SPRINT 2

Team ID : PNT2022TMID43250

Project Name : Natural Disaster Intensity Analysis and Classification Using Artificial Intelligence

FROM SPRINT 1

```
import numpy as np
import tensorflow #open source used for both ML and DL for computation
from tensorflow.keras.datasets import disaster #disaster dataset
from tensorflow.keras.models import Sequential #it is a plain stack of layers
from tensorflow.keras import layers #A Layer consists of a tensor- in
tensor-out computation function
from tensorflow.keras.layers import Dense, Flatten #Dense-Dense Layer is the
regular deeply connected
#flatten-used for flattening the input or change the dimension
from tensorflow.keras.layers import Conv2D #convoLutional Layer
from keras.optimizers import Adam #optimizer
from keras. utils import np_utils #used for one-hot encoding
import matplotlib.pyplot as plt   #used for data visualization
```

```
(x_train, y_train), (x_test, y_test)=disaster.load_data() #splitting the
disaster data
```

```
#Reshaping to format which CNN expects (batch, height, width, channels)
x_train=x_train.reshape (60000, 28, 28, 1).astype('float32')
x_test=x_test.reshape (10000, 28, 28, 1).astype ('float32')
```

```
#one hot encode
number_of_classes = 10  #storing the no of classes in a variable
y_train = np_utils.to_categorical (y_train, number_of_classes) #converts the
output in binary format
y_test = np_utils.to_categorical (y_test, number_of_classes)
```

```
Downloading data from
https://storage.googleapis.com/tensorflow/tf-keras-datasets/disaster.npz
11490434/11490434 [==============================] - 1s 0us/step
```

ADD CNN LAYERS

```
#create model
model=Sequential ()


#adding model Layer
model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation='relu'))
model.add(Conv2D(32, (3, 3), activation = 'relu'))
#model.add(conv2D(32, (3,3), activation = 'relu))


#flatten the dimension of the image
model.add(Flatten())


#output layer with 10 neurons
model.add(Dense(number_of_classes,activation = 'softmax'))
```

COMPILING THE MODEL

```
#Compile model
model.compile(loss= 'categorical_crossentropy', optimizer="Adam",
metrics=['accuracy'])
```

TRAIN THE MODEL

```
#fit the model
model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=5,
batch_size=32)
```

```
Epoch 1/5
1875/1875 [==============================] - 194s 103ms/step - loss: 0.1173 -
accuracy: 0.9662 - val_loss: 0.0771 - val_accuracy: 0.9777
```

```
Epoch 2/5
1875/1875 [==============================] - 195s 104ms/step - loss: 0.0644 -
accuracy: 0.9801 - val_loss: 0.0795 - val_accuracy: 0.9777
Epoch 3/5
1875/1875 [==============================] - 197s 105ms/step - loss: 0.0441 -
accuracy: 0.9863 - val_loss: 0.1046 - val_accuracy: 0.9759
Epoch 4/5
1875/1875 [==============================] - 205s 110ms/step - loss: 0.0351 -
accuracy: 0.9887 - val_loss: 0.0871 - val_accuracy: 0.9782
Epoch 5/5
1875/1875 [==============================] - 207s 110ms/step - loss: 0.0284 -
accuracy: 0.9909 - val_loss: 0.1242 - val_accuracy: 0.9762

<keras.callbacks.History at 0x7fab39e6e9d0>
```

OBSERVING THE METRICS

```
# Final evaluation of the model
metrics = model.evaluate(x_test, y_test, verbose=0)
print("Metrics (Test loss & Test Accuracy): ")
print(metrics)
```

```
Metrics (Test loss & Test Accuracy):
[0.12423925846815109, 0.9761999845504761]
```

TEST THE MODEL

```
# Predicting the Output
prediction=model.predict(x_test[:4])
print(prediction)
```

```
1/1 [==============================] - 0s 85ms/step
[[6.25729828e-13 8.39843610e-19 3.52224987e-07 3.49750486e-08
  2.61816901e-21 4.89236403e-17 6.80994400e-23 9.99999642e-01
  1.00192285e-10 1.46840540e-09]
 [2.13350471e-09 7.24474439e-11 1.00000000e+00 1.42506189e-12
  1.07695855e-18 2.63603979e-20 9.05597333e-11 3.16722711e-12
  1.44268256e-12 2.35227114e-22]
 [5.82387694e-09 9.99992609e-01 2.25220695e-08 7.11702832e-15
  1.89918569e-06 1.03023368e-07 8.88878637e-10 1.41979017e-09
  5.35583422e-06 6.07789372e-13]
 [1.00000000e+00 1.34929596e-16 1.43765699e-14 2.60143985e-17
  2.02902851e-16 6.25009593e-13 1.38456402e-09 4.86662780e-15
  3.15356907e-11 1.25656317e-11]]
```

```
import numpy as np
print(np.argmax(prediction, axis=1)) #printing our Labels from first 4 images
print(y_test[:4]) #printing the actual Labels
```

```
[7 2 1 0]
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

SAVE THE MODEL

```
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).
```

```
%cd /content/drive/MyDrive/DISASTER DATASET/dataset.zip
```

```
/content/drive/MyDrive/DISASTER DATASET/dataset.zip
```

```
# Save the model
model.save('models/disaster.h5')
```

TEST WITH SAVED MODEL

```
# Taking images as input and checking results

# Importing the keras libraries and packages
from tensorflow.keras.models import load_model
model = load_model(r:'/content/drive/MyDrive/DISASTER DATASET/dataset.zip')
from PIL import Image #used for manipulating image upload by the user.
for index in range(0):
    img = Image.open("data/" + str(index) + ".png").convert('L') # convert
image to monochrome
    img = img.resize((28,28)) # resizing of input image
    im2arr = np.array(img) #convert to image
    im2arr = im2arr.reshape(28,28,1) #reshaping according to our requirement
```

```
#predicting the Test set results
y_pred = model.predict(im2arr)
print(y_pred)
```

```
[[2.6514371e-08 9.9987853e-01 2.6678819e-09 5.0345729e-17 1.1578899e-04
  3.5318081e-13 1.3091828e-13 5.5813430e-06 1.7989708e-10 2.9838541e-09]]
1/1 [==============================] - 0s 17ms/step
[[3.7167319e-03 1.4217998e-03 8.2274787e-03 7.9998100e-01 1.1364324e-01
  2.0436779e-02 6.3992090e-08 3.5715982e-02 3.3161716e-05 1.6823808e-02]]
1/1 [==============================] - 0s 16ms/step
[[3.89392152e-02 1.03533266e-05 1.84860080e-01 2.39315932e-03
  2.41028378e-04 8.98893850e-05 1.02114845e-02 3.54044059e-05
  7.63198674e-01 2.07284338e-05]]
```

```
index = ['Cyclone' , 'Earthquake' , 'Flood' , 'Wildfire']
result = str(index[pred[0]])
result
```

'Cyclone'