

EARLY DETECTION OF FOREST FIRE USING DEEP LEARNING

MODEL BUILDING

ADDING CNN LAYERS

Team ID	PNT2022TMID23181
Project Name	Project-Early detection of forest fire using deep learning

ADDING CNN LAYERS:

We will be adding three layers for CNN

- Convolution layer
- Pooling layer
- Flattening layer

Adding Convolutional Layer:

The convolutional layer is the first and core layer of CNN. It is one of the building blocks of a CNN and is used for extracting important features from the image.

In the Convolution operation, the input image will be convolved with the feature detector/filters to get a feature map. The important role of the feature detector is to extract the features from the image. The group of feature maps is called a feature layer.

In the convolution2D function, we gave arguments that include 32,(3,3), that refers to we are applying 32 filters of 3x3 matrix filter, and input_shape is the input image shape with RGB, here 64x64 is the size and 3 represent the channel, RGB colour images.

Activation Function: These are the functions that help us to decide if we need to activate the node or not. These functions introduce non-linearity in the networks.

Adding Pooling Layer

Max Pooling selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

After the convolution layer, a pooling layer is added. Max pooling layer can be added using MaxPooling2D class. It takes the pool size as a parameter. Efficient size of the pooling matrix is (2,2). It returns the pooled feature maps. (Note: Any number of convolution layers, pooling and dropout layers can be added)

In the above code, pool_size refers to pooling filter or kernel size.

Task 3: Adding Flatten Layer

Now the pooled feature map from the pooling layer will be converted into one single dimension matrix or map, where each pixel in one single column, nothing but flattening. The flattening layer converts the multi-dimension matrix to one single dimension layer.

IMPORT LIBRARIES:

11/7/22, 12:35 AM

Untitled8.ipynb - Colaboratory

▾ Importing Keras libraries

```
import keras
```

▾ Importing ImageDataGenerator from Keras

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

IMPORT ImageDataGenerator FROM KERAS:

▾ Importing Keras libraries

```
✓ [1] import keras
```

▾ Importing ImageDataGenerator from Keras

```
✓ [13] from matplotlib import pyplot as plt  
      from keras.preprocessing.image import ImageDataGenerator
```

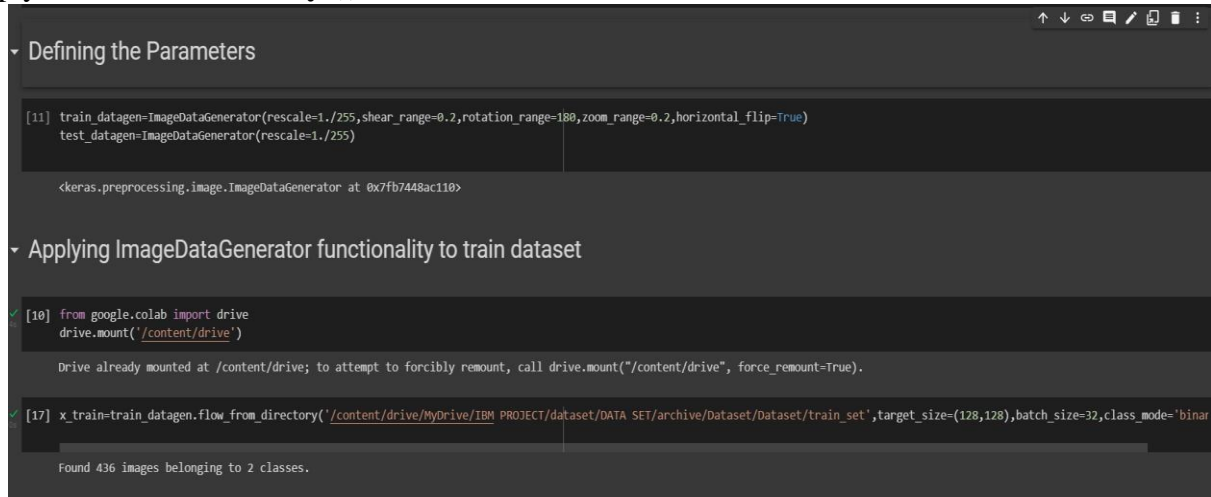
▾ Defining the Parameters

```
▶ train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, rotation_range=180, zoom_range=0.2, horizontal_flip=True)  
  test_datagen=ImageDataGenerator(rescale=1./255)
```

```
□ <keras.preprocessing.image.ImageDataGenerator at 0x7fb7448ac110>
```

APPLYING ImageDataGenerator to train dataset:

ply **flow_from_directory ()** method for Train folder.



The screenshot shows a Jupyter Notebook with two sections. The first section, 'Defining the Parameters', contains code to create ImageDataGenerator objects for training and testing. The second section, 'Applying ImageDataGenerator functionality to train dataset', shows the mounting of Google Drive and the use of flow_from_directory to load training data.

```
[11] train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, rotation_range=180, zoom_range=0.2, horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)

<keras.preprocessing.image.ImageDataGenerator at 0x7fb7448ac110>
```

Defining the Parameters

```
[10] 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).
```

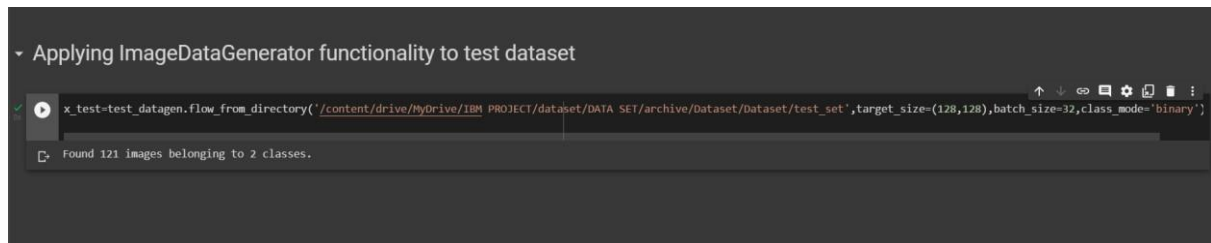
```
[17] x_train=train_datagen.flow_from_directory('/content/drive/MyDrive/IBM PROJECT/dataset/DATA SET/archive/Dataset/Dataset/train_set', target_size=(128,128), batch_size=32, class_mode='binary')

Found 436 images belonging to 2 classes.
```

Applying ImageDataGenerator functionality to train dataset

APPLYING ImageDataGenerator to test dataset:

Applying the **flow_from_directory ()** method for test folder.



The screenshot shows a Jupyter Notebook with one section, 'Applying ImageDataGenerator functionality to test dataset'. It contains code to use flow_from_directory to load testing data from Google Drive.

```
x_test=test_datagen.flow_from_directory('/content/drive/MyDrive/IBM PROJECT/dataset/DATA SET/archive/Dataset/Dataset/test_set', target_size=(128,128), batch_size=32, class_mode='binary')
```

Applying ImageDataGenerator functionality to test dataset

```
Found 121 images belonging to 2 classes.
```

IMPORTING MODEL BUILDING LIBRARIES:

▼ Importing Model Building Libraries

```
#to define the linear Initialisation import sequential
from keras.models import Sequential
#to add layers import Dense
from keras.layers import Dense
#to create Convolutional kernel import convolution2D
from keras.layers import Convolution2D
#import Maxpooling layer
from keras.layers import MaxPooling2D
#import flatten layer
from keras.layers import Flatten
import warnings
warnings.filterwarnings('ignore')
```

INITIALIZING THE MODEL:

▼ Initializing the model

```
model=Sequential()
```

ADDING CNN LAYERS:

▼ Adding CNN Layers

```
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
#add maxpooling layers
model.add(MaxPooling2D(pool_size=(2,2)))
#add faltten layer
model.add(Flatten())
```