```
import warnings
warnings.filterwarnings("ignore")

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from tensorflow.keras.models import Sequential,Model
from tensorflow.keras.layers import
Dense,Activation,Dropout,Conv2D,Flatten,MaxPool2D,Reshape,GlobalAveragePooling2D,Input
Layer
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import
ImageDataGenerator,load_img,img_to_array
from tensorflow.keras.callbacks import EarlyStopping,ReduceLROnPlateau

IMAGE_SIZE=[229,229]

train_path='../input/natural-disaster-intensity/dataset/train_set'
test_path='../input/natural-disaster-intensity/dataset/test_set'

train_data_gen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizo
ntal_flip=True,validation_split=0.30)
test_data_gen=ImageDataGenerator(rescale=1./255,validation_split=0.30)

training_set=train_data_gen.flow_from_directory(train_path,target_size=(229,229),batch
_size=100,class_mode='categorical',shuffle=True,color_mode='rgb',subset='training')
testing_set=test_data_gen.flow_from_directory(test_path,target_size=(229,229),batch_si
ze=100,class_mode='categorical',shuffle=True,color_mode='rgb',subset='validation')

Found 521 images belonging to 4 classes.
Found 58 images belonging to 4 classes.

model=Sequential()

model.add(Conv2D(32,(3,3),input_shape=(229,229,3),activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Conv2D(32,(3,3),activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(units=128,activation='relu'))
model.add(Dense(units=4,activation='softmax'))
model.summary()
Model: "sequential_18"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 227, 227, 32)      896
_________________________________________________________________
```

```
max_pooling2d_2 (MaxPooling2 (None, 113, 113, 32)      0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 111, 111, 32)      9248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 55, 55, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 96800)             0
_________________________________________________________________
dense (Dense)                (None, 128)               12390528
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 516
=================================================================
Total params: 12,401,188
Trainable params: 12,401,188
Non-trainable params: 0
_________________________________________________________________
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
model.fit_generator(
    generator=training_set,steps_per_epoch=len(training_set),
    epochs=20,validation_data=testing_set,validation_steps=len(testing_set))

Epoch 1/20
6/6 [==============================] - 40s 6s/step - loss: 7.4400 - accuracy: 0.2745 -
val_loss: 4.7555 - val_accuracy: 0.1379
Epoch 2/20
6/6 [==============================] - 30s 6s/step - loss: 1.6385 - accuracy: 0.3186 -
val_loss: 1.4745 - val_accuracy: 0.3103
Epoch 3/20
6/6 [==============================] - 30s 5s/step - loss: 1.2937 - accuracy: 0.4127 -
val_loss: 1.3408 - val_accuracy: 0.4138
Epoch 4/20
6/6 [==============================] - 31s 5s/step - loss: 1.2657 - accuracy: 0.4088 -
val_loss: 1.2716 - val_accuracy: 0.6034
Epoch 5/20
6/6 [==============================] - 31s 5s/step - loss: 1.1531 - accuracy: 0.5125 -
val_loss: 1.2474 - val_accuracy: 0.4138
Epoch 6/20
6/6 [==============================] - 31s 5s/step - loss: 1.0124 - accuracy: 0.5336 -
val_loss: 1.1508 - val_accuracy: 0.5862
Epoch 7/20
6/6 [==============================] - 31s 5s/step - loss: 0.8680 - accuracy: 0.6468 -
val_loss: 1.0622 - val_accuracy: 0.6034
Epoch 8/20
6/6 [==============================] - 30s 6s/step - loss: 0.8234 - accuracy: 0.6449 -
val_loss: 0.9700 - val_accuracy: 0.5862
Epoch 9/20
6/6 [==============================] - 30s 5s/step - loss: 0.7506 - accuracy: 0.7179 -
val_loss: 0.8130 - val_accuracy: 0.7586
Epoch 10/20
```

```
6/6 [==============================] - 31s 5s/step - loss: 0.7150 - accuracy: 0.7351 -
val_loss: 0.9505 - val_accuracy: 0.7241
Epoch 11/20
6/6 [==============================] - 30s 5s/step - loss: 0.7464 - accuracy: 0.7102 -
val_loss: 1.0083 - val_accuracy: 0.5517
Epoch 12/20
6/6 [==============================] - 30s 5s/step - loss: 0.6719 - accuracy: 0.7562 -
val_loss: 0.7746 - val_accuracy: 0.7586
Epoch 13/20
6/6 [==============================] - 30s 6s/step - loss: 0.5585 - accuracy: 0.7946 -
val_loss: 0.9316 - val_accuracy: 0.7414
Epoch 14/20
6/6 [==============================] - 30s 5s/step - loss: 0.5536 - accuracy: 0.7774 -
val_loss: 1.1241 - val_accuracy: 0.6552
Epoch 15/20
6/6 [==============================] - 31s 6s/step - loss: 0.5133 - accuracy: 0.8157 -
val_loss: 1.0897 - val_accuracy: 0.6897
Epoch 16/20
6/6 [==============================] - 33s 6s/step - loss: 0.5082 - accuracy: 0.8081 -
val_loss: 0.9379 - val_accuracy: 0.7069
Epoch 17/20
6/6 [==============================] - 33s 6s/step - loss: 0.4945 - accuracy: 0.8157 -
val_loss: 1.0962 - val_accuracy: 0.6379
Epoch 18/20
6/6 [==============================] - 34s 6s/step - loss: 0.5233 - accuracy: 0.7927 -
val_loss: 1.2129 - val_accuracy: 0.5862
Epoch 19/20
6/6 [==============================] - 33s 6s/step - loss: 0.4238 - accuracy: 0.8196 -
val_loss: 1.0063 - val_accuracy: 0.7241
Epoch 20/20
6/6 [==============================] - 33s 5s/step - loss: 0.3634 - accuracy: 0.8676 -
val_loss: 1.0335 - val_accuracy: 0.7586
model.save("nature.h5")
from tensorflow.keras.models import load_model
from keras.preprocessing import image
model = load_model("nature.h5")
a = ['Cyclone','Earthquake','Flood','Wildfire']
img=image.load_img('../input/natural-disaster-
intensity/dataset/test_set/Cyclone/876.jpg',
                   target_size=(229,229))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
pred=np.argmax(model.predict(x))
a[pred]
'Cyclone'
testing_set.class_indices
{'Cyclone': 0, 'Earthquake': 1, 'Flood': 2, 'Wildfire': 3}
```