```
from google.colab import drive

drive.mount('/content/drive')

Mounted at /content/drive

ls

drive/  sample_data/

cd /content/drive/MyDrive/dataset

/content/drive/MyDrive/dataset

ls

readme.txt  test_set/  train_set/

pwd

'/content/drive/MyDrive/dataset'

# importing imagedatagenerator library


from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Image Data Agumentation

```
#Configuring image Data Generator Class

#Setting Parameter for Image Augmentation for training data


train_datagen=ImageDataGenerator(rescale=1./255,horizontal_flip=True,vertical_flip=True,
zoom_range=0.2)

#Image Data Augmentation for testing data


test_datagen= ImageDataGenerator(rescale=1./255)
```

Apply Image Data Generator Functionality To Train And Test Database

```
#Performing data augmentation to train data


x_train = train_datagen.flow_from_directory('/content/drive/MyDrive/dataset/train_set', target_size =
(64,64), batch_size = 5, color_mode = 'rgb', class_mode = 'categorical')
```

Found 744 images belonging to 4 classes.

#performing data augmentation to test data

x_test = test_datagen.flow_from_directory('/content/drive/MyDrive/dataset/test_set', target_size = (64,64), batch_size = 5, color_mode = 'rgb', class_mode = 'categorical')

Found 198 images belonging to 4 classes.

#importing neccessary libraries

import numpy as np

import tensorflow

from tensorflow.keras.models import Sequential

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

initialising the model and adding CNN layers

model = Sequential()

First convolution layer and pooling

model.add(Conv2D(32,(3,3),input_shape=(64,64,3),activation='relu'))

model.add(MaxPooling2D(pool_size=(2,2)))

#Second convolution layer and pooling

model.add(Conv2D(32,(3,3),activation='relu'))

model.add(MaxPooling2D(pool_size=(2,2)))

#Flattening the layers

model.add(Flatten())

#Adding Dense Layers

```
model.add(Dense(units=128,activation='relu'))
model.add(Dense(units=4,activation='softmax'))
# Summary of our model

model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_2 (MaxPooling 2D)	(None, 31, 31, 32)	0
conv2d_3 (Conv2D)	(None, 29, 29, 32)	9248
max_pooling2d_3 (MaxPooling 2D)	(None, 14, 14, 32)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_2 (Dense)	(None, 128)	802944
dense_3 (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),epochs=20,validation_data=x_test,validation_steps=len(x_test))

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

This is separate from the ipykernel package so we can avoid doing imports until

Epoch 1/20

149/149 [==============================] - 235s 2s/step - loss: 1.2298 - accuracy: 0.4355 - val_loss: 1.2029 - val_accuracy: 0.4343

Epoch 2/20

149/149 [==============================] - 28s 187ms/step - loss: 0.9725 - accuracy: 0.5860 - val_loss: 1.0199 - val_accuracy: 0.5657

Epoch 3/20

149/149 [==============================] - 28s 185ms/step - loss: 0.7891 - accuracy: 0.6909 - val_loss: 0.9436 - val_accuracy: 0.5556

Epoch 4/20

149/149 [==============================] - 27s 183ms/step - loss: 0.7616 - accuracy: 0.6586 - val_loss: 0.8652 - val_accuracy: 0.6061

Epoch 5/20

149/149 [==============================] - 27s 183ms/step - loss: 0.6975 - accuracy: 0.7097 - val_loss: 0.7742 - val_accuracy: 0.6717

Epoch 6/20

149/149 [==============================] - 29s 194ms/step - loss: 0.6322 - accuracy: 0.7325 - val_loss: 0.6910 - val_accuracy: 0.7020

Epoch 7/20

149/149 [==============================] - 28s 186ms/step - loss: 0.6149 - accuracy: 0.7500 - val_loss: 1.0359 - val_accuracy: 0.6162

Epoch 8/20

149/149 [==============================] - 27s 183ms/step - loss: 0.5845 - accuracy: 0.7715 - val_loss: 0.7130 - val_accuracy: 0.7273

Epoch 9/20

149/149 [==============================] - 27s 185ms/step - loss: 0.5708 - accuracy: 0.7608 - val_loss: 0.9826 - val_accuracy: 0.6515

Epoch 10/20

149/149 [==============================] - 29s 194ms/step - loss: 0.5512 - accuracy: 0.7890 - val_loss: 0.8693 - val_accuracy: 0.6717

Epoch 11/20

149/149 [==============================] - 27s 184ms/step - loss: 0.5503 - accuracy: 0.7836 - val_loss: 0.7070 - val_accuracy: 0.7374

Epoch 12/20

149/149 [==============================] - 27s 184ms/step - loss: 0.4977 - accuracy: 0.8011 - val_loss: 0.6304 - val_accuracy: 0.7929

Epoch 13/20

149/149 [==============================] - 27s 184ms/step - loss: 0.4205 - accuracy: 0.8441 - val_loss: 0.6953 - val_accuracy: 0.7626

Epoch 14/20

149/149 [==============================] - 27s 184ms/step - loss: 0.4129 - accuracy: 0.8454 - val_loss: 0.7047 - val_accuracy: 0.7576

Epoch 15/20

149/149 [==============================] - 29s 190ms/step - loss: 0.4337 - accuracy: 0.8387 - val_loss: 0.5759 - val_accuracy: 0.8434

Epoch 16/20

149/149 [==============================] - 27s 184ms/step - loss: 0.3875 - accuracy: 0.8602 - val_loss: 0.6379 - val_accuracy: 0.7677

Epoch 17/20

149/149 [==============================] - 27s 184ms/step - loss: 0.3608 - accuracy: 0.8817 - val_loss: 0.7366 - val_accuracy: 0.7778

Epoch 18/20

149/149 [==============================] - 30s 199ms/step - loss: 0.3530 - accuracy: 0.8589 - val_loss: 0.8593 - val_accuracy: 0.7323

Epoch 19/20

149/149 [==============================] - 27s 184ms/step - loss: 0.3634 - accuracy: 0.8602 - val_loss: 0.7928 - val_accuracy: 0.7475

Epoch 20/20

149/149 [==============================] - 27s 184ms/step - loss: 0.2910 - accuracy: 0.8952 - val_loss: 0.7734 - val_accuracy: 0.8030

Save the model

```
model.save('naturaldisaster.h5')

model_json = model.to_json()

with open("model-bw.json", "w") as json_file:

  json_file.write(model_json)
```

Load the saved model

```
from tensorflow.keras.models import load_model

from tensorflow.keras.preprocessing import image

model = load_model('naturaldisaster.h5')

x_train.class_indices
```

{'Cyclone': 0, 'Earthquake': 1, 'Flood': 2, 'Wildfire': 3}

taking image as input

```
img =
image.load_img('/content/drive/MyDrive/dataset/test_set/Earthquake/1333.jpg',target_size=(64,64))

x=image.img_to_array(img)

x=np.expand_dims(x,axis=0)

index=['Cyclone','Earthquake','Flood','Wildfire']

y=np.argmax(model.predict(x),axis=1)
```

```
print(index[int(y)])
1/1 [==============================] - 0s 93ms/step
Earthquake
# input 2

img = image.load_img('/content/drive/MyDrive/dataset/test_set/Flood/1009.jpg',target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
index=['Cyclone','Earthquake','Flood','Wildfire']
y=np.argmax(model.predict(x),axis=1)
print(index[int(y)])
1/1 [==============================] - 0s 16ms/step
Flood
# input 3

img =
image.load_img('/content/drive/MyDrive/dataset/test_set/Wildfire/1065.jpg',target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
index=['Cyclone','Earthquake','Flood','Wildfire']
y=np.argmax(model.predict(x),axis=1)
print(index[int(y)])
1/1 [==============================] - 0s 14ms/step
Wildfire
# input 4

img = image.load_img('/content/drive/MyDrive/dataset/test_set/Cyclone/903.jpg',target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
```

```
index=['Cyclone','Earthquake','Flood','Wildfire']
y=np.argmax(model.predict(x),axis=1)
print(index[int(y)])
1/1 [==============================] - 0s 14ms/step
Cyclone
```