EMERGING METHODS FOR EARLY DETECTION OF FORESTFIRES

MODEL

BUILDINGTRAINING

THEMODEL

TeamID	PNT2022TMID30362
ProjectName	Project-Emerging methods for Early detection of forest fires

TRAININGTHEMODEL:

At this point, we have training data and a fully configured neural network to train. All that isleft is to pass the data to the model for the training process to commence, a process that iscompletedby iterating the training data. Training begins by calling the fit () method.

The arguments are the batch size as you are using "adam" (bath gradient descent and epochs:no:of times themodel should get trained).

steps_per_epoch:

- It specifies the total number of steps taken from the generator as soon as one epoch isfinished and then extep och has started. We can calculate the value of steps_per_epoch as the total number of samples in your training folder divided by the batch size.
- Epochs:anintegerandnumberofepochs wewanttotrain ourmodel for.
- Validation_datacanbeeitherinputandtargetslistgeneratorinputs,targets,andsample_weig htslist which can be used to evaluate.

Theloss andmetrics for any modelafter anyepoch hasended.

Validation_steps:

Only if the validation_data is a generator then only this argument can be used. It specifies thetotalnumberofstepstakenfromthegeneratorbeforeitisstoppedateveryepochanditsvalueis calculated as the total number of validation data points in your dataset divided by the validation batch size.

IMPORTLIBRARIES:

11/7/22, 12:35 AM

Untitled8.ipynb - Colaboratory

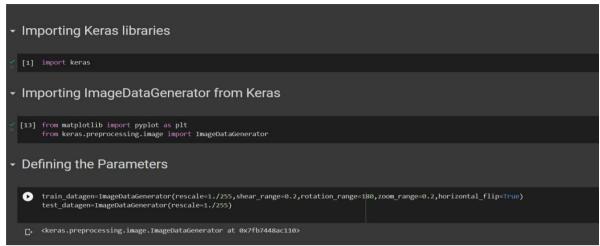
Importing Keras libraries

import keras

Importing ImageDataGenerator from Keras

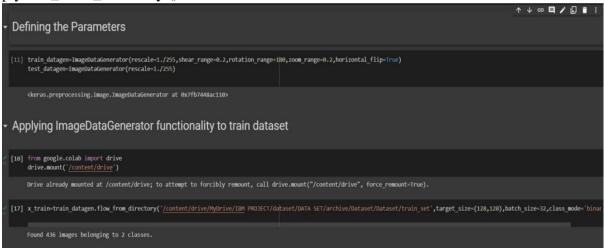
from keras.preprocessing.image import ImageDataGenerator

IMPORTImageDataGenerator FROMKERAS:



APPLYINGImageDataGeneratortotraindataset:

plyflow_from_directory ()methodforTrainfolder.



APPLYINGImageDataGeneratortotestdataset:

Applyingtheflow_from_directory()methodfortestfolder.



IMPORTINGMODELBUILDINGLIBRARIES:

11/8/22, 1:16 AM

Main code - Colaboratory

→ 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')
```

INITIALIZINGTHEMODEL:

Initializing the model

```
model=Sequential()
```

ADDINGCNNLAYERS:

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())
```

ADDINGDENSELAYERS:

Add Dense layers

```
#add hidden layers
model.add(Dense(150,activation='relu'))
#add output layer
model.add(Dense(1,activation='sigmoid'))
```

CONFIGURINGTHELEARNING PROCESS:

configuring the learning process

```
model.compile(loss='binary_crossentropy',optimizer="adam",metrics=["accuracy"])
```

TRAININGTHEMODEL:

Training the model

```
\verb|model.fit_generator(x_train, steps_per_epoch=14, epochs=10, validation_data=x_test, validation_dat
           Epoch 1/10
           Epoch 2/10
           Epoch 3/10
           14/14 [==============] - 32s 2s/step - loss: 0.2979 - accuracy: 0.885
           Epoch 4/10
           Epoch 5/10
           Epoch 6/10
           14/14 [============== ] - 30s 2s/step - loss: 0.1971 - accuracy: 0.926
           Epoch 7/10
           14/14 [=============] - 32s 2s/step - loss: 0.1781 - accuracy: 0.928
           Epoch 8/10
           Epoch 9/10
           14/14 [============== ] - 31s 2s/step - loss: 0.2306 - accuracy: 0.896
           Epoch 10/10
           14/14 [==============] - 27s 2s/step - loss: 0.2593 - accuracy: 0.889
           <keras.callbacks.History at 0x7fd537101390>
          4
```