```
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).
```

Libaries

```
# Importing all the necessary libraries
import keras
import h5py
from keras import optimizers
from keras.models import load_model
from keras.layers import Bidirectional
from keras.layers.core import Reshape, Dropout
from keras.utils.vis_utils import plot_model
import os
# import keras_metrics
import matplotlib.pyplot as plt
from keras.layers import Conv1D,Dense, MaxPooling1D, Flatten, GlobalAveragePooling2D,GlobalAveragePooling3D
from keras import regularizers
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from keras import regularizers
from keras.applications.inception_v3 import InceptionV3
from tensorflow.keras import models
from tensorflow.keras import layers
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import re
from nltk.corpus import stopwords
from nltk import word_tokenize
from keras.preprocessing import image
from PIL import Image, ImageFile
from keras import preprocessing, Input
import numpy as np
#model
from keras.applications.vgg16 import VGG16
from keras.applications.vgg19 import VGG19
from keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.resnet50 import ResNet50
from keras.models import Model
from keras.callbacks import ModelCheckpoint, EarlyStopping
from sklearn.metrics import classification_report,f1_score
import pickle as pkl
```

```
# Keras Functional API

import tensorflow as tf
from tensorflow import keras
from keras.layers import Input, Dense, Activation, Dropout,Flatten,Embedding
from keras.layers import Conv1D,MaxPooling1D,GlobalAveragePooling1D
from keras.models import Model
from tensorflow.keras.optimizers import Adam,SGD,Nadam,RMSprop
from tensorflow.keras.models import load_model
```

```
# Keras Functional API

import tensorflow as tf
from tensorflow import keras
from keras.layers import Input, Dense, Activation, Dropout,Flatten,Embedding
from keras.layers import Conv1D,MaxPooling1D,GlobalAveragePooling1D, Bidirectional, LSTM, GRU
from keras.models import Model
from tensorflow.keras.models import load_model
```

```
dataset_path = "/content/drive/MyDrive/DATASET/"
# Validation_path = "/content/drive/MyDrive/DATASET/val_data.csv"
img_dir = "/content/drive/MyDrive/multimodal_image"
```

Fetching Data

```
train_data = pd.read_excel(dataset_path+'train.xlsx')
test_data = pd.read_csv(dataset_path+'test.xlsx')
print("training size:", len(train_data))
print("testing size:", len(test_data))
```

```
training size: 5247
testing size: 584
```

```
train_data.head()
```

	filename	tweet	label	image
0	buildingfire_2017-02-05_04-06-10.txt	Here's some video of the smoldering ruins in W...	damaged_infrastructure	buildingfire_2017-02-05_04-06-10.JPG
1	isiscrimes_2015-08-04_00-18-33.txt	27 person were killed yesterday near idlib nor...	human_damage	isiscrimes_2015-08-04_00-18-33.JPG
2	earthquake_2017-11-	تصویری از خسارات زلزله در سرپل	damaged_infrastructure	earthquake_2017-11-

```
train_data.columns
```

```
Index(['filename', 'tweet', 'label', 'image'], dtype='object')
```

```
train_data['label'].value_counts()
```

```
non_damage                2666
damaged_infrastructure    1246
damaged_nature             459
flood                      348
fires                      309
human_damage               219
Name: label, dtype: int64
```

```
# Encode Labels
train_data["enc_label"]=train_data.label.replace({"non_damage":0,"damaged_infrastructure":1,"damaged_nature":2,"flood":3,"fires":4,"human_da
test_data["enc_label"]=test_data.label.replace({"non_damage":0,"damaged_infrastructure":1,"damaged_nature":2,"flood":3,"fires":4,"human_dama
```

```
# convert float type data into str if any occurs
train_data["tweet"]=train_data["tweet"].astype(str)
test_data["tweet"]=test_data["tweet"].astype(str)
```

```
train_data['enc_label'].value_counts()
```

```
0    2666
1    1246
2     459
3     348
4     309
5     219
Name: enc_label, dtype: int64
```

```
def replace_string(row):
  return row.replace('.JPG','.jpg')

train_data['image'] = train_data['image'].apply(replace_string)
test_data['image'] = test_data['image'].apply(replace_string)
```

Reading Image

[ ] ↳ *3 cells hidden*

Image Fetching

```
def get_input(path):
    ImageFile.LOAD_TRUNCATED_IMAGES = True
    img = Keras.utilis.load_img(path,target_size = (100,100))
    return(img)


# Takes in image and preprocess it
def process_input(img):
    # Converting image to array
    img_data = keras.utils.img_to_array(img)
    # Adding one more dimension to array
    img_data = np.expand_dims(img_data, axis=0)
    #
    # img_data = preprocess_input(img_data)
    return(img_data)


    from tensorflow.keras.preprocessing.image import load_img
    from tensorflow.keras.preprocessing.image import img_to_array
    from tensorflow.keras.applications.vgg16 import preprocess_input
    from tensorflow.keras.applications.vgg16 import decode_predictions
    from tensorflow.keras.applications.vgg16 import VGG16


# Create an array of training images
train_images =[]
for n,i in enumerate(train_img_path):
      input_img = get_input(i)
      process_img = process_input(input_img)
      print(n)
      train_images.append(process_img[0])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-105-5c2161991938> in <cell line: 3>()
      2 train_images =[]
      3 for n,i in enumerate(train_img_path):
----> 4       input_img = get_input(i)
      5       process_img = process_input(input_img)
      6       print(n)

<ipython-input-104-f662ff4b0522> in get_input(path)
      1 def get_input(path):
      2     ImageFile.LOAD_TRUNCATED_IMAGES = True
----> 3     img = Keras.utilis.load_img(path,target_size = (100,100))
      4     return(img)

NameError: name 'Keras' is not defined
```

SEARCH STACK OVERFLOW

```
#  #Create an array of test images
 test_images = []
 for n,i in enumerate(test_img_path):
 input_img = get_input(i)
  process_img = process_input(input_img)
  print(n)
   test_images.append(process_img[0])
```

```
  File "<ipython-input-22-c51ceeae8d30>", line 2
    test_images = []
    ^
IndentationError: unexpected indent
```

SEARCH STACK OVERFLOW

```
#  convert into numpy array
 train_images  =  np.array(train_images)
# # convert into numpy array
 test_image = np.array(test_images)
```

```
  File "<ipython-input-48-b77e0415cec1>", line 2
    train_images  =  np.array(train_images)
    ^
IndentationError: unexpected indent
```

SEARCH STACK OVERFLOW

```
with open(dataset_path+'train.pkl','wb') as f:
    pkl.dump(train_image, f)

with open(dataset_path+'test.pkl','wb') as f:
    pkl.dump(test_image, f)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-27-0e2d450a21ed> in <cell line: 1>()
      1 with open(dataset_path+'train.pkl','wb') as f:
----> 2     pkl.dump(train_image, f)
      3
      4 with open(dataset_path+'test.pkl','wb') as f:
      5     pkl.dump(test_image, f)

NameError: name 'train_image' is not defined
```

SEARCH STACK OVERFLOW

Load Images

```
with open(dataset_path+'train.pkl','rb') as f:
  train_image = pkl.load(f)
  print("Training Images:-- ",train_image.shape)

with open(dataset_path+'test.pkl','rb') as f:
  test_image = pkl.load(f)
  print("Test Images:-- ",test_image.shape)
```

```
---------------------------------------------------------------------------
EOFError                                  Traceback (most recent call last)
<ipython-input-68-28c364eb67e9> in <cell line: 1>()
      1 with open(dataset_path+'train.pkl','rb') as f:
----> 2   train_image = pkl.load(f)
      3   print("Training Images:-- ",train_image.shape)
      4
      5 with open(dataset_path+'test.pkl','rb') as f:

EOFError: Ran out of input
```

SEARCH STACK OVERFLOW

Callback

[ ] ↳ *1 cell hidden*

VGG16 model

[ ] ↳ *9 cells hidden*

Build VGG19 model

[ ] ↳ *10 cells hidden*

Build Inception Model

[ ]

↳ *10 cells hidden*

Build Resnet Model

```
# from tensorflow.keras.applications.resnet50  import preprocess_input
```

Resnet50

[ ] ↳ *10 cells hidden*

Text Model

Clean Text

```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
STOPWORDS = set(stopwords.words('english'))
REPLACE_BY_SPACE_RE = re.compile('[/(){}\[\]\|@,;]')
BAD_SYMBOLS_RE = re.compile('[^0-9a-z #+_]')
EMAIL = re.compile('^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$')
# NUMBERS = re.compile(['0-9'])
STOPWORDS = set(stopwords.words('english'))
```

```
# def clean_text(text):
#     """
#         text: a string

#         return: modified initial string
#     """
#     text = text.lower()
#     text = EMAIL.sub('', text)
# # #     text = NUMBERS.sub('',text)
# #     text = REPLACE_BY_SPACE_RE.sub(' ',text)
# #     text = BAD_SYMBOLS_RE.sub('',text)
# #     text = text.replace('x','')
#     # text = ' '.join(word for word in text.split() if word not in STOPWORDS)

#     return text
```

```
'''
Text Cleaning
'''
from bs4 import BeautifulSoup
def clean_text(row):
  #to remove HTML tags
  text = BeautifulSoup(row, 'html.parser').get_text()
  d = re.sub(r'(https|http)?:\/\/(\w|\.|\/|\?|\=|\&|\%)*\b', '', text, flags=re.MULTILINE) #This line is for removing url
  post = d.replace('\n', '')
  post = post.replace('—', ' ')
  # to remove accented characters
  # to remove special characters and numbers
  # define the pattern to keep
  pat = r'[^#@a-zA-z0-9]'
  text = re.sub(pat, ' ', post)
  #to remove punctuation
  #text = ''.join([c for c in text if c not in string.punctuation])
  # to remove special characters
  #pattern = r'^\s*|\s\s*'
  #text = re.sub(pattern, ' ', text).strip()
  # convert into lower case
  text = text.lower()
  # Stopword Removing
  #tokenizer = ToktokTokenizer()
  # convert sentence into token of words
  #tokens = tokenizer.tokenize(text)
  #tokens = [token.strip() for token in tokens]


  return text


train_data['clean_Text'] = train_data['tweet'].apply(clean_text)
test_data['clean_Text'] = test_data['tweet'].apply(clean_text)


test_data['clean_Text'][0]
```

```
'  we are really getting into the christmas spirit at miracle tiny toes    this little
miracle and her friends are getting excited for christmas watch to find out why   featu
ring kendal leigh #baby #babyboutique #babies1stchristmas #babiesofinstagram #babykeeps
ake #keepsake #christmas #winter #instagram #babyshower #babygift #unisex #babyboy #bab
```

```
np.unique(train_data['enc_label'])
```

```
array([0, 1, 2, 3, 4, 5])
```

```
train_data['enc_label'].value_counts()
```

```
0    2666
1    1246
2     459
3     348
4     309
5     219
Name: enc_label, dtype: int64
```

```
test_data.head()
```

	filename	tweet	label	image	enc_label	clean_Text
0	ad_2017-11-25_10-36-26.txt	☆ We are really getting into the christmas spi...	non_damage	ad_2017-11-25_10-36-26.jpg	0	we are really getting into the christmas spi...
1	building_2017-10-30_17-26-34.txt	IJOY uv board has the competetive price and	non_damage	building_2017-10-30_17-26-34.jpg	0	ijoy uv board has the competetive price and

Tokenization

[ ] *2 cells hidden*

Encoding Data into Numbers

[ ] ↳ *16 cells hidden*

BiLstm

[ ] ↳ *5 cells hidden*

BiLstm + CNN model

[ ] ↳ *12 cells hidden*

BiLstm + Attention

[ ] ↳ *8 cells hidden*

Bert-Embedding

[ ] ↳ *3 cells hidden*

multilingual-bert

[ ] ↳ *19 cells hidden*

English-bert

[ ] ↳ *16 cells hidden*

Tweet-bert

[ ] ↳ *7 cells hidden*

Multimodal

[ ] ↳ *9 cells hidden*

Inception + BilstmAttention

[ ] ↳ *5 cells hidden*

Resnet50 + BilstmAttention

```
keras.backend.clear_session()

max_length = 100
embedding_dim = 100
number_of_classes = 6

inputs=Input(shape=(max_length,), name='input')
x=Embedding(max_words,embedding_dim)(inputs)
att_in=Bidirectional(LSTM(256,return_sequences=True,dropout=0.3,recurrent_dropout=0.2))(x)
att_out=attention()(att_in)
branch_1 = Dense(256, activation='relu')(att_out)
```

```
# Image input branch - a pre-trained Inception module followed by an added fully connected layer
base_model = ResNet50(weights='imagenet', include_top=False)
# Freeze Inception's weights - we don't want to train these
for layer in base_model.layers:
    layer.trainable = False

# add a fully connected layer after Inception - we do want to train these
branch_2 = base_model.output
branch_2 = GlobalAveragePooling2D()(branch_2)
branch_2 = Dense(512, activation='relu')(branch_2)

# merge the text input branch and the image input branch and add another fully connected layer
joint = concatenate([branch_1, branch_2])
joint = Dense(100, activation='relu')(joint)
# joint = Dropout(0.5)(joint)
predictions = Dense(6, activation='softmax')(joint)

full_model = Model(inputs=[base_model.input,inputs], outputs=[predictions])

full_model.compile(loss='sparse_categorical_crossentropy',
                   optimizer=Adam(),
                   metrics=['accuracy'])



history = full_model.fit([train_image, train_corpus], train_data['enc_label'],
                         epochs=10, batch_size=32,
                         verbose=1, validation_split=0.1, callbacks=checkpoint_fn('ResBiLstmAttention'),shuffle=True)
```

```
Epoch 1/10
148/148 [==============================] - ETA: 0s - loss: 0.8883 - accuracy: 0.6872
Epoch 1: val_accuracy improved from -inf to 0.73905, saving model to /content/drive/MyDrive/New folder/Model/ResBiLstmAttention.h5
148/148 [==============================] - 153s 933ms/step - loss: 0.8883 - accuracy: 0.6872 - val_loss: 0.7871 - val_accuracy: 0.7390
Epoch 2/10
148/148 [==============================] - ETA: 0s - loss: 0.4523 - accuracy: 0.8386
Epoch 2: val_accuracy improved from 0.73905 to 0.77524, saving model to /content/drive/MyDrive/New folder/Model/ResBiLstmAttention.h5
148/148 [==============================] - 133s 899ms/step - loss: 0.4523 - accuracy: 0.8386 - val_loss: 0.6807 - val_accuracy: 0.7752
Epoch 3/10
148/148 [==============================] - ETA: 0s - loss: 0.2665 - accuracy: 0.9039
Epoch 3: val_accuracy improved from 0.77524 to 0.79048, saving model to /content/drive/MyDrive/New folder/Model/ResBiLstmAttention.h5
148/148 [==============================] - 135s 912ms/step - loss: 0.2665 - accuracy: 0.9039 - val_loss: 0.6223 - val_accuracy: 0.7905
Epoch 4/10
148/148 [==============================] - ETA: 0s - loss: 0.1670 - accuracy: 0.9460
Epoch 4: val_accuracy did not improve from 0.79048
148/148 [==============================] - 132s 894ms/step - loss: 0.1670 - accuracy: 0.9460 - val_loss: 0.6684 - val_accuracy: 0.7733
Epoch 5/10
148/148 [==============================] - ETA: 0s - loss: 0.1158 - accuracy: 0.9608
Epoch 5: val_accuracy improved from 0.79048 to 0.83238, saving model to /content/drive/MyDrive/New folder/Model/ResBiLstmAttention.h5
148/148 [==============================] - 133s 898ms/step - loss: 0.1158 - accuracy: 0.9608 - val_loss: 0.6221 - val_accuracy: 0.8324
Epoch 6/10
148/148 [==============================] - ETA: 0s - loss: 0.0453 - accuracy: 0.9869
Epoch 6: val_accuracy did not improve from 0.83238
148/148 [==============================] - 129s 874ms/step - loss: 0.0453 - accuracy: 0.9869 - val_loss: 0.6631 - val_accuracy: 0.8210
Epoch 7/10
148/148 [==============================] - ETA: 0s - loss: 0.0272 - accuracy: 0.9915
Epoch 7: val_accuracy did not improve from 0.83238
148/148 [==============================] - 130s 881ms/step - loss: 0.0272 - accuracy: 0.9915 - val_loss: 0.7568 - val_accuracy: 0.8114
Epoch 8/10
148/148 [==============================] - ETA: 0s - loss: 0.0122 - accuracy: 0.9968
Epoch 8: val_accuracy did not improve from 0.83238
148/148 [==============================] - 132s 891ms/step - loss: 0.0122 - accuracy: 0.9968 - val_loss: 0.7935 - val_accuracy: 0.8324
Epoch 9/10
148/148 [==============================] - ETA: 0s - loss: 0.0110 - accuracy: 0.9972
Epoch 9: val_accuracy did not improve from 0.83238
148/148 [==============================] - 128s 867ms/step - loss: 0.0110 - accuracy: 0.9972 - val_loss: 0.8000 - val_accuracy: 0.8305
Epoch 10/10
148/148 [==============================] - ETA: 0s - loss: 0.0019 - accuracy: 0.9998
Epoch 10: val_accuracy improved from 0.83238 to 0.84571, saving model to /content/drive/MyDrive/New folder/Model/ResBiLstmAttention.h5
148/148 [==============================] - 133s 899ms/step - loss: 0.0019 - accuracy: 0.9998 - val_loss: 0.8132 - val_accuracy: 0.8457
```

```
y_pred = np.argmax(full_model.predict([test_image,test_corpus]), axis=-1)
```

```
19/19 [==============================] - 3s 95ms/step
```

```
print(classification_report(test_data["enc_label"], y_pred))
```

```
                  precision    recall  f1-score   support

               0       0.97      0.97      0.97       291
```

```
                  1       0.85      0.81      0.83       144
                  2       0.66      0.75      0.70        55
                  3       0.73      0.75      0.74        36
                  4       0.79      0.81      0.80        37
                  5       0.88      0.71      0.79        21

           accuracy                           0.88       584
          macro avg       0.81      0.80      0.81       584
       weighted avg       0.88      0.88      0.88       584
```

```
f1_score(test_data["enc_label"], y_pred, average='weighted')
```