```
#importing the required libraries
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt

#Import ImageDataGenerator Library.
train_datagon=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_r
ange=0.2,horizontal_flip=True)
test_datagon=ImageDataGenerator(rescale=1./255)

x_train=train_datagon.flow_from_directory('/content/drive/MyDrive/
dataset/
train_set',target_size=(64,64),batch_size=5,color_mode='rgb',class_mod
e='categorical')
x_test=test_datagon.flow_from_directory('/content/drive/MyDrive/datase
t/
test_set',target_size=(64,64),batch_size=5,color_mode='rgb',class_mode
='categorical')

Found 750 images belonging to 4 classes.
Found 198 images belonging to 4 classes.

from tensorflow.keras.layers import Dense,Flatten
from tensorflow.keras.layers import Conv2D,MaxPooling2D

#Initializing the model
model=Sequential()

# Adding CNN Layers
model.add(Conv2D(32,(3,3),input_shape=(64,64,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())

# Adding Dense Layers
model.add(Dense(units=128,activation='relu'))
model.add(Dense(units=4,activation='softmax'))


#summary of the model
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896
```

```
 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 128)               802944

 dense_1 (Dense)             (None, 4)                 516

=================================================================
Total params: 813,604
Trainable params: 813,604
Non-trainable params: 0
_________________________________________________________________
```

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

#fitting the model
model.fit_generator(generator=x_train,steps_per_epoch=len(x_train),epo
chs=20,validation_data=x_test,validation_steps=len(x_test))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
UserWarning: `Model.fit_generator` is deprecated and will be removed
in a future version. Please use `Model.fit`, which supports
generators.
  """Entry point for launching an IPython kernel.

Epoch 1/20
150/150 [==============================] - 613s 4s/step - loss: 1.2105
- accuracy: 0.4640 - val_loss: 1.0827 - val_accuracy: 0.5354
Epoch 2/20
150/150 [==============================] - 32s 208ms/step - loss:
0.9050 - accuracy: 0.6373 - val_loss: 0.8418 - val_accuracy: 0.6566
Epoch 3/20
150/150 [==============================] - 32s 213ms/step - loss:
0.6965 - accuracy: 0.7467 - val_loss: 0.8132 - val_accuracy: 0.6768
Epoch 4/20
150/150 [==============================] - 32s 214ms/step - loss:
0.5961 - accuracy: 0.7627 - val_loss: 0.8859 - val_accuracy: 0.6717
Epoch 5/20
150/150 [==============================] - 33s 222ms/step - loss:
0.6220 - accuracy: 0.7520 - val_loss: 0.5450 - val_accuracy: 0.7929
Epoch 6/20
```

```
150/150 [==============================] - 32s 213ms/step - loss:
0.5501 - accuracy: 0.7960 - val_loss: 0.5818 - val_accuracy: 0.7879
Epoch 7/20
150/150 [==============================] - 32s 212ms/step - loss:
0.4719 - accuracy: 0.8320 - val_loss: 0.6466 - val_accuracy: 0.8081
Epoch 8/20
150/150 [==============================] - 32s 214ms/step - loss:
0.4331 - accuracy: 0.8427 - val_loss: 0.8055 - val_accuracy: 0.7121
Epoch 9/20
150/150 [==============================] - 32s 208ms/step - loss:
0.4473 - accuracy: 0.8360 - val_loss: 0.5916 - val_accuracy: 0.8030
Epoch 10/20
150/150 [==============================] - 33s 220ms/step - loss:
0.4006 - accuracy: 0.8453 - val_loss: 0.9656 - val_accuracy: 0.7475
Epoch 11/20
150/150 [==============================] - 32s 212ms/step - loss:
0.4163 - accuracy: 0.8533 - val_loss: 0.6551 - val_accuracy: 0.7929
Epoch 12/20
150/150 [==============================] - 33s 223ms/step - loss:
0.3447 - accuracy: 0.8840 - val_loss: 1.0778 - val_accuracy: 0.7323
Epoch 13/20
150/150 [==============================] - 32s 212ms/step - loss:
0.3264 - accuracy: 0.8760 - val_loss: 0.9580 - val_accuracy: 0.7374
Epoch 14/20
150/150 [==============================] - 32s 211ms/step - loss:
0.3002 - accuracy: 0.8933 - val_loss: 0.8860 - val_accuracy: 0.7677
Epoch 15/20
150/150 [==============================] - 32s 212ms/step - loss:
0.2955 - accuracy: 0.8947 - val_loss: 0.8513 - val_accuracy: 0.7626
Epoch 16/20
150/150 [==============================] - 32s 213ms/step - loss:
0.2750 - accuracy: 0.8947 - val_loss: 0.8177 - val_accuracy: 0.7424
Epoch 17/20
150/150 [==============================] - 32s 210ms/step - loss:
0.2751 - accuracy: 0.8973 - val_loss: 0.8954 - val_accuracy: 0.7273
Epoch 18/20
150/150 [==============================] - 32s 212ms/step - loss:
0.2111 - accuracy: 0.9267 - val_loss: 0.7851 - val_accuracy: 0.8131
Epoch 19/20
150/150 [==============================] - 32s 214ms/step - loss:
0.1918 - accuracy: 0.9267 - val_loss: 0.7158 - val_accuracy: 0.8283
Epoch 20/20
150/150 [==============================] - 33s 223ms/step - loss:
0.1934 - accuracy: 0.9280 - val_loss: 0.7524 - val_accuracy: 0.7879

<keras.callbacks.History at 0x7fe34015d950>
```

```
#saving the model
model.save('disaster.h5')
```

```
model_json=model.to_json()
with open("model-bw.json","w")as json_file:
    json_file.write(model_json)

#Test the model
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
model=load_model("disaster.h5")

img=image.load_img('/content/drive/MyDrive/dataset/test_set/
Earthquake/1321.jpg',target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
pred=model.predict(x)
np.argmax(pred)
pred

1/1 [==============================] - 0s 16ms/step

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

index=['Cyclone','Earthquake','Flood','Wildfire']
y=np.argmax(model.predict(x),axis=1)
print(index[int(y)])

1/1 [==============================] - 0s 17ms/step
Earthquake
```