

Model Building

Train The Model

Date: 08 November 2022

Team ID: PNT2022TMID50335

Project Name : Emerging Methods for early Detection Of forest fire

```
'''import model building libraries'''
```

```
#to define linear initialisation import sequential
```

```
from keras.models import Sequential
```

```
#to add layers import Dense
```

```
from keras.layers import Dense
```

```
#to create Convolution kernel import Covolution2D
```

```
from keras.layers import Convolution2D
```

```
#import Maxpooling Layer
```

```
from keras.layers import Dense
```

```
from keras.layers import Convolution2D
```

```
#import Flatten Layer
```

```
from keras.layers import Flatten
```

```
#import maxpooling layer
```

```
from keras.layers import MaxPooling2D
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

```
#initializing model
```

```
model=Sequential()
```

Adding CNN Layer

Task 1:

```
#add cnn layer
```

```
model.add(Conv2D(filters=32,kernel_size=2,padding="same",activation  
="relu"))
```

```
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=64, kernel_size=2, padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=2))
```

Task 2:

```
#flattening layer
model.add(Flatten())
```

Task 3:

```
#Adding PoolingLayer
```

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

Adding Dense Layer

```
#adding DenseLayer
```

```
model.add(Dense(500, activation="relu"))
```

```
model.add(Dense(2, activation="softmax"))
```

```
print("Adding dense layer on top")
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
print("Complete architecture of the model")
model.summary()
```

Configuring the Learning Process

```
#Configuring the learning process
```

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

Test The Model

```
#importing the required libraries
```

```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPool2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Dense
```

```
#loading data
```

```
(X_train,y_train) , (X_test,y_test)=mnist.load_data()
```

```
#reshaping data
```

```

X_train = X_train.reshape((X_train.shape[0], X_train.shape[1],
X_train.shape[2], 1))
X_test =
X_test.reshape((X_test.shape[0],X_test.shape[1],X_test.shape[2],1))
#checking the shape after reshaping
print(X_train.shape)
print(X_test.shape)
#normalizing the pixel values
X_train=X_train/255
X_test=X_test/255

```

```

Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/mnist.npz
11490434/11490434 [=====] - 0s 0us/step
(60000, 28, 28, 1)
(10000, 28, 28, 1)

```

```

#defining model
model=Sequential()
#adding convolution layer
model.add(Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)))
#adding pooling layer
model.add(MaxPool2D(2,2))
#adding fully connected layer
model.add(Flatten())
model.add(Dense(100,activation='relu'))
#adding output layer
model.add(Dense(10,activation='softmax'))
#compiling the model
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',
metrics=['accuracy'])
#fitting the model
model.fit(X_train,y_train,epochs=10)

```

```

Epoch 1/10
1875/1875 [=====] - 41s 21ms/step - loss:
0.1474 - accuracy: 0.9566
Epoch 2/10
1875/1875 [=====] - 40s 21ms/step - loss:
0.0534 - accuracy: 0.9837
Epoch 3/10
1875/1875 [=====] - 39s 21ms/step - loss:
0.0338 - accuracy: 0.9893
Epoch 4/10
1875/1875 [=====] - 41s 22ms/step - loss:
0.0218 - accuracy: 0.9931
Epoch 5/10
1875/1875 [=====] - 39s 21ms/step - loss:
0.0149 - accuracy: 0.9952
Epoch 6/10
1875/1875 [=====] - 41s 22ms/step - loss:

```

0.0101 - accuracy: 0.9966

Epoch 7/10

1875/1875 [=====] - 39s 21ms/step - loss:

0.0084 - accuracy: 0.9973

Epoch 8/10

373/1875 [====>.....] - ETA: 35s - loss: 0.0070 -

accuracy: 0.9976