```
!pip3 install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (1.5
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages (1
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from k
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages
```

```
!mkdir .kaggle
```

```
!mkdir ~/.kaggle
!touch .kaggle/kaggle.json
```

```
import json
token={"username":"farazg","key":"b5be7130eafb77e0096caa86fec3d14c"}
with open('/root/.kaggle/kaggle.json','w') as file:
    json.dump(token, file)

!chmod 600 ~/.kaggle/kaggle.json

!kaggle datasets download -d srbhshinde/flickr8k-sau
```

```
Downloading flickr8k-sau.zip to /content
100% 2.07G/2.08G [01:02<00:00, 38.9MB/s]
100% 2.08G/2.08G [01:02<00:00, 35.6MB/s]
```

```
ls
```

```
flickr8k-sau.zip  glove.6B.50d.txt  sample_data/
```

```
!unzip flickr8k-sau.zip
```

```
Streaming output truncated to the last 5000 lines.
  inflating: flickr8k-sau/Flickr_Data/Images/2844747252_64567cf14a.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2844846111_8c1cbfc75d.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2844963839_ff09cdb81f.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2845246160_d0d1bbd6f0.jpg
```

```
  inflating: flickr8k-sau/Flickr_Data/Images/2845691057_d4ab89d889.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2845845721_d0bc113ff7.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2846037553_1a1de50709.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2846785268_904c5fcf9f.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2846843520_b0e6211478.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2847514745_9a35493023.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2847615962_c330bded6e.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2847859796_4d9cb0d31f.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2848266893_9693c66275.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2848571082_26454cb981.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2848895544_6d06210e9d.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2848977044_446a31d86e.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2849194983_2968c72832.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2850719435_221f15e951.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2851198725_37b6027625.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2851304910_b5721199bc.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2851931813_eaf8ed7be3.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2852982055_8112d0964f.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/285306009_f6ddabe687.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2853205396_4fbe8d7a73.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2853407781_c9fea8eef4.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2853743795_e90ebc669d.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2853811730_fbb8ab0878.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2854207034_1f00555703.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2854234756_8c0e472f51.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2854291706_d4c31dbf56.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2854959952_3991a385ab.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2855417531_521bf47b50.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2855594918_1d1e6a6061.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2855667597_bf6ceaef8e.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2855695119_4342aae0a3.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2855727603_e917ded363.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/285586547_c81f8905a1.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2855910826_d075845288.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2856080862_95d793fa9d.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2856252334_1b1a230e70.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2856456013_335297f587.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2856524322_1d04452a21.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2856699493_65edef80a1.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2856700531_312528eea4.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2856923934_6eb8832c9a.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2857372127_d86639002c.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2857473929_4f52662c30.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2857558098_98e9249284.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2857609295_16aaa85293.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2858439751_daa3a30ab8.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2858759108_6e697c5f3e.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2858903676_6278f07ee3.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2860035355_3fe7a5caa4.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2860040276_eac0aca4fc.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2860041212_797afd6ccf.jpg
  inflating: flickr8k-sau/Flickr_Data/Images/2860202109_97b2b22652.jpg
```

```
ls
```

```
flickr8k-sau/  flickr8k-sau.zip  Flickr_Data/  glove.6B.50d.txt  sample_data/
```

```
import pandas as pd
import numpy as  np
import matplotlib.pyplot as plt
import keras
import re
import nltk
from nltk.corpus import stopwords

import string
import json
from time import time
import pickle
from keras.applications.vgg16 import VGG16
from keras.applications.resnet import ResNet50, preprocess_input,decode_predictions
from keras.preprocessing import image
from keras.models import Model,load_model
from keras.preprocessing.sequence import pad_sequences

from keras.layers import Input, Dense, Dropout,Embedding,LSTM

from keras.layers.merge import add
from tensorflow.keras.utils import to_categorical

cd Flickr_Data/
```

```
/content/Flickr_Data
```

```
ls
```

```
Flickr_TextData/  Images/
```

```
cd ..
```

```
/content
```

```
fliker_path='/content/Flickr_Data/'



with open(fliker_path+"Flickr_TextData/Flickr8k.token.txt") as filepath:
  caption=filepath.read()
  filepath.close()


captions= caption.split("\n")[:-1]


len(captions)
```

```
40460
```

```
captions[:5]
```

```
['1000268201_693b08cb0e.jpg#0\tA child in a pink dress is climbing up a set of stairs
 '1000268201_693b08cb0e.jpg#1\tA girl going into a wooden building .',
 '1000268201_693b08cb0e.jpg#2\tA little girl climbing into a wooden playhouse .',
 '1000268201_693b08cb0e.jpg#3\tA little girl climbing the stairs to her playhouse .',
 '1000268201_693b08cb0e.jpg#4\tA little girl in a pink dress going into a wooden cabi
```

```
dicti={}

for i in captions:
     captions_da=i.split("\t")
     image_name=captions_da[0].split(".")[0]
     captions_data=captions_da[1]

     if dicti.get(image_name) == None:
          dicti[image_name]=[]

     dicti[image_name].append(captions_data)


dicti["1000268201_693b08cb0e"]
```

```
['A child in a pink dress is climbing up a set of stairs in an entry way .',
 'A girl going into a wooden building .',
 'A little girl climbing into a wooden playhouse .',
 'A little girl climbing the stairs to her playhouse .',
 'A little girl in a pink dress going into a wooden cabin .']
```

```
#cleaning the data


def data_clean(caption):
    caption= caption.lower()
    caption= re.sub("[^a-z]+"," ",caption)
    caption= caption.split()
    caption= [x for x in caption if len(x)>1]
    caption =" ".join(caption)
    return caption


test_data="xvnoizhdnsdoha&5svndklvakcavak;vnadpvjddvm;dvoadjvdAABSBD"
result=data_clean(test_data)
print(result)
```

```
xvnoizhdnsdoha svndklvakcavak vnadpvjddvm dvoadjvdaabsbd
```

```
for keys,values in dicti.items():
  for  i  in range(0,len(values)):
        values[i]=data_clean(values[i])

for keys,values in dicti.items():
       for i in range(0,len(values)):
            print(values[i])
```

```
Streaming output truncated to the last 5000 lines.
dog is jumping over log in wooded area while carrying another log
dog with stick in his mouth jumps over fallen tree in the forest
dog carries stick and jumps over log
the dog carries stick and jumps over log in the woods
the dog jumps over the log with stick in its mouth
black and brown dog is running between two cement barriers with snow
dog runs down the cold aisle
black and tan small dog walking with perked ears
the black and brown dog walks toward the camera in an enclosed snowy area
the brown and black dog is running through snowy street
furry dog is running through doorway leading to plants
furry tan dog is outside on patio
short furry dog stands on brick floor in front of group of potted plants
very hairy dog is running down hall
flowers are behind the fluffy dog that is coming up the step
brown and black dog runs through the leaves
brown dog is running
brown dog with red collar jumping across leafy lawn
dog with brindle colored coat is running across the yard
the brown dog is wearing red collar
crowd walks along sidewalk of farmers market
street market filled with white tents pedestrians and vendors
woman looks over table of organic vegetables at farmer market
men and women walking on the sidewalk outside marketplace
street with many white tents with display table with people walking nearby
fruit stand with group of people standing around it
group of people are standing in linet at fruit stand
group of people shop for fruit at an urban farmers market
many people stand near fruit vendor on street
people shopping at fruit stand
group of people stand at farmers market on dreary day
white tented fruit stand with several people shopping in it
people shop for fresh produce at an outdoor market in the city
people visiting street market
several people shop at an outdoor farmer market on cloudy day
busy highway scene with woman crossing crosswalk on the left
lady crosses busy road on the crosswalk
woman crosses street while traffic lines up in the opposite direction
street with traffic with one person in crosswalk two others at corner and one pers
woman walking in crosswalk near busy street
firetruck fights fire
yellow firetruck is parked next to fire with man on ladder pouring water on it
firefighters putting out big fire
fireman fighting fire
people putting out fire
black dog is swimming while carrying tennis
black dog swimming in the water with tennis ball in his mouth
black dog swims through the water with tennis ball in its mouth
dog swims in water with blue and green tennis ball in its mouth
dog with ball in its mouth swims in the water
brown dog is chewing on bone with stuffed animal underneath it
dog chews on bone
big brown dog chews on bone lying down
big hairy dog chews on bone while lying on furry toy
brown dog chews on bone while laying on the rug
```

```
dicti["1000268201_693b08cb0e"]
```

```
     ['child in pink dress is climbing up set of stairs in an entry way',
      'girl going into wooden building',
      'little girl climbing into wooden playhouse',
      'little girl climbing the stairs to her playhouse',
      'little girl in pink dress going into wooden cabin']
```

```
clean_file=open("dicti.txt","w")
clean_file.write(str(dicti))
clean_file.close()
```

```
cleaned_data=open("dicti.txt",'r')
clear_data=cleaned_data.read()
cleaned_data.close()
```

```
clear_data
```

```
     '{'1000268201_693b08cb0e': ['child in pink dress is climbing up set of stairs in an
     entry way', 'girl going into wooden building', 'little girl climbing into wooden pla
     yhouse', 'little girl climbing the stairs to her playhouse', 'little girl in pink dr
     ess going into wooden cabin'], '1001773457_577c3a7d70': ['black dog and spotted dog
     are fighting', 'black dog and tri colored dog playing with each other on the road',
     'black dog and white dog with brown spots are staring at each other in the street',
     'two dogs of different breeds looking at each other on the road', 'two dogs on pavem
     ent moving toward each other'], '1002674143_1b742ab4b8': ['little girl covered in pa
```

```
unique_vocabulary = set()

for key in dicti.keys():
      for x in dicti[key]:
         unique_vocabulary.update(x.split())
print("vocabularay size:%d" % len(unique_vocabulary))
```

```
     vocabularay size:8424
```

```
total_vocabularies=[]
```

```
for lis in dicti.keys():
   for dic in dicti[keys]:
      for x in dic.split():
          total_vocabularies.append(x)

print("length:",len(total_vocabularies))
print(total_vocabularies[:3000])
```

```
     length: 315588
     ['man', 'in', 'pink', 'shirt', 'climbs', 'rock', 'face', 'man', 'is', 'rock', 'climbi
```

```
import collections

counting= collections.Counter(total_vocabularies)
count_of_words=dict(counting)

print(count_of_words)
print(len(count_of_words))
```

```
{'man': 16184, 'in': 40460, 'pink': 8092, 'shirt': 24276, 'climbs': 8092, 'rock': 485
22
```

```
sorted_data = sorted(count_of_words.items(),reverse=True,key=lambda x:x[1])
print(sorted_data)
len(sorted_data)
```

```
[('rock', 48552), ('in', 40460), ('shirt', 24276), ('climbing', 24276), ('man', 16184
22
```

```
threshold_value=5

vocab= [x for x in sorted_data if x[1]>threshold_value]
print(vocab)
len(vocab)
```

```
[('rock', 48552), ('in', 40460), ('shirt', 24276), ('climbing', 24276), ('man', 16184
22
```

```
final_vocab_list=[x[0] for x in vocab]
print(final_vocab_list)
len(final_vocab_list)
```

```
['rock', 'in', 'shirt', 'climbing', 'man', 'face', 'red', 'climber', 'pink', 'climbs
22
```

```
#prep imagetraining data


traning_data=open("/content/flickr8k-sau/Flickr_Data/Flickr_TextData/Flickr_8k.trainImages
train=traning_data.read()
traning_data.close()

print(train)
len(train)
```

```
2513260012_03d33305cf.jpg
2903617548_d3e38d7f88.jpg
3338291921_fe7ae0c8f8.jpg
488416045_1c6d903fe0.jpg
2644326817_8f45080b87.jpg
218342358_1755a9cce1.jpg
2501968935_02f2cd8079.jpg
2699342860_5288e203ea.jpg
2638369467_8fc251595b.jpg
2926786902_815a99a154.jpg
2851304910_b5721199bc.jpg
3423802527_94bd2b23b0.jpg
3356369156_074750c6cc.jpg
2294598473_40637b5c04.jpg
1191338263_a4fa073154.jpg
2380765956_6313d8cae3.jpg
3197891333_b1b0fd1702.jpg
3119887967_271a097464.jpg
2276499757_b44dc6f8ce.jpg
2506892928_7e79bec613.jpg
2187222896_c206d63396.jpg
2826769554_85c90864c9.jpg
3097196395_ec06075389.jpg
3603116579_4a28a932e2.jpg
3339263085_6db9fd0981.jpg
2532262109_87429a2cae.jpg
2076906555_c20dc082db.jpg
2502007071_82a8c639cf.jpg
3113769557_9edbb8275c.jpg
3325974730_3ee192e4ff.jpg
1655781989_b15ab4cbff.jpg
1662261486_db967930de.jpg
2410562803_56ec09f41c.jpg
2469498117_b4543e1460.jpg
69710415_5c2bfb1058.jpg
3414734842_beb543f400.jpg
3006217970_90b42e6b27.jpg
2192411521_9c7e488c5e.jpg
3535879138_9281dc83d5.jpg
2685788323_ceab14534a.jpg
3465606652_f380a38050.jpg
2599131872_65789d86d5.jpg
2244613488_4d1f9edb33.jpg
2738077433_10e6264b6f.jpg
3537201804_ce07aff237.jpg
1597557856_30640e0b43.jpg
3357194782_c261bb6cbf.jpg
3682038869_585075b5ff.jpg
236474697_0c73dd5d8b.jpg
2641288004_30ce961211.jpg
267164457_2e8b4d30aa.jpg
2453891449_fedb277908.jpg
281419391_522557ce27.jpg
354999632_915ea81e53.jpg
3109136206_f7d201b368.jpg
2281054343_95d6d3b882.jpg
3296584432_bef3c965a3.jpg
```

```
        3526431764_056d2c61dc.jpg
```

```
train=[x.split(".")[0] for x in train.split("\n")[:-1]]
train
```

```
        ['2513260012_03d33305cf',
         '2903617548_d3e38d7f88',
         '3338291921_fe7ae0c8f8',
         '488416045_1c6d903fe0',
         '2644326817_8f45080b87',
         '218342358_1755a9cce1',
         '2501968935_02f2cd8079',
         '2699342860_5288e203ea',
         '2638369467_8fc251595b',
         '2926786902_815a99a154',
         '2851304910_b5721199bc',
         '3423802527_94bd2b23b0',
         '3356369156_074750c6cc',
         '2294598473_40637b5c04',
         '1191338263_a4fa073154',
         '2380765956_6313d8cae3',
         '3197891333_b1b0fd1702',
         '3119887967_271a097464',
         '2276499757_b44dc6f8ce',
         '2506892928_7e79bec613',
         '2187222896_c206d63396',
         '2826769554_85c90864c9',
         '3097196395_ec06075389',
         '3603116579_4a28a932e2',
         '3339263085_6db9fd0981',
         '2532262109_87429a2cae',
         '2076906555_c20dc082db',
         '2502007071_82a8c639cf',
         '3113769557_9edbb8275c',
         '3325974730_3ee192e4ff',
         '1655781989_b15ab4cbff',
         '1662261486_db967930de',
         '2410562803_56ec09f41c',
         '2469498117_b4543e1460',
         '69710415_5c2bfb1058',
         '3414734842_beb543f400',
         '3006217970_90b42e6b27',
         '2192411521_9c7e488c5e',
         '3535879138_9281dc83d5',
         '2685788323_ceab14534a',
         '3465606652_f380a38050',
         '2599131872_65789d86d5',
         '2244613488_4d1f9edb33',
         '2738077433_10e6264b6f',
         '3537201804_ce07aff237',
         '1597557856_30640e0b43',
         '3357194782_c261bb6cbf',
         '3682038869_585075b5ff',
         '236474697_0c73dd5d8b',
         '2641288004_30ce961211',
         '267164457_2e8b4d30aa',
         '2453891449_fedb277908',
         '281419391_522557ce27',
         '354999632_915ea81e53',
```

```
        '3109136206_f7d201b368',
        '2281054343_95d6d3b882',
        '3296584432_bef3c965a3',
        '3526431764_056d2c61dc'
```

```
testing_data=open("/content/flickr8k-sau/Flickr_Data/Flickr_TextData/Flickr_8k.testImages.
test_data=testing_data.read()
testing_data.close()

print(test_data)
```

```
     3385593926_d3e9c21170.jpg
     2677656448_6b7e7702af.jpg
     311146855_0b65fdb169.jpg
     1258913059_07c613f7ff.jpg
     241347760_d44c8d3a01.jpg
     2654514044_a70a6e2c21.jpg
     2339106348_2df90aa6a9.jpg
     256085101_2c2617c5d0.jpg
     280706862_14c30d734a.jpg
     3072172967_630e9c69d0.jpg
     3482062809_3b694322c4.jpg
     1167669558_87a8a467d6.jpg
     2847615962_c330bded6e.jpg
     3344233740_c010378da7.jpg
     2435685480_a79d42e564.jpg
     3110649716_c17e14670e.jpg
     2511019188_ca71775f2d.jpg
     2521770311_3086ca90de.jpg
     2723477522_d89f5ac62b.jpg
     2218609886_892dcd6915.jpg
     3745451546_fc8ec70cbd.jpg
     2844018783_524b08e5aa.jpg
     3100251515_c68027cc22.jpg
     2207244634_1db1a1890b.jpg
     2943023421_e297f05e11.jpg
     3286822339_5535af6b93.jpg
     2479652566_8f9fac8af5.jpg
     1394368714_3bc7c19969.jpg
     872622575_ba1d3632cc.jpg
     2309860995_c2e2a0feeb.jpg
     241347204_007d83e252.jpg
     3502343542_f9b46688e5.jpg
     757332692_6866ae545c.jpg
     2748729903_3c7c920c4d.jpg
     494792770_2c5f767ac0.jpg
     3213992947_3f3f967a9f.jpg
     2295750198_6d152d7ceb.jpg
     2358898017_24496b80e8.jpg
     3222055946_45f7293bb2.jpg
     444481722_690d0cadcf.jpg
     2647049174_0fb47cee2e.jpg
     1174629344_a2e1a2bdbf.jpg
     2921094201_2ed70a7963.jpg
     2553550034_5901aa9d6c.jpg
     3045613316_4e88862836.jpg
     2706766641_a9df81969d.jpg
     510531976_90bbee22a2.jpg
     485245061_5a5de43e20.jpg
```

```
3070011270_390e597783.jpg
1352410176_af6b139734.jpg
1131932671_c8d17751b3.jpg
3155451946_c0862c70cb.jpg
2762301555_48a0d0aa24.jpg
3442242092_e579538d82.jpg
2415803492_56a673dc25.jpg
2884301336_dc8e974431.jpg
3453259666_9ecaa8bb4b.jpg
```

```
test=[x.split(".")[0] for x in test_data.split("\n")[:-1]]
print(test)
```

```
['3385593926_d3e9c21170', '2677656448_6b7e7702af', '311146855_0b65fdb169', '125891305
```

```
train_dicti={}

print(train[0])
for i in train:
  train_dicti[i]=[]
  for x in dicti[i]:
   final_cap="<starting of the sequence>"+ x +"<ending of  the sequence>"
   train_dicti[i].append(final_cap)
```

```
2513260012_03d33305cf
```

```
train_dicti["2513260012_03d33305cf"]
```

```
['<starting of the sequence>black dog is running after white dog in the snow<ending o
 '<starting of the sequence>black dog chasing brown dog through snow<ending of  the s
 '<starting of the sequence>two dogs chase each other across the snowy ground<ending
 '<starting of the sequence>two dogs play together in the snow<ending of  the sequenc
 '<starting of the sequence>two dogs running through low lying body of water<ending o
```

```
train_dicti['1000268201_693b08cb0e']
```

```
['<starting of the sequence>child in pink dress is climbing up set of stairs in an en
 '<starting of the sequence>girl going into wooden building<ending of  the sequence>
 '<starting of the sequence>little girl climbing into wooden playhouse<ending of  the
 '<starting of the sequence>little girl climbing the stairs to her playhouse<ending o
 '<starting of the sequence>little girl in pink dress going into wooden cabin<ending
```

```
img="/content/flickr8k-sau/Flickr_Data/Images"
print(img)
```

```
/content/flickr8k-sau/Flickr_Data/Images
```

```
mymodel=ResNet50(weights="imagenet",input_shape=(224,224,3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/re
102973440/102967424 [==============================] - 1s 0us/step
102981632/102967424 [==============================] - 1s 0us/step
```

```
mymodel.summary()
```

```
Model: "resnet50"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 conv1_pad (ZeroPadding2D)      (None, 230, 230, 3)  0           ['input_1[0][0]']

 conv1_conv (Conv2D)            (None, 112, 112, 64  9472        ['conv1_pad[0][0]
                                )

 conv1_bn (BatchNormalization)  (None, 112, 112, 64  256         ['conv1_conv[0][0
                                )

 conv1_relu (Activation)        (None, 112, 112, 64  0           ['conv1_bn[0][0]'
                                )

 pool1_pad (ZeroPadding2D)      (None, 114, 114, 64  0           ['conv1_relu[0][0
                                )

 pool1_pool (MaxPooling2D)      (None, 56, 56, 64)   0           ['pool1_pad[0][0]

 conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)   4160        ['pool1_pool[0][0

 conv2_block1_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block1_1_
 ization)

 conv2_block1_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2_block1_1_
 n)

 conv2_block1_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2_block1_1_

 conv2_block1_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block1_2_
 ization)

 conv2_block1_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2_block1_2_
 n)

 conv2_block1_0_conv (Conv2D)   (None, 56, 56, 256)  16640       ['pool1_pool[0][0

 conv2_block1_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2_block1_2_

 conv2_block1_0_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2_block1_0_
 ization)

 conv2_block1_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2_block1_3_
 ization)

 conv2_block1_add (Add)         (None, 56, 56, 256)  0           ['conv2_block1_0_
                                                                  'conv2_block1_3_
```

```
 conv2_block1_out (Activation)  (None, 56, 56, 256)  0           ['conv2_block1_ad
 conv2_block2_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2_block1_ou
 conv2_block2_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block2_1_
```

```
new_model=Model(mymodel.input,mymodel.layers[-2].output)
```

```
new_model.summary()
```

```
Model: "model"
_________________________________________________________________________________
 Layer (type)                Output Shape              Param #     Connected to
=================================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3       0           []
                             )]

 conv1_pad (ZeroPadding2D)   (None, 230, 230, 3)       0           ['input_1[0][0]']

 conv1_conv (Conv2D)         (None, 112, 112, 64       9472        ['conv1_pad[0][0]
                             )

 conv1_bn (BatchNormalization)  (None, 112, 112, 64    256         ['conv1_conv[0][0
                             )

 conv1_relu (Activation)     (None, 112, 112, 64       0           ['conv1_bn[0][0]'
                             )

 pool1_pad (ZeroPadding2D)   (None, 114, 114, 64       0           ['conv1_relu[0][0
                             )

 pool1_pool (MaxPooling2D)   (None, 56, 56, 64)        0           ['pool1_pad[0][0]

 conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)     4160        ['pool1_pool[0][0

 conv2_block1_1_bn (BatchNormal  (None, 56, 56, 64)    256         ['conv2_block1_1_
 ization)

 conv2_block1_1_relu (Activatio  (None, 56, 56, 64)    0           ['conv2_block1_1_
 n)

 conv2_block1_2_conv (Conv2D)   (None, 56, 56, 64)     36928       ['conv2_block1_1_

 conv2_block1_2_bn (BatchNormal  (None, 56, 56, 64)    256         ['conv2_block1_2_
 ization)

 conv2_block1_2_relu (Activatio  (None, 56, 56, 64)    0           ['conv2_block1_2_
 n)

 conv2_block1_0_conv (Conv2D)   (None, 56, 56, 256)    16640       ['pool1_pool[0][0

 conv2_block1_3_conv (Conv2D)   (None, 56, 56, 256)    16640       ['conv2_block1_2_

 conv2_block1_0_bn (BatchNormal  (None, 56, 56, 256)   1024        ['conv2_block1_0_
 ization)

 conv2_block1_3_bn (BatchNormal  (None, 56, 56, 256)   1024        ['conv2_block1_3_
 ization)
```

```
conv2_block1_add (Add)         (None, 56, 56, 256)  0           ['conv2_block1_0_
                                                                 'conv2_block1_3_

conv2_block1_out (Activation)  (None, 56, 56, 256)  0           ['conv2_block1_ad

conv2_block2_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2_block1_ou

conv2_block2_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block2_1_
```

```
def preprocess_image(img):
  img=image.load_img(img,target_size=(224,224))
  img=image.img_to_array(img)
  img=np.expand_dims(img,axis=0)
  img=preprocess_input(img)
  return img



def encoding_images(img):
  images=preprocess_image(img)
  featured_vector=new_model.predict(images)
  featured_vector=featured_vector.reshape(featured_vector.shape[1],)
  return featured_vector


for key, values in train_dicti.items():
    print(values)
```

```
Streaming output truncated to the last 5000 lines.
['<starting of the sequence>the couple eat their meal outside<ending of  the seque
['<starting of the sequence>boy and girl playing cricket<ending of  the sequence>'
['<starting of the sequence>dark haired man in his twenties drinks green liquid fr
['<starting of the sequence>brown dog chasing yellow toy<ending of  the sequence>'
['<starting of the sequence>brown dog carrying black object<ending of  the sequenc
['<starting of the sequence>man attached to strings is watched by crowd<ending of
['<starting of the sequence>little boy kicks soccer ball in the park<ending of  th
['<starting of the sequence>girl and boy bounce on large balls<ending of  the sequ
['<starting of the sequence>kid in red falls as he struggles with kid in white to
['<starting of the sequence>man and woman are wearing black and looking at somethi
['<starting of the sequence>man is playing fetch with dog<ending of  the sequence>
['<starting of the sequence>fruit stand with group of people standing around it<en
['<starting of the sequence>man holds ball in the air for brown dog to catch on th
['<starting of the sequence>girl in black tank top catches fish<ending of  the seq
['<starting of the sequence>man and little girl walking down the street<ending of
['<starting of the sequence>man playing sport in green uniform holding paddle over
['<starting of the sequence>child is thrown by man in the swimming pool<ending of
['<starting of the sequence>blond girl in shorts sits on the top of monkey bars<en
['<starting of the sequence>girl messily eats plate of pasta<ending of  the sequen
['<starting of the sequence>group of guys are playing soccer on the beach<ending o
['<starting of the sequence>group of adults are walking<ending of  the sequence>',
['<starting of the sequence>skiiers on snowy mountain<ending of  the sequence>', '
['<starting of the sequence>dachshund puppy jumps on bed<ending of  the sequence>'
['<starting of the sequence>dogs running through snow<ending of  the sequence>', '
['<starting of the sequence>man is sitting by large plant waiting to shine custome
['<starting of the sequence>man sits in chair smoking cigarette<ending of  the seq
['<starting of the sequence>dog carries stick in its mouth<ending of  the sequence
```

```
['<starting of the sequence>man in bright orange shorts is skateboarding along cou
['<starting of the sequence>girl is blowing bubbles heavily<ending of  the sequenc
['<starting of the sequence>couple sits at cramped table in busy restaurant<ending
['<starting of the sequence>bearded man and woman in dress holding cup<ending of
['<starting of the sequence>dad celebrates birthday with his family via web cam<en
['<starting of the sequence>brown dog in red harness chasing red ball<ending of  t
['<starting of the sequence>brown and white dog jumping over red yellow and white
['<starting of the sequence>woman in black walks down the sidewalk<ending of  the
['<starting of the sequence>boy is jumping on an inflatable ring and girl is watch
['<starting of the sequence>bunch of people are watching ice skaters<ending of  th
['<starting of the sequence>lone surfer surfing large collapsing wave in the ocean
['<starting of the sequence>leather clad biker wearing face paint sits on his moto
['<starting of the sequence>bunch of dogs are competing in race<ending of  the seq
['<starting of the sequence>person in the distance hikes among hoodoos with stars
['<starting of the sequence>dog in the forest<ending of  the sequence>', '<startin
['<starting of the sequence>group of people sitting around desk<ending of  the seq
['<starting of the sequence>brown dog is running outside<ending of  the sequence>'
['<starting of the sequence>man standing on rocky mountain with gray clouds in the
['<starting of the sequence>man in an orange jacket raising his hands to the sky i
['<starting of the sequence>man parachuting down snowy mountain<ending of  the seq
['<starting of the sequence>man with rock climbing equipment is hanging from verti
['<starting of the sequence>man and boy sitting with their back to stone wall the
['<starting of the sequence>the two girls are playing on yellow sit and bounce<end
['<starting of the sequence>girl in bathing suit carries paddle<ending of  the seq
['<starting of the sequence>blonde girl is wearing silver helmet elbow and knee pa
['<starting of the sequence>two boys splashing each other in the ocean<ending of
['<starting of the sequence>brown dog is chewing on white rug whilst standing on r
['<starting of the sequence>boy in black pants and green shirt jumps high with his
['<starting of the sequence>dog with tigon colorings steps into the water<ending o
```

```
encode_train={}


for x ,y in enumerate(train):
  y="/content/Flickr_Data/Images/{}.jpg".format(train[x])
  encode_train[y]= encoding_images(y)
  if x%100==0:
    print("encoded images:"+str(x))
```

```
encoded images:0
encoded images:100
encoded images:200
encoded images:300
encoded images:400
encoded images:500
encoded images:600
encoded images:700
encoded images:800
encoded images:900
encoded images:1000
encoded images:1100
encoded images:1200
encoded images:1300
encoded images:1400
encoded images:1500
encoded images:1600
encoded images:1700
```

```
encoded images:1800
encoded images:1900
encoded images:2000
encoded images:2100
encoded images:2200
encoded images:2300
encoded images:2400
encoded images:2500
encoded images:2600
encoded images:2700
encoded images:2800
encoded images:2900
encoded images:3000
encoded images:3100
encoded images:3200
encoded images:3300
encoded images:3400
encoded images:3500
encoded images:3600
encoded images:3700
encoded images:3800
encoded images:3900
encoded images:4000
encoded images:4100
encoded images:4200
encoded images:4300
encoded images:4400
encoded images:4500
encoded images:4600
encoded images:4700
encoded images:4800
encoded images:4900
encoded images:5000
encoded images:5100
encoded images:5200
encoded images:5300
encoded images:5400
encoded images:5500
encoded images:5600
encoded images:5700
```

```
with open("/encoded_train_images.pkl","wb") as encode_pickle:
  pickle.dump(encode_train,encode_pickle)
```

```
mode_path="/content/Flickr_Data/Images/"
encode_train[mode_path+"111766423_4522d36e56.jpg"]
```

```
array([0.2145782 , 0.0010708 , 0.9212484 , ..., 0.25015894, 0.06509534,
       0.        ], dtype=float32)
```

```
encode_test={}
```

```
for x ,y in enumerate(test):
  y="/content/flickr8k-sau/Flickr_Data/Images/{}.jpg".format(test[x])
  encode_test[y[len(img):]]= encoding_images(y)
```

```
  if x%100==0:
    print("encoded images:",str(x))
```

```
encoded images: 0
encoded images: 100
encoded images: 200
encoded images: 300
encoded images: 400
encoded images: 500
encoded images: 600
encoded images: 700
encoded images: 800
encoded images: 900
```

```
with open("/encoded_test_images.pkl","wb") as encode1_pickle:
  pickle.dump(encode_test,encode1_pickle)
```

```
with open("/encoded_train_images.pkl","rb") as encode_pickle:
  encode_train=pickle.load(encode_pickle)
```

```
with open("/encoded_test_images.pkl","rb") as encode1_pickle:
  encode_test=pickle.load(encode1_pickle)
```

```
x=1
index_to_word={}
word_to_index={}

for i in final_vocab_list:
  word_to_index[i]=x
  index_to_word[x]=i
  x=x+1
```

Double-click (or enter) to edit

```
word_to_index['shirt']
```

```
3
```

```
index_to_word[3]
```

```
'shirt'
```

```
word_to_index["<starting of the sequence>"]=23
word_to_index["<ending of  the sequence>"]=24
```

```
word_to_index['<ending of  the sequence>']
```

```
      24

index_to_word[23]="<starting of the sequence>"
index_to_word[24]="<ending of  the sequence>"


vocab_size=len(index_to_word)+2
voc=len(word_to_index)+2
print(vocab_size)
```

```
     24
```

```
length_of_captions=[]


for key in train_dicti.keys():
   for x in train_dicti[key]:
     length_of_captions.append(len(x.split()))




print(length_of_captions)
```

```
     [16, 13, 15, 13, 15, 10, 13, 14, 17, 19, 18, 14, 15, 19, 16, 12, 15, 14, 12, 18, 17,
```

```
MAX_LEN=max(length_of_captions)
print(MAX_LEN)
```

```
     39
```

```
for key, values in train_dicti.items():
           photo=encode_train[mode_path+key+".jpg"]
           print(photo)
           for x in values:
                 print(x)
```

```
     Streaming output truncated to the last 5000 lines.
     <starting of the sequence>person is eating pasta while dog is watching<ending of
     <starting of the sequence>someone in blue and white striped sweater is eating and
     [0.02735292 0.8034104  0.24806657 ... 3.2092013  0.2867026  0.12857762]
     <starting of the sequence>camera team wearing jackets stands in front of white bui
     <starting of the sequence>man is setting up camera to take shots of something that
     <starting of the sequence>man shoots footage with camera as two women look on<endi
     <starting of the sequence>man with long hair is looking through camera<ending of
     <starting of the sequence>three people stand outside one has camera on tripod<endi
     [0.05535315 0.         0.25236526 ... 3.6711478  0.33235058 0.0365812 ]
     <starting of the sequence>group of people at distance on beach<ending of  the seque
     <starting of the sequence>man woman and two girls walk on the beach barefoot<ending
     <starting of the sequence>four people are walking on beach<ending of  the sequence
     <starting of the sequence>people walking in the water along the beach<ending of  t
     <starting of the sequence>this family is walking on the beach<ending of  the seque
     [0.47931138 2.5576572  0.0475945  ... 0.29315484 0.3938042  0.14372586]
```

```
 <starting of the sequence>rock climber<ending of  the sequence>
 <starting of the sequence>there are two people rock climbing one is on the ground v
 <starting of the sequence>two men climb large rock<ending of  the sequence>
 <starting of the sequence>two people rock climbing<ending of  the sequence>
 <starting of the sequence>two rock climbers scaling sheer cliff<ending of  the seq
 [0.40514416 0.8642238  0.8449308  ... 0.26682353 0.43824437 0.6617339 ]
 <starting of the sequence>german shepherd biting the protected arm of an attack tr
 <starting of the sequence>man training dog to attack his padded arm<ending of  the
 <starting of the sequence>man wearing protective clothing is being bitten on the a
 <starting of the sequence>man with stick and arm protection is being bitten by larg
 <starting of the sequence>the man wearing padded clothing is fending off an attack
 [0.18087433 0.38334808 0.         ... 0.00971541 1.738766   0.15751246]
 <starting of the sequence>float representing the times of hanging is shown in the
 <starting of the sequence>man in judge costume stand on red truck and hangs dummy<e
 <starting of the sequence>man dressed as judge pretending to hang another man on t
 <starting of the sequence>three people in costumes stand on the back of pickup tru
 <starting of the sequence>three people stand on the back of truck<ending of  the s
 [0.0690265  0.23239897 0.1679178  ... 1.6400832  1.0242746  1.2327104 ]
 <starting of the sequence>two girls in colourful clothes inside shower enclosure<e
 <starting of the sequence>two girls in shorts are standing inside shower stall<end
 <starting of the sequence>two girls play in the shower with their clothes on<ending
 <starting of the sequence>two girls stand in shower stall<ending of  the sequence>
 <starting of the sequence>two girls wearing colorful shirts and shorts standing in
 [0.07332668 0.4231612  0.11535167 ... 0.00066768 0.         0.00208407]
 <starting of the sequence>beaver on the shore of stream<ending of  the sequence>
 <starting of the sequence>brown furry animal stands behind some plants<ending of  t
 <starting of the sequence>dog hides in the bushes<ending of  the sequence>
 <starting of the sequence>dog hides in the tall grasses along rocky shore<ending of
 <starting of the sequence>dog makes it way through tall weeds<ending of  the seque
 [0.20941599 1.2539737  0.         ... 0.19628383 0.27663782 0.47924826]
 <starting of the sequence>girl walking with her grey umbrella<ending of  the seque
 <starting of the sequence>woman holding an umbrella is standing near parking meter
 <starting of the sequence>woman is walking down the street holding an umbrella<end
 <starting of the sequence>woman with blue umbrella stands next to parking meter<en
 <starting of the sequence>woman in heels black skirt and white shirt holds her blu
 [0.16829225 2.066853   0.03211915 ... 0.09061269 2.4233851  0.9670853 ]
 <starting of the sequence>crowd scene in front of mosque<ending of  the sequence>
 <starting of the sequence>large group is shown in the street<ending of  the sequen
 <starting of the sequence>lot of people are gathered around mosque like building<e
 <starting of the sequence>busy marketplace in an arabian country<ending of  the se
 <starting of the sequence>there are lot of people in front of tan building with co
```

```
def generator(train_dicti,encode_train,word_to_index,MAX_LEN,number_of_pictures_per_bath):
    n_samples = 0

    X= []
    y_in = []
    y_out = []

    while True:

       for key, values in train_dicti.items():
           n_samples=n_samples+1
           photo=encode_train[mode_path+key+".jpg"]
           for x in values:
```

```
                    seq=[word_to_index[word] for word in x.split() if word in word_to_index]
                    for i in range(1, len(seq)):

                        in_seq= [seqeuence[0:i]]
                        out_seq = seq[i]

                        in_seq = pad_sequences([in_seq], maxlen=MAX_LEN,value=0, padding='pos
                        out_seq = to_categorical([out_seq], num_classes=vocab_size)[0]
                        X.append(photo)
                        y_in.append(in_seq)
                        y_out.append(out_seq)

    return X, y_in, y_out

    if n_samples==number_of_pictures_per_bath:
       yield[[np.array(X),np.array(y_in)],np.array( y_out)]
       X= []
       y_in = []
       y_out = []
       n_samples=0


f=open("/content/glove.6B.50d.txt",encoding="utf8")
f
```

```
     <_io.TextIOWrapper name='/content/glove.6B.50d.txt' mode='r' encoding='utf8'>
```

```
embeddings_index = dict()
fid = open('glove.6B.50d.txt',encoding="utf8")
for line in fid:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:],dtype='float32')
    embeddings_index[word] = coefs
fid.close()


def embedding_output():
   e_dim=50
   emb_output=np.zeros((vocab_size,e_dim))

   for x,y in word_to_index.items():
     embedding_vector=embedding_ind.get(x)

     if embedding_vector is not None:
          emb_output[y]=embedding_vector

   return emb_output



embedding_output=embedding_output()


embedding_output
```

```
array([[ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [-0.64670002,  0.98997998, -0.14379001, ..., -1.22590005,
        -0.79443002, -0.4614    ],
       [ 0.33041999,  0.24995001, -0.60873997, ..., -0.50703001,
        -0.027273  , -0.53285003],
       ...,
       [ 0.30045   ,  0.25005999, -0.16692001, ..., -0.07131   ,
         0.23052   , -0.51938999],
       [ 0.26381999,  0.32453001,  0.74185002, ..., -0.77772999,
         0.16744   , -0.81748998],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ]])
```

```
embedding_output.shape
```

```
(24, 50)
```

```
inputs1 = Input(shape=(2048,))
image_feature = Dropout(0.5)(inputs1)
image_feature = Dense(256, activation='relu')(image_feature)
    # sequence model
inputs2 = Input(shape=(MAX_LEN,))
language_feature = Embedding(vocab_size,50, weights=[embedding_output],input_length=MAX_LE
    #Embedding(vocab_size, 256, mask_zero=True)(inputs2) #<<<<<< fix me, add pretrianed em
language_feature = Dropout(0.3)(language_feature)
language_feature = LSTM(256)(language_feature)
    # decoder model
output = add([image_feature, language_feature])
output = Dense(256, activation='relu')(output)
output = Dense(vocab_size, activation='softmax')(output)
    # tie it together [image, seq] [word]
model = Model(inputs=[inputs1, inputs2], outputs=output)
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
    # summarize model
print(model.summary())
```

```
Model: "model_1"
__________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==========================================================================================
 input_3 (InputLayer)           [(None, 39)]         0           []

 input_2 (InputLayer)           [(None, 2048)]       0           []

 embedding (Embedding)          (None, 39, 50)       1200        ['input_3[0][0]']

 dropout (Dropout)              (None, 2048)         0           ['input_2[0][0]']

 dropout_1 (Dropout)            (None, 39, 50)       0           ['embedding[0][0]']

 dense (Dense)                  (None, 256)          524544      ['dropout[0][0]']

 lstm (LSTM)                    (None, 256)          314368      ['dropout_1[0][0]']
```

```
 add (Add)                      (None, 256)          0           ['dense[0][0]',
                                                                  'lstm[0][0]']

 dense_1 (Dense)                (None, 256)          65792       ['add[0][0]']

 dense_2 (Dense)                (None, 24)           6168        ['dense_1[0][0]']

==================================================================================================
Total params: 912,072
Trainable params: 910,872
Non-trainable params: 1,200
__________________________________________________________________________________________________
None
```

```
epochs=20
number_of_pictures_per_bath=3
steps=len(train_dicti)
steps
```

```
6000
```

```
for i in range(epochs):
  gen=generator(train_dicti,encode_train,MAX_LEN,word_to_index,number_of_pictures_per_bath
  model.fit_generator(gen,epochs=1,steps_per_epoch=steps,verbose=1)
  model.save("./model_"+str(i)+".h5")
```

Double-click (or enter) to edit

```
model=load_model("./model_19.h5")
```

```
def predicting_caption(img):
  adding_start="<starting of the sequence>"

  for i in range(MAX_LEN):
    sequence=[word_to_index[x] for x in adding_start.split() if x in word_to_index]
    sequence=pad_sequences([sequence],maxlen=MAX_LEN,padding="post")

    yprediction=model.predict([photo,sequence])
    yprediction=yprediction.argmax()
    word=index_to_word[yprediction]
    adding_start+=" "+word

    if word=="<end of the seqeuence>":
       break
    final_caption=adding_start.split()
    final_caption=final_caption[1:-1]
    final_caption=" ".join(final_caption)
```

```
for i in range(20):
  reimg=np.random.randint(0,1000)
  imgg_name=list(encode_test.keys())[reimg]
  fin_photo=encode_test[imgg_name].reshape((1,2048))


  i=plt.imread(img+imgg_name)
  plt.imshow(i)
  plt.axis("off")
  plt.show()


  final=predicting_caption(fin_photo)
  print(final)
```

```
pip install gTTS
```

```
from gtts import gTTS

# This module is imported so that we can
# play the converted audio
import os

# The text that you want to convert to audio
mytext =text

# Language in which you want to convert
language = 'en'

# Passing the text and language to the engine,
# here we have marked slow=False. Which tells
# the module that the converted audio should
# have a high speed
myobj = gTTS(text=mytext, lang=language, slow=False)

# Saving the converted audio in a mp3 file named
# welcome
myobj.save("introoooooo.mp3")

# Playing the converted file
os.system("mpg321 intro.mp3")
```