dropout

# Define the model architecture
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(256, 256, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))



x_train[0].shape
```

```
(1499, 1966)
```

```
model.compile(optimizer='adam',
              loss='binary_crossentropy',

              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=10)



# Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc)

# Make predictions
predictions = model.predict(new_data)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-16-50b81af510cc> in <cell line: 5>()
      3
      4               metrics=['accuracy'])
----> 5 model.fit(x_train, y_train, epochs=10)
      6
      7

                                 1 frames
/usr/local/lib/python3.10/dist-packages/tensorflow/python/framework/constant_op.py in convert_to_eager_tensor(value, ctx, dtype)
    101       dtype = dtypes.as_dtype(dtype).as_datatype_enum
    102   ctx.ensure_initialized()
--> 103   return ops.EagerTensor(value, ctx.device_name, dtype)
    104
    105

ValueError: Failed to convert a NumPy array to a Tensor (Unsupported object type numpy.ndarray).
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import os
import numpy as np
import pandas as pd
import random
import cv2
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Activation, Conv2D, MaxPooling2D,MaxPool2D ,Flatten, Dropout, BatchNormalization
import tensorflow as tf
```

```
data=[]
labels=[]
covid=os.listdir("/content/drive/MyDrive/LDL/covid")
for a in covid:
```

```
    try:
      image = cv2.imread("/content/drive/MyDrive/LDL/covid/"+a)
      image = cv2.resize(image, (224, 224))
      data.append(image)
      labels.append(0)
    except:
      continue
len(labels)
```

```
537
```

```
normal=os.listdir("/content/drive/MyDrive/LDL/normal")
for a in normal:
  try:
    image = cv2.imread("/content/drive/MyDrive/LDL/normal/"+a)
    image = cv2.resize(image, (224, 224))
    data.append(image)
    labels.append(1)
  except:
    continue
len(labels)
```

```
1098
```

```
data = np.array(data) / 255.0
img_labels = np.array(labels)
```

```
X_train, X_test, y_train, y_test = train_test_split(data, img_labels, test_size=0.25, random_state=20)
y_train = tf.keras.utils.to_categorical(y_train, num_classes=2)
y_test = tf.keras.utils.to_categorical(y_test, num_classes=2)
```

```
model1 = Sequential()
model1.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))

model1.add(Conv2D(32, (3, 3), activation='tanh'))
model1.add(MaxPooling2D((2, 2)))
model1.add(Conv2D(16, (3, 3), activation='tanh'))
model1.add(MaxPooling2D((2, 2)))
```

```
model1.add(Flatten())
model1.add(Dense(64, activation='relu'))

model1.add(Dense(2, activation='sigmoid'))
model1.compile(optimizer='RMSProp',loss='binary_crossentropy',metrics=['accuracy'])


# New
model = Sequential()
model.add(Conv2D(16, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(2, activation='sigmoid'))
model.compile(optimizer='RmsProp', loss='binary_crossentropy', metrics=['accuracy'])




model2 = Sequential()
model2.add(Conv2D(input_shape = (224,224,3), filters=3,padding="same", kernel_size= (2,2),activation='relu'))
model2.add(Conv2D(filters=32,padding='same', kernel_size= (2,2),activation='relu'))
model2.add(MaxPool2D((2,2)))
model2.add(Conv2D(filters=32,padding='same', kernel_size= (2,2),activation='relu'))
model2.add(MaxPool2D((2,2)))
model2.add(Flatten())
model2.add(Dense(256, activation="relu"))
model2.add(Dense(2, activation="softmax"))
model2.compile(optimizer='Adam',loss='binary_crossentropy',metrics=['accuracy'])


model1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 222, 222, 32)      896

 conv2d_3 (Conv2D)           (None, 220, 220, 32)      9248
```

```
 max_pooling2d_2 (MaxPooling  (None, 110, 110, 32)     0
 2D)

 conv2d_4 (Conv2D)           (None, 108, 108, 16)      4624

 max_pooling2d_3 (MaxPooling  (None, 54, 54, 16)       0
 2D)

 flatten_1 (Flatten)         (None, 46656)             0

 dense_2 (Dense)             (None, 64)                2986048

 dense_3 (Dense)             (None, 2)                 130

=================================================================
Total params: 3,000,946
Trainable params: 3,000,946
Non-trainable params: 0
_________________________________________________________________
```

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_5 (Conv2D)           (None, 222, 222, 16)      448

 max_pooling2d_4 (MaxPooling  (None, 111, 111, 16)     0
 2D)

 conv2d_6 (Conv2D)           (None, 109, 109, 32)      4640

 max_pooling2d_5 (MaxPooling  (None, 54, 54, 32)       0
 2D)

 flatten_2 (Flatten)         (None, 93312)             0

 dense_4 (Dense)             (None, 64)                5972032

 dense_5 (Dense)             (None, 2)                 130

=================================================================
```

```
Total params: 5,977,250
Trainable params: 5,977,250
Non-trainable params: 0
_________________________________________________________________
```

```
model2.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_17 (Conv2D)          (None, 224, 224, 3)       39

 conv2d_18 (Conv2D)          (None, 224, 224, 32)      416

 max_pooling2d_12 (MaxPoolin  (None, 112, 112, 32)     0
 g2D)

 conv2d_19 (Conv2D)          (None, 112, 112, 32)      4128

 max_pooling2d_13 (MaxPoolin  (None, 56, 56, 32)       0
 g2D)

 flatten_6 (Flatten)         (None, 100352)            0

 dense_12 (Dense)            (None, 256)               25690368

 dense_13 (Dense)            (None, 2)                 514

=================================================================
Total params: 25,695,465
Trainable params: 25,695,465
Non-trainable params: 0
_________________________________________________________________
```

```
his = model.fit(X_train, y_train, validation_split=0.1, epochs=5)
```

```
Epoch 1/5
24/24 [==============================] - 2s 44ms/step - loss: 0.7433 - accuracy: 0.7757 - val_loss: 0.1936 - val_accuracy: 0.9036
Epoch 2/5
24/24 [==============================] - 1s 31ms/step - loss: 0.2295 - accuracy: 0.9270 - val_loss: 0.1958 - val_accuracy: 0.9157
Epoch 3/5
24/24 [==============================] - 1s 32ms/step - loss: 0.1312 - accuracy: 0.9500 - val_loss: 2.9827 - val_accuracy: 0.4699
```

```
Epoch 4/5
24/24 [==============================] - 1s 31ms/step - loss: 0.2495 - accuracy: 0.9446 - val_loss: 0.0917 - val_accuracy: 0.9639
Epoch 5/5
24/24 [==============================] - 1s 31ms/step - loss: 0.0742 - accuracy: 0.9811 - val_loss: 0.1017 - val_accuracy: 0.9518
```

```
his1 = model1.fit(X_train, y_train, validation_split=0.3, epochs=10)
```

```
Epoch 1/10
18/18 [==============================] - 4s 129ms/step - loss: 1.7419 - accuracy: 0.6163 - val_loss: 0.5666 - val_accuracy: 0.8381
Epoch 2/10
18/18 [==============================] - 2s 91ms/step - loss: 0.6130 - accuracy: 0.6788 - val_loss: 0.4740 - val_accuracy: 0.8543
Epoch 3/10
18/18 [==============================] - 2s 90ms/step - loss: 0.5479 - accuracy: 0.8247 - val_loss: 0.1858 - val_accuracy: 0.9514
Epoch 4/10
18/18 [==============================] - 2s 91ms/step - loss: 0.2413 - accuracy: 0.9097 - val_loss: 0.3759 - val_accuracy: 0.8462
Epoch 5/10
18/18 [==============================] - 2s 84ms/step - loss: 0.3165 - accuracy: 0.8924 - val_loss: 0.1262 - val_accuracy: 0.9514
Epoch 6/10
18/18 [==============================] - 2s 91ms/step - loss: 0.1406 - accuracy: 0.9583 - val_loss: 0.1107 - val_accuracy: 0.9555
Epoch 7/10
18/18 [==============================] - 2s 85ms/step - loss: 0.1611 - accuracy: 0.9288 - val_loss: 0.1419 - val_accuracy: 0.9474
Epoch 8/10
18/18 [==============================] - 2s 91ms/step - loss: 0.2091 - accuracy: 0.9375 - val_loss: 0.3632 - val_accuracy: 0.9514
Epoch 9/10
18/18 [==============================] - 2s 90ms/step - loss: 0.1948 - accuracy: 0.9444 - val_loss: 0.1186 - val_accuracy: 0.9514
Epoch 10/10
18/18 [==============================] - 2s 93ms/step - loss: 0.1170 - accuracy: 0.9566 - val_loss: 0.0825 - val_accuracy: 0.9636
```

```
his2 = model2.fit(X_train, y_train, validation_split=0.3, epochs=10)
```

```
Epoch 1/10
18/18 [==============================] - 5s 87ms/step - loss: 0.4025 - accuracy: 0.8385 - val_loss: 0.1688 - val_accuracy: 0.9312
Epoch 2/10
18/18 [==============================] - 1s 51ms/step - loss: 0.1293 - accuracy: 0.9531 - val_loss: 0.1040 - val_accuracy: 0.9555
Epoch 3/10
18/18 [==============================] - 1s 53ms/step - loss: 0.0633 - accuracy: 0.9809 - val_loss: 0.0511 - val_accuracy: 0.9757
Epoch 4/10
18/18 [==============================] - 1s 52ms/step - loss: 0.0271 - accuracy: 0.9931 - val_loss: 0.0513 - val_accuracy: 0.9798
Epoch 5/10
18/18 [==============================] - 1s 56ms/step - loss: 0.0197 - accuracy: 0.9878 - val_loss: 0.0455 - val_accuracy: 0.9838
Epoch 6/10
18/18 [==============================] - 1s 60ms/step - loss: 0.0053 - accuracy: 1.0000 - val_loss: 0.0404 - val_accuracy: 0.9838
Epoch 7/10
```

```
18/18 [==============================] - 1s 66ms/step - loss: 0.0026 - accuracy: 1.0000 - val_loss: 0.0409 - val_accuracy: 0.9838
Epoch 8/10
18/18 [==============================] - 1s 67ms/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0476 - val_accuracy: 0.9838
Epoch 9/10
18/18 [==============================] - 1s 52ms/step - loss: 7.2236e-04 - accuracy: 1.0000 - val_loss: 0.0467 - val_accuracy: 0.9838
Epoch 10/10
18/18 [==============================] - 1s 53ms/step - loss: 5.5130e-04 - accuracy: 1.0000 - val_loss: 0.0561 - val_accuracy: 0.9838
```

```
score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:',score[0])
print('Test accuracy:',score[1])
```

```
Test loss: 0.055435944348573685
Test accuracy: 0.9745454788208008
```

```
score = model1.evaluate(X_test, y_test, verbose=0)
print('Test loss:',score[0])
print('Test accuracy:',score[1])
```

```
Test loss: 0.07828155159950256
Test accuracy: 0.9672726988792419
```

```
score = model2.evaluate(X_test, y_test, verbose=0)
print('Test loss:',score[0])
print('Test accuracy:',score[1])
```

```
Test loss: 0.027258126065135002
Test accuracy: 0.9927272796630859
```

```
image = cv2.imread("/content/drive/MyDrive/LDL/NORMAL/NORMAL_10.png")
```

```
image1 = cv2.resize(image, (224, 224))
```

```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
<ipython-input-70-366c87ed3a2e> in <cell line: 1>()
----> 1 image1 = cv2.resize(image, (224, 224,3))

error: OpenCV(4.7.0) :-1: error: (-5:Bad argument) in function 'resize'
```

```
image.shape
```

```
(232, 232, 3)
```

SEARCH STACK OVERFLOW

```
image1=image1/255
```

```
len(image1[5])
```

```
224
```

```
model.predict(np.array(image1))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-73-9e014516f095> in <cell line: 1>()
----> 1 model.predict(np.array(image1))

                                 1 frames
/usr/local/lib/python3.10/dist-packages/keras/engine/training.py in tf__predict_function(iterator)
     13                 try:
     14                     do_return = True
---> 15                     retval_ = ag__.converted_call(ag__.ld(step_function), (ag__.ld(self), ag__.ld(iterator)),
None, fscope)
     16                 except:
```

```
his.history.keys()
```

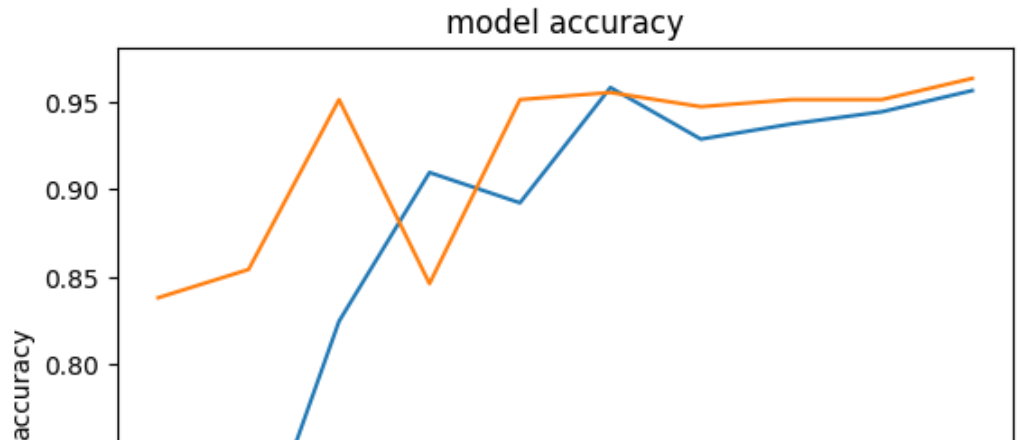
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
    File "/usr/local/lib/python3.10/dist-packages/keras/engine/training.py", line 2155, in step_function  **
```

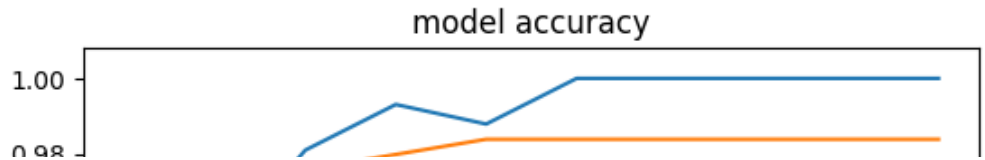
```
import matplotlib.pyplot as plt
```

```
    File "/usr/local/lib/python3.10/dist-packages/keras/engine/training.py", line 2111, in predict_step
```

```
plt.plot(his1.history['accuracy'])
plt.plot(his1.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='lower right')
plt.show()
```




```
plt.plot(his2.history['accuracy'])
plt.plot(his2.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='lower right')
plt.show()
```




```
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_12 (Conv2D)          (None, 222, 222, 16)      448

 max_pooling2d_10 (MaxPoolin  (None, 111, 111, 16)     0
 g2D)

 conv2d_13 (Conv2D)          (None, 109, 109, 32)      4640

 max_pooling2d_11 (MaxPoolin  (None, 54, 54, 32)       0
 g2D)

 flatten_5 (Flatten)         (None, 93312)             0

 dense_10 (Dense)            (None, 64)                5972032

 dense_11 (Dense)            (None, 2)                 130

=================================================================
Total params: 5,977,250
Trainable params: 5,977,250
Non-trainable params: 0
_________________________________________________________________
```

```
X_train[0].shape
```

```
(224, 224, 3)
```