Image Captioning-Attention mechanism

Image Captioning -Scratch(attention mechanism)

Introduction

Caption generation is a challenging artificial intelligence problem where a textual description must be generated for a given photograph.It requires both methods from computer vision to understand the content of the image and a language model from the field of natural language processing to turn the understanding of the image into words in the right order.

Problem Statement

With the recent advancement in computational hardware & software, Deep learning (DL) has revolutionized the future of artificial intelligence (AI) and cognition based intelligence. It has shown outstanding results in the areas of computer vision and Natural Language Processing. Many research efforts have produced remarkable results in mapping images to high-level features, as explored in Alex Krizhevsky et al.'s 2012 paper which won the ImageNet Large Scale Visual Recognition Challenge and more recent work has extended deep learning to various use cases

But contextual data predictions in image video and audio have been difficult due to complexity of human language, with various phonetics and meaning of the same words producing various context

Aim

We plan to produce a better model that tries to perform image captioning based on the scene graph and entity relations followed by the reversal of the context to generate the image to demonstrate how the context is interpreted from the image and vice versa

Team Members

1. Ahmed Osama Ahmed
2. Islam Waheed
3. Khaled Nabil
4. Mohamed Hosny ElGharawy
5. Mina Fawzy Bekhit
6. Abdelrahman ElSayed

Techniques Used

- Raditional CNN-RNN model
- Attention model(if required)

- Text and Image preprocessing
- Data Visualization using wordcloud
- Model training and model evaluation (INCEPTIONV3)
- Greedy search vs Beam search and BLUE Score

IMPORTING PACKAGES

In [2]:
```
#Import all the required libraries


!pip install wordcloud
!pip install gtts
!pip install playsound
!pip install tensorflow
```

```
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuron.amazonaws.com
Requirement already satisfied: wordcloud in /home/ubuntu/anaconda3/envs/tensorflow2_p38/
lib/python3.8/site-packages (1.8.2.2)
Requirement already satisfied: pillow in /home/ubuntu/anaconda3/envs/tensorflow2_p38/li
b/python3.8/site-packages (from wordcloud) (9.0.0)
Requirement already satisfied: numpy>=1.6.1 in /home/ubuntu/anaconda3/envs/tensorflow2_p
38/lib/python3.8/site-packages (from wordcloud) (1.18.5)
Requirement already satisfied: matplotlib in /home/ubuntu/anaconda3/envs/tensorflow2_p3
8/lib/python3.8/site-packages (from wordcloud) (3.4.3)
Requirement already satisfied: cycler>=0.10 in /home/ubuntu/anaconda3/envs/tensorflow2_p
38/lib/python3.8/site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: python-dateutil>=2.7 in /home/ubuntu/anaconda3/envs/tenso
rflow2_p38/lib/python3.8/site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /home/ubuntu/anaconda3/envs/tensorfl
ow2_p38/lib/python3.8/site-packages (from matplotlib->wordcloud) (1.3.2)
Requirement already satisfied: pyparsing>=2.2.1 in /home/ubuntu/anaconda3/envs/tensorflo
w2_p38/lib/python3.8/site-packages (from matplotlib->wordcloud) (3.0.6)
Requirement already satisfied: six>=1.5 in /home/ubuntu/anaconda3/envs/tensorflow2_p38/l
ib/python3.8/site-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuron.amazonaws.com
Requirement already satisfied: gtts in /home/ubuntu/anaconda3/envs/tensorflow2_p38/lib/p
ython3.8/site-packages (2.2.4)
Requirement already satisfied: requests in /home/ubuntu/anaconda3/envs/tensorflow2_p38/l
ib/python3.8/site-packages (from gtts) (2.26.0)
Requirement already satisfied: click in /home/ubuntu/anaconda3/envs/tensorflow2_p38/lib/
python3.8/site-packages (from gtts) (8.0.3)
Requirement already satisfied: six in /home/ubuntu/anaconda3/envs/tensorflow2_p38/lib/py
thon3.8/site-packages (from gtts) (1.16.0)
Requirement already satisfied: charset-normalizer~=2.0.0 in /home/ubuntu/anaconda3/envs/
tensorflow2_p38/lib/python3.8/site-packages (from requests->gtts) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /home/ubuntu/anaconda3/envs/tensorf
low2_p38/lib/python3.8/site-packages (from requests->gtts) (2021.10.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/ubuntu/anaconda3/envs/tens
orflow2_p38/lib/python3.8/site-packages (from requests->gtts) (1.26.7)
Requirement already satisfied: idna<4,>=2.5 in /home/ubuntu/anaconda3/envs/tensorflow2_p
38/lib/python3.8/site-packages (from requests->gtts) (2.10)
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuron.amazonaws.com
```

```
Requirement already satisfied: playsound in /home/ubuntu/anaconda3/envs/tensorflow2_p38/
lib/python3.8/site-packages (1.3.0)
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuron.amazonaws.com
Requirement already satisfied: tensorflow in /home/ubuntu/anaconda3/envs/tensorflow2_p3
8/lib/python3.8/site-packages (2.7.0)
Requirement already satisfied: flatbuffers<3.0,>=1.12 in /home/ubuntu/anaconda3/envs/ten
sorflow2_p38/lib/python3.8/site-packages (from tensorflow) (2.0)
Requirement already satisfied: protobuf>=3.9.2 in /home/ubuntu/anaconda3/envs/tensorflow
2_p38/lib/python3.8/site-packages (from tensorflow) (3.19.1)
Requirement already satisfied: gast<0.5.0,>=0.2.1 in /home/ubuntu/anaconda3/envs/tensorf
low2_p38/lib/python3.8/site-packages (from tensorflow) (0.4.0)
Requirement already satisfied: keras<2.8,>=2.7.0rc0 in /home/ubuntu/anaconda3/envs/tenso
rflow2_p38/lib/python3.8/site-packages (from tensorflow) (2.7.0)
Requirement already satisfied: google-pasta>=0.1.1 in /home/ubuntu/anaconda3/envs/tensor
flow2_p38/lib/python3.8/site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.21.0 in /home/ubuntu/anac
onda3/envs/tensorflow2_p38/lib/python3.8/site-packages (from tensorflow) (0.23.1)
Requirement already satisfied: h5py>=2.9.0 in /home/ubuntu/anaconda3/envs/tensorflow2_p3
8/lib/python3.8/site-packages (from tensorflow) (3.4.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /home/ubuntu/anaconda3/envs/tensorfl
ow2_p38/lib/python3.8/site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: six>=1.12.0 in /home/ubuntu/anaconda3/envs/tensorflow2_p3
8/lib/python3.8/site-packages (from tensorflow) (1.16.0)
Requirement already satisfied: wrapt>=1.11.0 in /home/ubuntu/anaconda3/envs/tensorflow2_
p38/lib/python3.8/site-packages (from tensorflow) (1.12.1)
Requirement already satisfied: tensorboard~=2.6 in /home/ubuntu/anaconda3/envs/tensorflo
w2_p38/lib/python3.8/site-packages (from tensorflow) (2.8.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /home/ubuntu/anaconda3/envs/tensor
flow2_p38/lib/python3.8/site-packages (from tensorflow) (1.41.0)
Requirement already satisfied: tensorflow-estimator<2.8,~=2.7.0rc0 in /home/ubuntu/anaco
nda3/envs/tensorflow2_p38/lib/python3.8/site-packages (from tensorflow) (2.7.0)
Requirement already satisfied: termcolor>=1.1.0 in /home/ubuntu/anaconda3/envs/tensorflo
w2_p38/lib/python3.8/site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in /home/ubuntu/anaconda3/env
s/tensorflow2_p38/lib/python3.8/site-packages (from tensorflow) (1.1.2)
Requirement already satisfied: astunparse>=1.6.0 in /home/ubuntu/anaconda3/envs/tensorfl
ow2_p38/lib/python3.8/site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: typing-extensions>=3.6.6 in /home/ubuntu/anaconda3/envs/t
ensorflow2_p38/lib/python3.8/site-packages (from tensorflow) (4.3.0)
Requirement already satisfied: numpy>=1.14.5 in /home/ubuntu/anaconda3/envs/tensorflow2_
p38/lib/python3.8/site-packages (from tensorflow) (1.18.5)
Requirement already satisfied: absl-py>=0.4.0 in /home/ubuntu/anaconda3/envs/tensorflow2
_p38/lib/python3.8/site-packages (from tensorflow) (1.0.0)
Requirement already satisfied: wheel<1.0,>=0.32.0 in /home/ubuntu/anaconda3/envs/tensorf
low2_p38/lib/python3.8/site-packages (from tensorflow) (0.37.0)
Requirement already satisfied: libclang>=9.0.1 in /home/ubuntu/anaconda3/envs/tensorflow
2_p38/lib/python3.8/site-packages (from tensorflow) (12.0.0)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /home/ubuntu/ana
conda3/envs/tensorflow2_p38/lib/python3.8/site-packages (from tensorboard~=2.6->tensorfl
ow) (0.6.1)
Requirement already satisfied: requests<3,>=2.21.0 in /home/ubuntu/anaconda3/envs/tensor
flow2_p38/lib/python3.8/site-packages (from tensorboard~=2.6->tensorflow) (2.26.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in /home/ubuntu/anaconda3/envs/tens
orflow2_p38/lib/python3.8/site-packages (from tensorboard~=2.6->tensorflow) (2.5.0)
Requirement already satisfied: werkzeug>=0.11.15 in /home/ubuntu/anaconda3/envs/tensorfl
ow2_p38/lib/python3.8/site-packages (from tensorboard~=2.6->tensorflow) (2.0.2)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /home/ubuntu/anaconda
3/envs/tensorflow2_p38/lib/python3.8/site-packages (from tensorboard~=2.6->tensorflow)
(0.4.6)
Requirement already satisfied: markdown>=2.6.8 in /home/ubuntu/anaconda3/envs/tensorflow
2_p38/lib/python3.8/site-packages (from tensorboard~=2.6->tensorflow) (3.3.6)
Requirement already satisfied: setuptools>=41.0.0 in /home/ubuntu/anaconda3/envs/tensorf
low2_p38/lib/python3.8/site-packages (from tensorboard~=2.6->tensorflow) (59.1.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /home/ubuntu/anaconda3/e
nvs/tensorflow2_p38/lib/python3.8/site-packages (from tensorboard~=2.6->tensorflow) (1.
```

```
8.1)
Requirement already satisfied: rsa<5,>=3.1.4 in /home/ubuntu/anaconda3/envs/tensorflow2_
p38/lib/python3.8/site-packages (from google-auth<3,>=1.6.3->tensorboard~=2.6->tensorflo
w) (4.7.2)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /home/ubuntu/anaconda3/envs/ten
sorflow2_p38/lib/python3.8/site-packages (from google-auth<3,>=1.6.3->tensorboard~=2.6->
tensorflow) (5.0.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /home/ubuntu/anaconda3/envs/tens
orflow2_p38/lib/python3.8/site-packages (from google-auth<3,>=1.6.3->tensorboard~=2.6->t
ensorflow) (0.2.8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /home/ubuntu/anaconda3/envs/t
ensorflow2_p38/lib/python3.8/site-packages (from google-auth-oauthlib<0.5,>=0.4.1->tenso
rboard~=2.6->tensorflow) (1.3.0)
Requirement already satisfied: importlib-metadata>=4.4 in /home/ubuntu/anaconda3/envs/te
nsorflow2_p38/lib/python3.8/site-packages (from markdown>=2.6.8->tensorboard~=2.6->tenso
rflow) (4.8.2)
Requirement already satisfied: charset-normalizer~=2.0.0 in /home/ubuntu/anaconda3/envs/
tensorflow2_p38/lib/python3.8/site-packages (from requests<3,>=2.21.0->tensorboard~=2.6-
>tensorflow) (2.0.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/ubuntu/anaconda3/envs/tens
orflow2_p38/lib/python3.8/site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->ten
sorflow) (1.26.7)
Requirement already satisfied: certifi>=2017.4.17 in /home/ubuntu/anaconda3/envs/tensorf
low2_p38/lib/python3.8/site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensor
flow) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in /home/ubuntu/anaconda3/envs/tensorflow2_p
38/lib/python3.8/site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow)
(2.10)
Requirement already satisfied: zipp>=0.5 in /home/ubuntu/anaconda3/envs/tensorflow2_p38/
lib/python3.8/site-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard~
=2.6->tensorflow) (3.6.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /home/ubuntu/anaconda3/envs/tenso
rflow2_p38/lib/python3.8/site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.
3->tensorboard~=2.6->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in /home/ubuntu/anaconda3/envs/tensorflow
2_p38/lib/python3.8/site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<
0.5,>=0.4.1->tensorboard~=2.6->tensorflow) (3.1.1)
```

In [3]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter

import tensorflow as tf
import keras
from keras.preprocessing.image import load_img
import string
import time
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer

from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.models import Model
from tensorflow.keras import layers
from tensorflow.keras import activations
from tensorflow.keras import Input
from PIL import Image
#used for creating Progress Meters or Progress Bars
from tqdm import tqdm
```

In [4]:
```
import glob
from gtts import gTTS
from playsound import playsound
from IPython import display
import collections
import wordcloud
from wordcloud import WordCloud, STOPWORDS
```

```
playsound is relying on another python subprocess. Please use `pip install pygobject` if
you want playsound to run more efficiently.
```

In [5]:
```
import numpy as np
import pandas as pd
import os
```

In [6]:
```
!pwd
```

```
/home/ubuntu/trainContent
```

Data Understanding and Visualization

- Import the dataset and read image and captions into two seperate variables
- Visualise both the images & text present in the dataset
- Visualise the top 30 occuring words in the captions
  li>Visualise the top 30 occuring words in the captions
- Create a list which contains all the captions & path

In [7]:
```
images='flickr30k_images/flickr30k_images'
all_imgs = glob.glob(images + '/*.jpg',recursive=True)
print("The total images present in the dataset: {}".format(len(all_imgs)))
```

```
The total images present in the dataset: 31783
```

In [8]:
```
# import imageio
# #Visualising first 5 images :
# Display_Images = all_imgs[0:5]
# figure, axes = plt.subplots(1,5)
# figure.set_figwidth(20)
# for ax, image in zip(axes, Display_Images):
#   ax.imshow(imageio.imread(image), cmap=None)
```

In [9]:
```
# # view a random image
# import random
# Image.open(all_imgs[random.randrange(40, 60, 3)])
```

In [10]:
```
# text_file = 'flickr30k_images/captions.txt'
```

```
# def load_doc(filename):
#     open_file = open(text_file, 'r', encoding='latin-1' )
#     text = open_file.read()
#     open_file.close()
#     return text
# doc = load_doc(text_file)
# print(doc[:300])
```

In [11]:

```
img_path = 'flickr30k_images/flickr30k_images/'

all_img_id = []
all_img_vector = []
annotations = []

with open('flickr30k_images/captions.txt' , 'r') as fo:
  next(fo)
  for line in fo :
    split_arr = line.split(',')
    all_img_id.append(split_arr[0])
    annotations.append(split_arr[1].rstrip('\n.')) #removing out the \n.
    all_img_vector.append(img_path+split_arr[0])

df = pd.DataFrame(list(zip(all_img_id, all_img_vector,annotations)),columns =['ID','Pat

df
```

Out[11]:

	ID	Path	Captions
0	1000092795.jpg	flickr30k_images/flickr30k_images/1000092795.jpg	Two young guys with shaggy hair look at their ...
1	1000092795.jpg	flickr30k_images/flickr30k_images/1000092795.jpg	Two young White males are outside near many b...
2	1000092795.jpg	flickr30k_images/flickr30k_images/1000092795.jpg	Two men in green shirts are standing in a yard
3	1000092795.jpg	flickr30k_images/flickr30k_images/1000092795.jpg	A man in a blue shirt standing in a garden
4	1000092795.jpg	flickr30k_images/flickr30k_images/1000092795.jpg	Two friends enjoy time spent together
...	...	...	...
158910	998845445.jpg	flickr30k_images/flickr30k_images/998845445.jpg	A man in shorts and a Hawaiian shirt leans ove...
158911	998845445.jpg	flickr30k_images/flickr30k_images/998845445.jpg	A young man hanging over the side of a boat w...
158912	998845445.jpg	flickr30k_images/flickr30k_images/998845445.jpg	A man is leaning off of the side of a blue and...
158913	998845445.jpg	flickr30k_images/flickr30k_images/998845445.jpg	A man riding a small boat in a harbor with fo...
158914	998845445.jpg	flickr30k_images/flickr30k_images/998845445.jpg	A man on a moored blue and white boat with hil...

158915 rows × 3 columns

In [12]:
```
# #check total captions and images present in dataset
# print("Total captions present in the dataset: "+ str(len(annotations)))
# print("Total images present in the dataset: " + str(len(all_imgs)))
```

In [13]:
```
#Create the vocabulary & the counter for the captions
vocabulary = [word.lower() for line in annotations for word in line.split()]
val_count = Counter(vocabulary)
val_count
```

Out[13]:
```
Counter({'two': 21574,
         'young': 13212,
         'guys': 658,
         'with': 36207,
         'shaggy': 51,
         'hair': 2221,
         'look': 1328,
         'at': 16256,
         'their': 4037,
         'hands': 1556,
         'while': 11710,
         'hanging': 662,
         'out': 3432,
         'in': 83464,
         'the': 62963,
         'yard': 394,
         'white': 13178,
         'males': 248,
         'are': 20196,
         'outside': 4701,
         'near': 3016,
         'many': 1285,
         'bushes': 95,
         'men': 9497,
         'green': 5221,
         'shirts': 942,
         'standing': 9113,
         'a': 271141,
         'man': 42584,
         'blue': 11310,
         'shirt': 12981,
         'garden': 247,
         'friends': 398,
         'enjoy': 204,
         'time': 314,
         'spent': 2,
         'together': 1783,
         'several': 2096,
         'hard': 645,
         'hats': 836,
         'operating': 107,
         'giant': 204,
         'pulley': 18,
         'system': 35,
         'workers': 1104,
         'down': 7678,
         'from': 4041,
         'up': 5114,
         'above': 678,
         'on': 45666,
         'piece': 723,
```

```
'of': 38776,
'equipment': 471,
'working': 2175,
'machine': 521,
'wearing': 15709,
'four': 2202,
'top': 2415,
'tall': 495,
'structure': 370,
'three': 5749,
'large': 4752,
'rig': 12,
'child': 5078,
'pink': 2861,
'dress': 2130,
'is': 41116,
'climbing': 983,
'set': 596,
'stairs': 503,
'an': 15855,
'entry': 10,
'way': 276,
'little': 4793,
'girl': 9654,
'going': 612,
'into': 3731,
'wooden': 1066,
'cabin': 21,
'to': 17607,
'her': 7414,
'playhouse': 19,
'building': 3360,
'someone': 814,
'and': 44263,
'hat': 4037,
'stair': 31,
'leaning': 591,
'against': 1053,
'window': 1037,
'ladder': 328,
'cleaning': 399,
'cleans': 77,
'jeans': 1918,
'windows': 195,
'one': 6483,
'gray': 2086,
'black': 12309,
'stove': 97,
'guy': 1473,
'cooking': 467,
'joking': 8,
'around': 3134,
'camera': 2410,
'kitchen': 772,
'food': 1755,
'preparing': 566,
'meal': 283,
'people': 17316,
'photo': 594,
'playing': 8620,
'guitar': 1994,
'other': 3986,
'poking': 20,
'him': 2022,
'holds': 1777,
```

```
 'observes': 66,
 'his': 11495,
 'fixing': 216,
 'players': 1114,
 'costume': 551,
 'stitching': 4,
 'another': 3372,
 "'s": 2289,
 'coat': 1310,
 'boys': 1975,
 'sits': 2612,
 'chair': 1028,
 'holding': 6985,
 'stuffed': 185,
 'animal': 285,
 'lion': 41,
 'sitting': 9620,
 'completes': 15,
 'finishing': 41,
 'touches': 65,
 'toy': 1084,
 'smiling': 1952,
 'rollerskates': 59,
 'talking': 2072,
 'cellphone': 779,
 'parking': 325,
 'lot': 577,
 'trendy': 11,
 'gliding': 28,
 'slowly': 18,
 'street': 7998,
 'adult': 496,
 'rollerblades': 76,
 'cellular': 17,
 'phone': 608,
 'ear': 138,
 'there': 1634,
 'skating': 225,
 'woman': 22205,
 'asian': 2371,
 'suit': 1317,
 'stands': 2935,
 'dark-haired': 288,
 'brown-haired': 133,
 'pipes': 37,
 'metal': 652,
 'railing': 329,
 'walks': 1988,
 'past': 959,
 'dressed': 3460,
 'hip': 31,
 'outfits': 285,
 'purse': 301,
 'walking': 7339,
 'by': 5771,
 'gate': 142,
 'germany': 5,
 'jumping': 2735,
 'over': 3684,
 'rail': 279,
 'same': 297,
 'without': 141,
 'youths': 41,
 'roadside': 32,
 'night': 690,
```

```
'dancing': 1013,
'poles': 149,
'middle': 977,
'no': 377,
'five': 857,
'ballet': 97,
'dancers': 210,
'caught': 126,
'mid': 65,
'jump': 768,
'studio': 96,
'sunlight': 63,
'coming': 252,
'through': 4297,
'practice': 124,
'wonderful': 10,
'form': 78,
'girls': 2678,
'leaping': 230,
'simultaneously': 24,
'dance': 598,
'room': 1301,
'bending': 138,
'feet': 336,
'class': 250,
'sequence': 2,
'sneakers': 113,
'midair': 474,
'flight': 121,
'concrete': 489,
'casually': 65,
'stairway': 55,
'outdoors': 766,
'stone': 664,
'wall': 2324,
'behind': 2831,
'them': 1746,
'not': 198,
'staircase': 141,
'excited': 92,
'faces': 259,
'dog': 9090,
'brown': 4446,
'spots': 36,
'staring': 320,
'each': 1473,
'tri-colored': 14,
'road': 1770,
'dogs': 2354,
'different': 327,
'breeds': 5,
'looking': 5067,
'pavement': 162,
'moving': 203,
'toward': 456,
'spotted': 58,
'fighting': 263,
'reflective': 109,
'safety': 389,
'clothes': 749,
'protection': 36,
'drives': 156,
'john': 29,
'deere': 16,
'tractor': 142,
```

```
'tractors': 3,
'cruises': 3,
'driver': 136,
'wears': 279,
'easy': 7,
'see': 179,
'clothing': 1002,
'neon': 123,
'orange': 3086,
'uniform': 1158,
'driving': 402,
'headphones': 237,
'paved': 158,
'main': 40,
'country': 230,
'some': 3544,
'women': 5274,
'front': 7604,
'bus': 732,
'buildings': 429,
'it': 2112,
'stand': 2336,
'city': 2325,
'group': 7851,
'wait': 264,
'dark': 1390,
'glasses': 1616,
'putting': 462,
'powder': 20,
'cake': 254,
'using': 1159,
'sifter': 3,
'lady': 2043,
'sprinkling': 6,
'powdered': 5,
'sugar': 5,
'bundt': 2,
'sprinkles': 6,
'onto': 624,
'jacket': 3348,
'sifting': 8,
'chocolate': 64,
'pan': 101,
'small': 3779,
'grass': 2598,
'plays': 1907,
'fingerpaints': 3,
'canvas': 61,
'rainbow': 97,
'covered': 895,
'paint': 321,
'painted': 408,
'bowl': 286,
'pigtails': 65,
'painting': 677,
'sleeping': 641,
'bench': 1753,
'next': 3752,
'lays': 197,
'which': 552,
'also': 178,
'tied': 100,
'laying': 771,
'leash': 208,
'ground': 1431,
```

```
'shirtless': 556,
'lies': 139,
'park': 2041,
'adults': 632,
'inside': 926,
'home': 338,
'chairs': 527,
'arranged': 23,
'circle': 209,
'type': 215,
'musical': 283,
'instruments': 511,
'musicians': 261,
'practicing': 341,
'sheet': 136,
'music': 645,
'(using': 1,
'flutes': 10,
')': 123,
'living': 153,
'gathered': 612,
'talk': 303,
'about': 1034,
'favorite': 24,
'tunes': 10,
'both': 602,
'clarinets': 6,
'elderly': 1016,
'stringed': 49,
'instrument': 407,
'least': 34,
'instrumentalists': 2,
'play': 1890,
'curtained': 4,
'clearly': 9,
'bunch': 342,
'as': 5028,
'they': 958,
'read': 91,
'off': 2154,
'performing': 1342,
'person': 4478,
'alone': 253,
'has': 2198,
'broken': 77,
'roadway': 31,
'washed': 18,
'bridge': 607,
'supports': 9,
'surveys': 11,
'damage': 5,
't-shirt': 1327,
'looks': 2465,
'surrounded': 785,
'crowd': 2961,
'metro': 40,
'station': 400,
'entrance': 98,
'portrayed': 1,
'area': 1444,
'goatee': 32,
'latex': 15,
'gloves': 374,
'tattoo': 208,
'gun': 194,
```

```
'place': 283,
'back': 1550,
'upper': 39,
'giving': 392,
'getting': 979,
'children': 4453,
'boy': 9429,
'writing': 286,
'sit': 1584,
'seesaw': 23,
'sand': 1156,
'teeter': 10,
'totter': 10,
'2': 450,
'kids': 1193,
'vest': 745,
'intersection': 122,
'flag': 555,
'caution': 34,
'waving': 242,
'bright': 783,
'corner': 517,
'spray': 137,
'construction': 1323,
'worker': 683,
'red': 9916,
'pierced': 13,
'ears': 134,
'beer': 388,
'can': 278,
'crocheted': 5,
'gauges': 5,
'blitz': 1,
'starring': 20,
'something': 2123,
'long': 1313,
'beret': 49,
'beige': 204,
'raincoat': 40,
'marketplace': 123,
'scenery': 66,
'artists': 54,
'paintings': 39,
'busy': 1007,
'sidewalk': 2812,
'studying': 49,
'scene': 332,
'public': 516,
'others': 745,
'who': 1436,
'passerby': 17,
'admiring': 57,
'outdoor': 962,
'art': 379,
'fair': 181,
'foot': 235,
'waste': 5,
'basket': 310,
'pants': 1986,
'pushing': 455,
'cart': 767,
'janitor': 20,
'dolly': 13,
'tools': 118,
'grips': 9,
```

```
'ropes': 90,
'playground': 424,
'climbs': 291,
'rope': 593,
'roping': 24,
'net': 372,
'toothpick': 3,
'end': 182,
'traveling': 83,
'flower': 301,
'teen': 82,
'unknown': 35,
'object': 459,
'for': 4899,
'photographers': 53,
'train': 1046,
'origami': 2,
'crane': 97,
'you': 126,
'know': 13,
'i': 111,
'am': 20,
'like': 615,
'justin': 2,
'bieber': 1,
'blond': 1593,
'yellow': 4556,
'gazing': 47,
'pole': 604,
'vault': 14,
'determined': 18,
'sky': 447,
'baseball': 1454,
'cap': 1037,
'bathroom': 92,
'coffee': 320,
'mug': 43,
'ball': 3892,
'cup': 354,
'between': 545,
'urinals': 1,
'posing': 931,
'urinal': 4,
'multicolored': 207,
'background': 2882,
'sunset': 260,
'clear': 222,
'skies': 42,
'sun': 422,
'setting': 513,
'silhouettes': 10,
'decorate': 12,
'horizon': 43,
'sets': 114,
'restaurant': 948,
'glass': 508,
'table': 2682,
'drink': 509,
'drinking': 656,
'old': 1588,
'having': 969,
'running': 3369,
'grassy': 773,
'fence': 886,
'boston': 21,
```

```
'terrier': 35,
'lush': 38,
'runs': 1208,
'officer': 239,
'van': 181,
'trained': 7,
'police': 489,
'handler': 8,
'security': 92,
'watch': 1121,
'policeman': 105,
'german': 112,
'shepherd': 89,
'stops': 92,
'search': 14,
'"""the': 15,
'"""""': 1551,
'conditions': 15,
'snow': 2534,
'seem': 66,
'almost': 59,
'obliverate': 1,
'details': 9,
'cold': 169,
'weather': 99,
'heavy': 184,
'riding': 3313,
'bicycle': 1557,
'suburban': 37,
'neighborhood': 100,
'."""': 656,
'rides': 1062,
'bike': 2533,
'snowy': 670,
'uniformly': 1,
'tie': 307,
'slacks': 73,
'converse': 82,
'open': 761,
'5': 135,
'suits': 353,
'short-sleeved': 54,
'ties': 35,
'colleges': 1,
'stop': 312,
'take': 448,
'break': 240,
'business': 282,
'meetings': 1,
'tan': 875,
'backwards': 122,
'factory': 83,
'older': 2268,
'too': 48,
'photograph': 204,
'camp': 40,
'among': 255,
'machines': 59,
'works': 398,
'machinery': 126,
'work': 1028,
'caucasian': 120,
'dark-skinned': 138,
'sleeveless': 100,
'conveyor': 19,
```

```
'packing': 12,
'jars': 13,
'candles': 79,
'boxes': 132,
'sorting': 39,
'warehouse': 48,
'manager': 5,
'assisting': 22,
'employee': 46,
'assembly': 16,
'line': 854,
'outfit': 711,
'broom': 107,
'traditional': 231,
'architecture': 23,
'baby': 1667,
'bottoms': 47,
'remove': 14,
'dirt': 1409,
'kimono': 29,
'sweep': 20,
'sweeping': 192,
'walkway': 250,
'florescent': 10,
'vests': 347,
'parked': 347,
'cars': 391,
'converses': 14,
'seen': 222,
'leans': 307,
'car': 1767,
'watching': 1496,
'row': 200,
'waiting': 993,
'go': 238,
'checkpoint': 4,
'ranger': 7,
'tourist': 85,
'eating': 1240,
'snack': 78,
'toddlers': 62,
'infants': 10,
'female': 1397,
'babies': 45,
'chips': 36,
'silver': 299,
'mobile': 77,
'weird': 30,
'vehicle': 412,
'plaza': 114,
'segway': 18,
'showing': 365,
'product': 45,
'advertise': 3,
'modern': 57,
'?': 36,
'approaches': 54,
'strange': 86,
'containing': 50,
'onlookers': 342,
'observe': 81,
'roped': 11,
'barrier': 81,
'futuristic': 5,
'watches': 1081,
```

```
'single-person': 2,
'4-wheeled': 1,
'bride': 284,
'groom': 161,
'side': 1698,
'focus': 42,
'pathway': 65,
'brick': 838,
'beautiful': 484,
'new': 308,
'husband': 25,
'recently': 21,
'married': 87,
'couple': 1469,
'pose': 581,
'arm': 617,
'nintendo': 15,
'gamecube': 4,
'controller': 8,
'mcdonald': 40,
'video': 214,
'game': 1779,
'kiosk': 43,
'kid': 601,
'water': 5957,
'head': 1603,
'turned': 95,
'shore': 371,
'shaking': 129,
'shakes': 58,
'edge': 457,
'beach': 2904,
'its': 1142,
'picnicking': 2,
'picnic': 185,
'tables': 282,
'enjoying': 677,
'day': 1341,
'reunion': 10,
'full': 648,
'swing': 710,
'moon': 20,
'bounce': 39,
'barbecue': 122,
'or': 1124,
'spanish': 36,
'pass': 236,
'sunglasses': 1269,
'puts': 150,
'blouse': 170,
'traffic': 254,
'seated': 483,
'balloon': 280,
'design': 61,
'burger': 11,
'touching': 156,
'coolers': 11,
'gray-haired': 86,
'enjoys': 147,
'sandwich': 74,
'student': 106,
'downward': 23,
'kick': 249,
'board': 592,
'held': 231,
```

```
'karate': 231,
'instructor': 42,
'wood': 503,
'half': 135,
'kicking': 306,
'stick': 757,
'that': 2498,
'tae': 5,
'kwon': 5,
'do': 226,
'breaking': 33,
'boards': 56,
'demonstrating': 51,
'martial': 365,
'arts': 336,
'youngsters': 18,
'crouched': 67,
'mat': 159,
'during': 1042,
'competition': 291,
'jumps': 1338,
'students': 337,
'demonstration': 54,
'perform': 359,
'balconies': 15,
'pipe': 143,
'towards': 818,
'lower': 35,
'balcony': 162,
'liquid': 94,
'poured': 6,
'sweatshirt': 481,
'reaching': 201,
'hand': 1921,
'empties': 6,
'bottle': 362,
'unto': 5,
'below': 209,
'pouring': 166,
'contents': 36,
'shielding': 20,
'himself': 197,
'trying': 888,
'paper': 648,
'block': 226,
'shades': 61,
'eyes': 415,
'face': 1649,
'carrying': 1783,
'backs': 112,
'sunny': 553,
'nice': 163,
'walk': 1709,
'stroll': 68,
'rock': 1474,
'overalls': 155,
'smiles': 601,
'stony': 10,
'aross': 1,
'mottled': 3,
'collar': 235,
'fallen': 183,
'tree': 1513,
'leaps': 296,
'log': 158,
```

```
'jumped': 66,
'stump': 57,
'crossing': 423,
'placards': 3,
'gathering': 265,
'across': 1298,
'assembles': 9,
'gentleman': 389,
'hurrying': 9,
'long-sleeved': 137,
'bikes': 431,
'body': 751,
'river': 735,
'barefooted': 39,
'olive': 35,
'shorts': 2066,
'grilling': 110,
'hotdogs': 42,
'propane': 7,
'grill': 314,
'plastic': 495,
'sausages': 27,
'prepares': 339,
'he': 1409,
'grills': 29,
'surface': 158,
'field': 2981,
'scooter': 250,
'attracts': 5,
'attention': 100,
'those': 30,
'light': 848,
'cross': 348,
'skis': 148,
'framed': 46,
'pictures': 442,
'displaying': 98,
'skier': 296,
'trees': 968,
'artwork': 87,
'sale': 140,
'climbers': 38,
'seven': 169,
'ascending': 18,
'whilst': 173,
'belays': 3,
'collage': 6,
'cliff': 312,
'gymnast': 160,
'high': 796,
'air': 2375,
'balance': 102,
'beam': 119,
'leotard': 58,
'doing': 1522,
'gymnastics': 73,
'audience': 425,
'supple': 1,
'soars': 19,
'wheeler': 35,
'pool': 1509,
'atv': 82,
'rubber': 94,
'miniature': 47,
'lawn': 411,
```

```
'quad': 29,
'truck': 780,
'filled': 396,
'cotton': 49,
'substance': 33,
'pile': 326,
'wool': 27,
'loading': 79,
'load': 52,
'sheared': 2,
'headscarf': 90,
'scenic': 69,
'view': 454,
'bay': 44,
'pay': 51,
'binoculars': 63,
'viewing': 67,
'deck': 193,
'mounted': 38,
'telescope': 196,
'windbreaker': 22,
'though': 74,
'rooftop': 54,
'uses': 253,
'spaceship': 1,
'ticket': 38,
'stuck': 46,
'plane': 137,
'examining': 137,
'airplane': 167,
'nose': 200,
'away': 626,
'sprinkler': 79,
'chases': 124,
'hose': 160,
'happily': 119,
'supermarket': 45,
'daughter': 170,
'grocery': 204,
'store': 1093,
'stroller': 273,
'catch': 585,
'thrown': 136,
'nearby': 445,
'ready': 654,
'flying': 333,
'mouth': 1332,
'after': 599,
'colored': 708,
'covering': 175,
'part': 165,
'booth': 196,
'sweater': 814,
'get': 432,
'hiking': 247,
'forest': 471,
'where': 433,
'partially': 75,
'wilderness': 36,
'hikers': 124,
'resting': 252,
'patch': 71,
'facial': 46,
'hutch': 2,
'cabinet': 22,
```

```
'doors': 88,
'long-sleeve': 98,
'under': 1227,
'middle-aged': 364,
'beard': 460,
'handmade': 15,
'creation': 11,
'floor': 1075,
'reading': 1143,
'father': 247,
'grown': 18,
'son': 233,
'camping': 76,
'trip': 51,
'wild': 53,
'items': 297,
'bearded': 296,
'map': 102,
'traveler': 10,
'decipher': 1,
'lake': 796,
'lone': 177,
'duck': 73,
'swimming': 964,
'waves': 323,
'facing': 344,
'skyline': 44,
'waters': 131,
'big': 837,
'infant': 160,
'being': 1373,
'male': 1473,
'pond': 200,
'care': 45,
'along': 1870,
'newborn': 101,
'shelter': 30,
'bicycles': 285,
'curved': 30,
'roof': 364,
'tonight': 4,
'questioning': 3,
'wagon': 123,
'nighttime': 71,
'dome': 12,
'beside': 807,
'lab': 201,
'tags': 42,
'frolicks': 2,
'this': 1044,
'splashing': 181,
'surf': 131,
'splashes': 96,
'drilling': 38,
'frozen': 59,
'ice': 842,
'hole': 218,
'fishing': 549,
'making': 940,
'turn': 216,
'soft': 33,
'sandy': 250,
'climber': 159,
'scaling': 44,
'picks': 47,
```

```
 'crampons': 2,
 'scale': 23,
 'waterfall': 149,
 'meadow': 45,
 'ocean': 1045,
 'path': 770,
 'attire': 305,
 'shovels': 82,
 'disregarding': 2,
 'all': 1061,
 'shoveling': 133,
 'roads': 6,
 'sidewalks': 13,
 'slush': 4,
 'wedding': 324,
 'flowers': 588,
 'reception': 44,
 'smile': 372,
 'cutting': 389,
 'snowmobile': 21,
 'riders': 141,
 'helmets': 265,
 'goggles': 345,
 'clearing': 49,
 'snowmobiles': 6,
 ';': 241,
 'left': 573,
 'right': 618,
 'winter': 422,
 'jackets': 359,
 'ski': 252,
 'gather': 319,
 'snowmobiling': 3,
 'carries': 371,
 'wet': 359,
 'item': 99,
 'rows': 75,
 'vegetables': 269,
 'sell': 96,
 ...})
```

In [14]:

```
#Visualise the top 30 occuring words in the captions
for word, count in val_count.most_common(30):
  print(word, ": ", count)

lst = val_count.most_common(30)
# most_common_words_df = pd.DataFrame(lst, columns = ['Word', 'Count'])
# most_common_words_df.plot.bar(x='Word', y='Count', width=0.6, color='orange', figsize
# plt.title("Top 30 maximum frequency words", fontsize = 18, color= 'navy')
# plt.xlabel("Words", fontsize = 14, color= 'navy')
# plt.ylabel("Count", fontsize = 14, color= 'navy')
```

```
a :  271141
in :  83464
the :  62963
on :  45666
and :  44263
man :  42584
is :  41116
of :  38776
with :  36207
woman :  22205
two :  21574
```

```
are :  20196
to :  17607
people :  17316
at :  16256
an :  15855
wearing :  15709
young :  13212
white :  13178
shirt :  12981
black :  12309
while :  11710
his :  11495
blue :  11310
red :  9916
girl :  9654
sitting :  9620
men :  9497
boy :  9429
standing :  9113
```

Clearly STOPWORDS form a clear majority

Stop words are a set of commonly used words in a language.

```
Examples of stop words in English are “a”, “the”, “is”, “are” and etc.
```

*Stop words are commonly used in **Text Mining and Natural Language Processing (NLP)** to eliminate words that are so commonly used that they carry very little useful information.*

In [15]:
```
# wordcloud = WordCloud(width = 1000, height = 500).generate_from_frequencies(val_count
# plt.figure(figsize = (12, 12))
# plt.imshow(wordcloud)
```

In [16]:
```
# def caption_with_img_plot(image_id, frame) :
#   capt = ("\n" *2).join(frame[frame['ID'] == image_id].Captions.to_list())
#   fig, ax = plt.subplots()
#   ax.set_axis_off()
#   idx = df.ID.to_list().index(image_id)
#   im =  Image.open(df.Path.iloc[idx])
#   w, h = im.size[0], im.size[-1]
#   ax.imshow(im)
#   ax.text(w+50, h, capt, fontsize = 18, color = 'navy')
# caption_with_img_plot(df.ID.iloc[8049], df)
```

Clean Data

- Remove punctuations
- Convert captions to lowercase
- Visualise the top 30 occuring words in the captions

- Visualise the top 30 occuring words in the captions
- Create a list which contains all the captions & path

In [17]:
```
#data cleaning
rem_punct = str.maketrans('', '', string.punctuation)
for r in range(len(annotations)) :
  line = annotations[r]
  line = line.split()

  # converting to lowercase
  line = [word.lower() for word in line]

  # remove punctuation from each caption and hanging letters
  line = [word.translate(rem_punct) for word in line]
  line = [word for word in line if len(word) > 1]

  # remove numeric values
  line = [word for word in line if word.isalpha()]

  annotations[r] = ' '.join(line)
```

In [18]:
```
#add the <start> & <end> token to all those captions as well
annotations = ['<start>' + ' ' + line + ' ' + '<end>' for line in annotations]

#Create a list which contains all the path to the images
all_img_path = all_img_vector
```

In [19]:
```
# ##list contatining captions for an image
# annotations[0:5]
```

Data Preprocessing

- Create the tokenized vectors by tokenizing the captions fore ex :split them using spaces & other filters./p>
- This gives us a vocabulary of all of the unique words in the data. Keep the total vocaublary to top 5,000 words for saving memory.
- Replace all other words with the unknown token "UNK"
- Create word-to-index and index-to-word mappings.
- Pad all sequences to be the same length as the longest one

In [20]:
```
# Creating the tokenizer
top_word_cnt = 5000
tokenizer = Tokenizer(num_words = top_word_cnt+1, filters= '!"#$%^&*()_+.,:;-?/~`{}[]|\
```

```
                        lower = True, char_level = False,
                        oov_token = 'UNK')
```

In [21]:
```
# Creating word-to-index and index-to-word mappings.

tokenizer.fit_on_texts(annotations)
#transform each text into a sequence of integers
train_seqs = tokenizer.texts_to_sequences(annotations)
```

In [22]:
```
# We add PAD token for zero
tokenizer.word_index['PAD'] = 0
tokenizer.index_word[0] = 'PAD'
```

In [23]:
```
# print(tokenizer.oov_token)
# print(tokenizer.index_word[0])
```

In [24]:
```
# tokenizer.index_word
```

In [25]:
```
# Creating a word count for our tokenizer to visualize the Top 30 occuring words after

tokenizer_top_words = [word for line in annotations for word in line.split() ]

#tokenizer_top_words_count
tokenizer_top_words_count = collections.Counter(tokenizer_top_words)
```

In [26]:
```
# for word, count in tokenizer_top_words_count.most_common(30) :
#   print(word, ": ", count)

# tokens = tokenizer_top_words_count.most_common(30)
# most_com_words_df = pd.DataFrame(tokens, columns = ['Word', 'Count'])

# #plot 30 most common words
# most_common_words_df.plot.bar(x = 'Word', y= 'Count', width=0.8, color = 'indigo', fi
# plt.title('Top 30 common words', fontsize =20, color= 'navy')
# plt.xlabel('Words', fontsize =14, color= 'navy')
# plt.ylabel('Counts', fontsize =14, color= 'navy')
# plt.grid(b=None)
```

In [27]:
```
# wordcloud_token = WordCloud(width = 1000, height = 500).generate_from_frequencies(tok
# plt.figure(figsize = (12, 8))
# plt.imshow(wordcloud_token)
# plt.grid(b = None)
```

In [28]:
```
# Pad each vector to the max_length of the captions  store it to a vairable

train_seqs_len = [len(seq) for seq in train_seqs]

longest_word_length = max(train_seqs_len)
```

```
cap_vector= tf.keras.preprocessing.sequence.pad_sequences(train_seqs, padding= 'post',
                                                          dtype='int32', value=0)
print("The shape of Caption vector is :" + str(cap_vector.shape))
```

```
The shape of Caption vector is :(158915, 74)
```

In [29]:
```
# creating list to store preprocessed images and setting up the Image Shape

preprocessed_image = []
IMAGE_SHAPE = (299, 299)
```

In [30]:
```
#checking image format

tf.keras.backend.image_data_format()
```

Out[30]:
```
'channels_last'
```

In [31]:
```
# for img in all_imgs[0:5] :
#     img = tf.io.read_file(img, name=None)
#     img = tf.image.decode_jpeg(img, channels=0)
#     img = tf.image.resize(img, (299, 299))
#     img = tf.keras.applications.inception_v3.preprocess_input(img)
#     preprocessed_image.append(img)
```

In [32]:
```
# # checking first five images post preprocessing
# Display_Images = preprocessed_image[0:5]
# figure, axes = plt.subplots(1,5)
# figure.set_figwidth(25)
# for ax, image in zip(axes, Display_Images) :
#   print('Shape after resize : ', image.shape)
#   ax.imshow(image)
#   ax.grid('off')
```

Dataset creation

- Create a function which maps the image path to their feature
- Create a builder function to create train & test dataset & apply the function created earlier to transform the dataset
- Apply train_test_split on both image path & captions to create the train & test list. Create the train-test spliit using 80-20 ratio & random state = 42
- Make sure you have done Shuffle and batch while building the dataset
- The shape of each image in the dataset after building should be (batch_size, 8*8, 2048)

In [33]:
```
def load_images(image_path) :
  img = tf.io.read_file(image_path, name = None)
```

```
    img = tf.image.decode_jpeg(img, channels=0)
    img = tf.image.resize(img, IMAGE_SHAPE)
    img = tf.keras.applications.inception_v3.preprocess_input(img)
    return img, image_path
```

In [34]:
```
# all_img_vector
```

In [35]:
```
# Map each image full path to the function, in order to preprocess the image
training_list = sorted(set(all_img_vector))
New_Img = tf.data.Dataset.from_tensor_slices(training_list)
New_Img = New_Img.map(load_images, num_parallel_calls = tf.data.experimental.AUTOTUNE)
New_Img = New_Img.batch(64, drop_remainder=False)
```

In [36]:
```
#Ratio = 80:20 and we will set random state = 42
path_train, path_test, caption_train, caption_test = train_test_split(all_img_vector, c
```

In [37]:
```
# print("Training data for images: " + str(len(path_train)))
# print("Testing data for images: " + str(len(path_test)))
# print("Training data for Captions: " + str(len(caption_train)))
# print("Testing data for Captions: " + str(len(caption_test)))
```

Load the pretrained Imagenet weights of Inception net V3

- To save the memory(RAM) from getting exhausted, extract the features of the images using the last layer of pre-trained model. Including this as part of training will lead to higher computational time.
- The shape of the output of this layer is 8x8x2048.
- Use a function to extract the features of each image in the train & test dataset such that the shape of each image should be (batch_size, 8*8, 2048)

In [38]:
```
image_model = tf.keras.applications.InceptionV3(include_top=False,weights='imagenet')
new_input = image_model.input
hidden_layer = image_model.layers[-1].output
image_features_extract_model = tf.compat.v1.keras.Model(new_input, hidden_layer)
```

In [39]:
```
# Once the features are created, you need to reshape them such that feature shape is in
image_features_extract_model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
 input_1 (InputLayer)           [(None, None, None,  0           []
                                 3)]

 conv2d (Conv2D)                (None, None, None,   864         ['input_1[0][0]']
                                32)

 batch_normalization (BatchNorm  (None, None, None,   96         ['conv2d[0][0]']
 alization)                     32)

 activation (Activation)        (None, None, None,   0           ['batch_normalization
[0][0]']
                                32)

 conv2d_1 (Conv2D)              (None, None, None,   9216        ['activation[0][0]']
                                32)

 batch_normalization_1 (BatchNo  (None, None, None,   96         ['conv2d_1[0][0]']
 rmalization)                   32)

 activation_1 (Activation)      (None, None, None,   0           ['batch_normalization_1
[0][0]']
                                32)

 conv2d_2 (Conv2D)              (None, None, None,   18432       ['activation_1[0][0]']
                                64)

 batch_normalization_2 (BatchNo  (None, None, None,   192        ['conv2d_2[0][0]']
 rmalization)                   64)

 activation_2 (Activation)      (None, None, None,   0           ['batch_normalization_2
[0][0]']
                                64)

 max_pooling2d (MaxPooling2D)   (None, None, None,   0           ['activation_2[0][0]']
                                64)

 conv2d_3 (Conv2D)              (None, None, None,   5120        ['max_pooling2d[0][0]']
                                80)

 batch_normalization_3 (BatchNo  (None, None, None,   240        ['conv2d_3[0][0]']
 rmalization)                   80)

 activation_3 (Activation)      (None, None, None,   0           ['batch_normalization_3
[0][0]']
                                80)

 conv2d_4 (Conv2D)              (None, None, None,   138240      ['activation_3[0][0]']
                                192)

 batch_normalization_4 (BatchNo  (None, None, None,   576        ['conv2d_4[0][0]']
 rmalization)                   192)

 activation_4 (Activation)      (None, None, None,   0           ['batch_normalization_4
[0][0]']
                                192)

 max_pooling2d_1 (MaxPooling2D)  (None, None, None,   0          ['activation_4[0][0]']
                                192)

 conv2d_8 (Conv2D)              (None, None, None,   12288       ['max_pooling2d_1[0]
[0]']
                                64)

 batch_normalization_8 (BatchNo  (None, None, None,   192        ['conv2d_8[0][0]']
 rmalization)                   64)
```

```
 activation_8 (Activation)      (None, None, None,   0           ['batch_normalization_8
[0][0]']
                                64)

 conv2d_6 (Conv2D)              (None, None, None,   9216        ['max_pooling2d_1[0]
[0]']
                                48)

 conv2d_9 (Conv2D)              (None, None, None,   55296       ['activation_8[0][0]']
                                96)

 batch_normalization_6 (BatchNo  (None, None, None,   144        ['conv2d_6[0][0]']
 rmalization)                   48)

 batch_normalization_9 (BatchNo  (None, None, None,   288        ['conv2d_9[0][0]']
 rmalization)                   96)

 activation_6 (Activation)      (None, None, None,   0           ['batch_normalization_6
[0][0]']
                                48)

 activation_9 (Activation)      (None, None, None,   0           ['batch_normalization_9
[0][0]']
                                96)

 average_pooling2d (AveragePool  (None, None, None,   0          ['max_pooling2d_1[0]
[0]']
 ing2D)                         192)

 conv2d_5 (Conv2D)              (None, None, None,   12288       ['max_pooling2d_1[0]
[0]']
                                64)

 conv2d_7 (Conv2D)              (None, None, None,   76800       ['activation_6[0][0]']
                                64)

 conv2d_10 (Conv2D)             (None, None, None,   82944       ['activation_9[0][0]']
                                96)

 conv2d_11 (Conv2D)             (None, None, None,   6144        ['average_pooling2d[0]
[0]']
                                32)

 batch_normalization_5 (BatchNo  (None, None, None,   192        ['conv2d_5[0][0]']
 rmalization)                   64)

 batch_normalization_7 (BatchNo  (None, None, None,   192        ['conv2d_7[0][0]']
 rmalization)                   64)

 batch_normalization_10 (BatchN  (None, None, None,   288        ['conv2d_10[0][0]']
 ormalization)                  96)

 batch_normalization_11 (BatchN  (None, None, None,   96         ['conv2d_11[0][0]']
 ormalization)                  32)

 activation_5 (Activation)      (None, None, None,   0           ['batch_normalization_5
[0][0]']
                                64)

 activation_7 (Activation)      (None, None, None,   0           ['batch_normalization_7
[0][0]']
                                64)

 activation_10 (Activation)     (None, None, None,   0           ['batch_normalization_1
```

```
0[0][0]']
                                96)

 activation_11 (Activation)     (None, None, None,   0           ['batch_normalization_1
1[0][0]']
                                32)

 mixed0 (Concatenate)           (None, None, None,   0           ['activation_5[0][0]',
                                256)                             'activation_7[0][0]',
                                                                 'activation_10[0][0]',
                                                                 'activation_11[0][0]']

 conv2d_15 (Conv2D)             (None, None, None,   16384       ['mixed0[0][0]']
                                64)

 batch_normalization_15 (BatchN  (None, None, None,   192        ['conv2d_15[0][0]']
 ormalization)                  64)

 activation_15 (Activation)     (None, None, None,   0           ['batch_normalization_1
5[0][0]']
                                64)

 conv2d_13 (Conv2D)             (None, None, None,   12288       ['mixed0[0][0]']
                                48)

 conv2d_16 (Conv2D)             (None, None, None,   55296       ['activation_15[0][0]']
                                96)

 batch_normalization_13 (BatchN  (None, None, None,   144        ['conv2d_13[0][0]']
 ormalization)                  48)

 batch_normalization_16 (BatchN  (None, None, None,   288        ['conv2d_16[0][0]']
 ormalization)                  96)

 activation_13 (Activation)     (None, None, None,   0           ['batch_normalization_1
3[0][0]']
                                48)

 activation_16 (Activation)     (None, None, None,   0           ['batch_normalization_1
6[0][0]']
                                96)

 average_pooling2d_1 (AveragePo  (None, None, None,   0          ['mixed0[0][0]']
 oling2D)                       256)

 conv2d_12 (Conv2D)             (None, None, None,   16384       ['mixed0[0][0]']
                                64)

 conv2d_14 (Conv2D)             (None, None, None,   76800       ['activation_13[0][0]']
                                64)

 conv2d_17 (Conv2D)             (None, None, None,   82944       ['activation_16[0][0]']
                                96)

 conv2d_18 (Conv2D)             (None, None, None,   16384       ['average_pooling2d_1
[0][0]']
                                64)

 batch_normalization_12 (BatchN  (None, None, None,   192        ['conv2d_12[0][0]']
 ormalization)                  64)

 batch_normalization_14 (BatchN  (None, None, None,   192        ['conv2d_14[0][0]']
 ormalization)                  64)

 batch_normalization_17 (BatchN  (None, None, None,   288        ['conv2d_17[0][0]']
```

```
 ormalization)                  96)

 batch_normalization_18 (BatchN  (None, None, None,   192        ['conv2d_18[0][0]']
 ormalization)                  64)

 activation_12 (Activation)     (None, None, None,   0          ['batch_normalization_1
2[0][0]']
                                64)

 activation_14 (Activation)     (None, None, None,   0          ['batch_normalization_1
4[0][0]']
                                64)

 activation_17 (Activation)     (None, None, None,   0          ['batch_normalization_1
7[0][0]']
                                96)

 activation_18 (Activation)     (None, None, None,   0          ['batch_normalization_1
8[0][0]']
                                64)

 mixed1 (Concatenate)           (None, None, None,   0          ['activation_12[0][0]',
                                288)                             'activation_14[0][0]',
                                                                 'activation_17[0][0]',
                                                                 'activation_18[0][0]']

 conv2d_22 (Conv2D)             (None, None, None,   18432      ['mixed1[0][0]']
                                64)

 batch_normalization_22 (BatchN  (None, None, None,   192        ['conv2d_22[0][0]']
 ormalization)                  64)

 activation_22 (Activation)     (None, None, None,   0          ['batch_normalization_2
2[0][0]']
                                64)

 conv2d_20 (Conv2D)             (None, None, None,   13824      ['mixed1[0][0]']
                                48)

 conv2d_23 (Conv2D)             (None, None, None,   55296      ['activation_22[0][0]']
                                96)

 batch_normalization_20 (BatchN  (None, None, None,   144        ['conv2d_20[0][0]']
 ormalization)                  48)

 batch_normalization_23 (BatchN  (None, None, None,   288        ['conv2d_23[0][0]']
 ormalization)                  96)

 activation_20 (Activation)     (None, None, None,   0          ['batch_normalization_2
0[0][0]']
                                48)

 activation_23 (Activation)     (None, None, None,   0          ['batch_normalization_2
3[0][0]']
                                96)

 average_pooling2d_2 (AveragePo  (None, None, None,   0          ['mixed1[0][0]']
 oling2D)                       288)

 conv2d_19 (Conv2D)             (None, None, None,   18432      ['mixed1[0][0]']
                                64)

 conv2d_21 (Conv2D)             (None, None, None,   76800      ['activation_20[0][0]']
                                64)
```

```
 conv2d_24 (Conv2D)             (None, None, None,   82944       ['activation_23[0][0]']
                                96)

 conv2d_25 (Conv2D)             (None, None, None,   18432       ['average_pooling2d_2
[0][0]']
                                64)

 batch_normalization_19 (BatchN  (None, None, None,   192        ['conv2d_19[0][0]']
 ormalization)                  64)

 batch_normalization_21 (BatchN  (None, None, None,   192        ['conv2d_21[0][0]']
 ormalization)                  64)

 batch_normalization_24 (BatchN  (None, None, None,   288        ['conv2d_24[0][0]']
 ormalization)                  96)

 batch_normalization_25 (BatchN  (None, None, None,   192        ['conv2d_25[0][0]']
 ormalization)                  64)

 activation_19 (Activation)     (None, None, None,   0           ['batch_normalization_1
9[0][0]']
                                64)

 activation_21 (Activation)     (None, None, None,   0           ['batch_normalization_2
1[0][0]']
                                64)

 activation_24 (Activation)     (None, None, None,   0           ['batch_normalization_2
4[0][0]']
                                96)

 activation_25 (Activation)     (None, None, None,   0           ['batch_normalization_2
5[0][0]']
                                64)

 mixed2 (Concatenate)           (None, None, None,   0           ['activation_19[0][0]',
                                288)                             'activation_21[0][0]',
                                                                 'activation_24[0][0]',
                                                                 'activation_25[0][0]']

 conv2d_27 (Conv2D)             (None, None, None,   18432       ['mixed2[0][0]']
                                64)

 batch_normalization_27 (BatchN  (None, None, None,   192        ['conv2d_27[0][0]']
 ormalization)                  64)

 activation_27 (Activation)     (None, None, None,   0           ['batch_normalization_2
7[0][0]']
                                64)

 conv2d_28 (Conv2D)             (None, None, None,   55296       ['activation_27[0][0]']
                                96)

 batch_normalization_28 (BatchN  (None, None, None,   288        ['conv2d_28[0][0]']
 ormalization)                  96)

 activation_28 (Activation)     (None, None, None,   0           ['batch_normalization_2
8[0][0]']
                                96)

 conv2d_26 (Conv2D)             (None, None, None,   995328      ['mixed2[0][0]']
                                384)

 conv2d_29 (Conv2D)             (None, None, None,   82944       ['activation_28[0][0]']
                                96)
```

```
 batch_normalization_26 (BatchN  (None, None, None,   1152       ['conv2d_26[0][0]']
 ormalization)                  384)

 batch_normalization_29 (BatchN  (None, None, None,   288        ['conv2d_29[0][0]']
 ormalization)                  96)

 activation_26 (Activation)     (None, None, None,   0          ['batch_normalization_2
6[0][0]']
                                384)

 activation_29 (Activation)     (None, None, None,   0          ['batch_normalization_2
9[0][0]']
                                96)

 max_pooling2d_2 (MaxPooling2D)  (None, None, None,   0         ['mixed2[0][0]']
                                288)

 mixed3 (Concatenate)           (None, None, None,   0          ['activation_26[0][0]',
                                768)                             'activation_29[0][0]',
                                                                 'max_pooling2d_2[0]
[0]']

 conv2d_34 (Conv2D)             (None, None, None,   98304      ['mixed3[0][0]']
                                128)

 batch_normalization_34 (BatchN  (None, None, None,   384        ['conv2d_34[0][0]']
 ormalization)                  128)

 activation_34 (Activation)     (None, None, None,   0          ['batch_normalization_3
4[0][0]']
                                128)

 conv2d_35 (Conv2D)             (None, None, None,   114688     ['activation_34[0][0]']
                                128)

 batch_normalization_35 (BatchN  (None, None, None,   384        ['conv2d_35[0][0]']
 ormalization)                  128)

 activation_35 (Activation)     (None, None, None,   0          ['batch_normalization_3
5[0][0]']
                                128)

 conv2d_31 (Conv2D)             (None, None, None,   98304      ['mixed3[0][0]']
                                128)

 conv2d_36 (Conv2D)             (None, None, None,   114688     ['activation_35[0][0]']
                                128)

 batch_normalization_31 (BatchN  (None, None, None,   384        ['conv2d_31[0][0]']
 ormalization)                  128)

 batch_normalization_36 (BatchN  (None, None, None,   384        ['conv2d_36[0][0]']
 ormalization)                  128)

 activation_31 (Activation)     (None, None, None,   0          ['batch_normalization_3
1[0][0]']
                                128)

 activation_36 (Activation)     (None, None, None,   0          ['batch_normalization_3
6[0][0]']
                                128)

 conv2d_32 (Conv2D)             (None, None, None,   114688     ['activation_31[0][0]']
                                128)
```

```
 conv2d_37 (Conv2D)             (None, None, None,   114688      ['activation_36[0][0]']
                                128)

 batch_normalization_32 (BatchN  (None, None, None,   384        ['conv2d_32[0][0]']
 ormalization)                  128)

 batch_normalization_37 (BatchN  (None, None, None,   384        ['conv2d_37[0][0]']
 ormalization)                  128)

 activation_32 (Activation)     (None, None, None,   0          ['batch_normalization_3
2[0][0]']
                                128)

 activation_37 (Activation)     (None, None, None,   0          ['batch_normalization_3
7[0][0]']
                                128)

 average_pooling2d_3 (AveragePo  (None, None, None,   0         ['mixed3[0][0]']
 oling2D)                       768)

 conv2d_30 (Conv2D)             (None, None, None,   147456      ['mixed3[0][0]']
                                192)

 conv2d_33 (Conv2D)             (None, None, None,   172032      ['activation_32[0][0]']
                                192)

 conv2d_38 (Conv2D)             (None, None, None,   172032      ['activation_37[0][0]']
                                192)

 conv2d_39 (Conv2D)             (None, None, None,   147456      ['average_pooling2d_3
[0][0]']
                                192)

 batch_normalization_30 (BatchN  (None, None, None,   576        ['conv2d_30[0][0]']
 ormalization)                  192)

 batch_normalization_33 (BatchN  (None, None, None,   576        ['conv2d_33[0][0]']
 ormalization)                  192)

 batch_normalization_38 (BatchN  (None, None, None,   576        ['conv2d_38[0][0]']
 ormalization)                  192)

 batch_normalization_39 (BatchN  (None, None, None,   576        ['conv2d_39[0][0]']
 ormalization)                  192)

 activation_30 (Activation)     (None, None, None,   0          ['batch_normalization_3
0[0][0]']
                                192)

 activation_33 (Activation)     (None, None, None,   0          ['batch_normalization_3
3[0][0]']
                                192)

 activation_38 (Activation)     (None, None, None,   0          ['batch_normalization_3
8[0][0]']
                                192)

 activation_39 (Activation)     (None, None, None,   0          ['batch_normalization_3
9[0][0]']
                                192)

 mixed4 (Concatenate)           (None, None, None,   0          ['activation_30[0][0]',
                                768)                             'activation_33[0][0]',
                                                                 'activation_38[0][0]',
```

```
                                                                  'activation_39[0][0]']

 conv2d_44 (Conv2D)             (None, None, None,   122880      ['mixed4[0][0]']
                                160)

 batch_normalization_44 (BatchN  (None, None, None,   480        ['conv2d_44[0][0]']
 ormalization)                  160)

 activation_44 (Activation)     (None, None, None,   0          ['batch_normalization_4
4[0][0]']
                                160)

 conv2d_45 (Conv2D)             (None, None, None,   179200      ['activation_44[0][0]']
                                160)

 batch_normalization_45 (BatchN  (None, None, None,   480        ['conv2d_45[0][0]']
 ormalization)                  160)

 activation_45 (Activation)     (None, None, None,   0          ['batch_normalization_4
5[0][0]']
                                160)

 conv2d_41 (Conv2D)             (None, None, None,   122880      ['mixed4[0][0]']
                                160)

 conv2d_46 (Conv2D)             (None, None, None,   179200      ['activation_45[0][0]']
                                160)

 batch_normalization_41 (BatchN  (None, None, None,   480        ['conv2d_41[0][0]']
 ormalization)                  160)

 batch_normalization_46 (BatchN  (None, None, None,   480        ['conv2d_46[0][0]']
 ormalization)                  160)

 activation_41 (Activation)     (None, None, None,   0          ['batch_normalization_4
1[0][0]']
                                160)

 activation_46 (Activation)     (None, None, None,   0          ['batch_normalization_4
6[0][0]']
                                160)

 conv2d_42 (Conv2D)             (None, None, None,   179200      ['activation_41[0][0]']
                                160)

 conv2d_47 (Conv2D)             (None, None, None,   179200      ['activation_46[0][0]']
                                160)

 batch_normalization_42 (BatchN  (None, None, None,   480        ['conv2d_42[0][0]']
 ormalization)                  160)

 batch_normalization_47 (BatchN  (None, None, None,   480        ['conv2d_47[0][0]']
 ormalization)                  160)

 activation_42 (Activation)     (None, None, None,   0          ['batch_normalization_4
2[0][0]']
                                160)

 activation_47 (Activation)     (None, None, None,   0          ['batch_normalization_4
7[0][0]']
                                160)

 average_pooling2d_4 (AveragePo  (None, None, None,   0          ['mixed4[0][0]']
 oling2D)                       768)
```

```
 conv2d_40 (Conv2D)             (None, None, None,   147456      ['mixed4[0][0]']
                                192)

 conv2d_43 (Conv2D)             (None, None, None,   215040      ['activation_42[0][0]']
                                192)

 conv2d_48 (Conv2D)             (None, None, None,   215040      ['activation_47[0][0]']
                                192)

 conv2d_49 (Conv2D)             (None, None, None,   147456      ['average_pooling2d_4
[0][0]']
                                192)

 batch_normalization_40 (BatchN  (None, None, None,   576        ['conv2d_40[0][0]']
 ormalization)                  192)

 batch_normalization_43 (BatchN  (None, None, None,   576        ['conv2d_43[0][0]']
 ormalization)                  192)

 batch_normalization_48 (BatchN  (None, None, None,   576        ['conv2d_48[0][0]']
 ormalization)                  192)

 batch_normalization_49 (BatchN  (None, None, None,   576        ['conv2d_49[0][0]']
 ormalization)                  192)

 activation_40 (Activation)     (None, None, None,   0          ['batch_normalization_4
0[0][0]']
                                192)

 activation_43 (Activation)     (None, None, None,   0          ['batch_normalization_4
3[0][0]']
                                192)

 activation_48 (Activation)     (None, None, None,   0          ['batch_normalization_4
8[0][0]']
                                192)

 activation_49 (Activation)     (None, None, None,   0          ['batch_normalization_4
9[0][0]']
                                192)

 mixed5 (Concatenate)           (None, None, None,   0          ['activation_40[0][0]',
                                768)                             'activation_43[0][0]',
                                                                 'activation_48[0][0]',
                                                                 'activation_49[0][0]']

 conv2d_54 (Conv2D)             (None, None, None,   122880      ['mixed5[0][0]']
                                160)

 batch_normalization_54 (BatchN  (None, None, None,   480        ['conv2d_54[0][0]']
 ormalization)                  160)

 activation_54 (Activation)     (None, None, None,   0          ['batch_normalization_5
4[0][0]']
                                160)

 conv2d_55 (Conv2D)             (None, None, None,   179200      ['activation_54[0][0]']
                                160)

 batch_normalization_55 (BatchN  (None, None, None,   480        ['conv2d_55[0][0]']
 ormalization)                  160)

 activation_55 (Activation)     (None, None, None,   0          ['batch_normalization_5
5[0][0]']
                                160)
```

```
 conv2d_51 (Conv2D)             (None, None, None,   122880      ['mixed5[0][0]']
                                160)

 conv2d_56 (Conv2D)             (None, None, None,   179200      ['activation_55[0][0]']
                                160)

 batch_normalization_51 (BatchN  (None, None, None,   480        ['conv2d_51[0][0]']
 ormalization)                  160)

 batch_normalization_56 (BatchN  (None, None, None,   480        ['conv2d_56[0][0]']
 ormalization)                  160)

 activation_51 (Activation)     (None, None, None,   0          ['batch_normalization_5
1[0][0]']
                                160)

 activation_56 (Activation)     (None, None, None,   0          ['batch_normalization_5
6[0][0]']
                                160)

 conv2d_52 (Conv2D)             (None, None, None,   179200      ['activation_51[0][0]']
                                160)

 conv2d_57 (Conv2D)             (None, None, None,   179200      ['activation_56[0][0]']
                                160)

 batch_normalization_52 (BatchN  (None, None, None,   480        ['conv2d_52[0][0]']
 ormalization)                  160)

 batch_normalization_57 (BatchN  (None, None, None,   480        ['conv2d_57[0][0]']
 ormalization)                  160)

 activation_52 (Activation)     (None, None, None,   0          ['batch_normalization_5
2[0][0]']
                                160)

 activation_57 (Activation)     (None, None, None,   0          ['batch_normalization_5
7[0][0]']
                                160)

 average_pooling2d_5 (AveragePo  (None, None, None,   0         ['mixed5[0][0]']
 oling2D)                       768)

 conv2d_50 (Conv2D)             (None, None, None,   147456      ['mixed5[0][0]']
                                192)

 conv2d_53 (Conv2D)             (None, None, None,   215040      ['activation_52[0][0]']
                                192)

 conv2d_58 (Conv2D)             (None, None, None,   215040      ['activation_57[0][0]']
                                192)

 conv2d_59 (Conv2D)             (None, None, None,   147456      ['average_pooling2d_5
[0][0]']
                                192)

 batch_normalization_50 (BatchN  (None, None, None,   576        ['conv2d_50[0][0]']
 ormalization)                  192)

 batch_normalization_53 (BatchN  (None, None, None,   576        ['conv2d_53[0][0]']
 ormalization)                  192)

 batch_normalization_58 (BatchN  (None, None, None,   576        ['conv2d_58[0][0]']
 ormalization)                  192)
```

```
 batch_normalization_59 (BatchN  (None, None, None,   576        ['conv2d_59[0][0]']
 ormalization)                  192)

 activation_50 (Activation)     (None, None, None,   0          ['batch_normalization_5
0[0][0]']
                                192)

 activation_53 (Activation)     (None, None, None,   0          ['batch_normalization_5
3[0][0]']
                                192)

 activation_58 (Activation)     (None, None, None,   0          ['batch_normalization_5
8[0][0]']
                                192)

 activation_59 (Activation)     (None, None, None,   0          ['batch_normalization_5
9[0][0]']
                                192)

 mixed6 (Concatenate)           (None, None, None,   0          ['activation_50[0][0]',
                                768)                             'activation_53[0][0]',
                                                                 'activation_58[0][0]',
                                                                 'activation_59[0][0]']

 conv2d_64 (Conv2D)             (None, None, None,   147456     ['mixed6[0][0]']
                                192)

 batch_normalization_64 (BatchN  (None, None, None,   576        ['conv2d_64[0][0]']
 ormalization)                  192)

 activation_64 (Activation)     (None, None, None,   0          ['batch_normalization_6
4[0][0]']
                                192)

 conv2d_65 (Conv2D)             (None, None, None,   258048     ['activation_64[0][0]']
                                192)

 batch_normalization_65 (BatchN  (None, None, None,   576        ['conv2d_65[0][0]']
 ormalization)                  192)

 activation_65 (Activation)     (None, None, None,   0          ['batch_normalization_6
5[0][0]']
                                192)

 conv2d_61 (Conv2D)             (None, None, None,   147456     ['mixed6[0][0]']
                                192)

 conv2d_66 (Conv2D)             (None, None, None,   258048     ['activation_65[0][0]']
                                192)

 batch_normalization_61 (BatchN  (None, None, None,   576        ['conv2d_61[0][0]']
 ormalization)                  192)

 batch_normalization_66 (BatchN  (None, None, None,   576        ['conv2d_66[0][0]']
 ormalization)                  192)

 activation_61 (Activation)     (None, None, None,   0          ['batch_normalization_6
1[0][0]']
                                192)

 activation_66 (Activation)     (None, None, None,   0          ['batch_normalization_6
6[0][0]']
                                192)
```

```
 conv2d_62 (Conv2D)             (None, None, None,   258048      ['activation_61[0][0]']
                                192)

 conv2d_67 (Conv2D)             (None, None, None,   258048      ['activation_66[0][0]']
                                192)

 batch_normalization_62 (BatchN  (None, None, None,   576        ['conv2d_62[0][0]']
 ormalization)                  192)

 batch_normalization_67 (BatchN  (None, None, None,   576        ['conv2d_67[0][0]']
 ormalization)                  192)

 activation_62 (Activation)     (None, None, None,   0          ['batch_normalization_6
2[0][0]']
                                192)

 activation_67 (Activation)     (None, None, None,   0          ['batch_normalization_6
7[0][0]']
                                192)

 average_pooling2d_6 (AveragePo  (None, None, None,   0         ['mixed6[0][0]']
 oling2D)                       768)

 conv2d_60 (Conv2D)             (None, None, None,   147456      ['mixed6[0][0]']
                                192)

 conv2d_63 (Conv2D)             (None, None, None,   258048      ['activation_62[0][0]']
                                192)

 conv2d_68 (Conv2D)             (None, None, None,   258048      ['activation_67[0][0]']
                                192)

 conv2d_69 (Conv2D)             (None, None, None,   147456      ['average_pooling2d_6
[0][0]']
                                192)

 batch_normalization_60 (BatchN  (None, None, None,   576        ['conv2d_60[0][0]']
 ormalization)                  192)

 batch_normalization_63 (BatchN  (None, None, None,   576        ['conv2d_63[0][0]']
 ormalization)                  192)

 batch_normalization_68 (BatchN  (None, None, None,   576        ['conv2d_68[0][0]']
 ormalization)                  192)

 batch_normalization_69 (BatchN  (None, None, None,   576        ['conv2d_69[0][0]']
 ormalization)                  192)

 activation_60 (Activation)     (None, None, None,   0          ['batch_normalization_6
0[0][0]']
                                192)

 activation_63 (Activation)     (None, None, None,   0          ['batch_normalization_6
3[0][0]']
                                192)

 activation_68 (Activation)     (None, None, None,   0          ['batch_normalization_6
8[0][0]']
                                192)

 activation_69 (Activation)     (None, None, None,   0          ['batch_normalization_6
9[0][0]']
                                192)

 mixed7 (Concatenate)           (None, None, None,   0          ['activation_60[0][0]',
```

```
                                768)                                 'activation_63[0][0]',
                                                                     'activation_68[0][0]',
                                                                     'activation_69[0][0]']

 conv2d_72 (Conv2D)             (None, None, None,   147456      ['mixed7[0][0]']
                                192)

 batch_normalization_72 (BatchN  (None, None, None,   576        ['conv2d_72[0][0]']
 ormalization)                  192)

 activation_72 (Activation)     (None, None, None,   0          ['batch_normalization_7
2[0][0]']
                                192)

 conv2d_73 (Conv2D)             (None, None, None,   258048      ['activation_72[0][0]']
                                192)

 batch_normalization_73 (BatchN  (None, None, None,   576        ['conv2d_73[0][0]']
 ormalization)                  192)

 activation_73 (Activation)     (None, None, None,   0          ['batch_normalization_7
3[0][0]']
                                192)

 conv2d_70 (Conv2D)             (None, None, None,   147456      ['mixed7[0][0]']
                                192)

 conv2d_74 (Conv2D)             (None, None, None,   258048      ['activation_73[0][0]']
                                192)

 batch_normalization_70 (BatchN  (None, None, None,   576        ['conv2d_70[0][0]']
 ormalization)                  192)

 batch_normalization_74 (BatchN  (None, None, None,   576        ['conv2d_74[0][0]']
 ormalization)                  192)

 activation_70 (Activation)     (None, None, None,   0          ['batch_normalization_7
0[0][0]']
                                192)

 activation_74 (Activation)     (None, None, None,   0          ['batch_normalization_7
4[0][0]']
                                192)

 conv2d_71 (Conv2D)             (None, None, None,   552960      ['activation_70[0][0]']
                                320)

 conv2d_75 (Conv2D)             (None, None, None,   331776      ['activation_74[0][0]']
                                192)

 batch_normalization_71 (BatchN  (None, None, None,   960        ['conv2d_71[0][0]']
 ormalization)                  320)

 batch_normalization_75 (BatchN  (None, None, None,   576        ['conv2d_75[0][0]']
 ormalization)                  192)

 activation_71 (Activation)     (None, None, None,   0          ['batch_normalization_7
1[0][0]']
                                320)

 activation_75 (Activation)     (None, None, None,   0          ['batch_normalization_7
5[0][0]']
                                192)

 max_pooling2d_3 (MaxPooling2D)  (None, None, None,   0          ['mixed7[0][0]']
```

```
                                768)

 mixed8 (Concatenate)           (None, None, None,   0           ['activation_71[0][0]',
                                1280)                             'activation_75[0][0]',
                                                                  'max_pooling2d_3[0]
[0]']

 conv2d_80 (Conv2D)             (None, None, None,   573440      ['mixed8[0][0]']
                                448)

 batch_normalization_80 (BatchN  (None, None, None,   1344       ['conv2d_80[0][0]']
 ormalization)                  448)

 activation_80 (Activation)     (None, None, None,   0           ['batch_normalization_8
0[0][0]']
                                448)

 conv2d_77 (Conv2D)             (None, None, None,   491520      ['mixed8[0][0]']
                                384)

 conv2d_81 (Conv2D)             (None, None, None,   1548288     ['activation_80[0][0]']
                                384)

 batch_normalization_77 (BatchN  (None, None, None,   1152       ['conv2d_77[0][0]']
 ormalization)                  384)

 batch_normalization_81 (BatchN  (None, None, None,   1152       ['conv2d_81[0][0]']
 ormalization)                  384)

 activation_77 (Activation)     (None, None, None,   0           ['batch_normalization_7
7[0][0]']
                                384)

 activation_81 (Activation)     (None, None, None,   0           ['batch_normalization_8
1[0][0]']
                                384)

 conv2d_78 (Conv2D)             (None, None, None,   442368      ['activation_77[0][0]']
                                384)

 conv2d_79 (Conv2D)             (None, None, None,   442368      ['activation_77[0][0]']
                                384)

 conv2d_82 (Conv2D)             (None, None, None,   442368      ['activation_81[0][0]']
                                384)

 conv2d_83 (Conv2D)             (None, None, None,   442368      ['activation_81[0][0]']
                                384)

 average_pooling2d_7 (AveragePo  (None, None, None,   0          ['mixed8[0][0]']
 oling2D)                       1280)

 conv2d_76 (Conv2D)             (None, None, None,   409600      ['mixed8[0][0]']
                                320)

 batch_normalization_78 (BatchN  (None, None, None,   1152       ['conv2d_78[0][0]']
 ormalization)                  384)

 batch_normalization_79 (BatchN  (None, None, None,   1152       ['conv2d_79[0][0]']
 ormalization)                  384)

 batch_normalization_82 (BatchN  (None, None, None,   1152       ['conv2d_82[0][0]']
 ormalization)                  384)

 batch_normalization_83 (BatchN  (None, None, None,   1152       ['conv2d_83[0][0]']
```

```
 ormalization)                  384)

 conv2d_84 (Conv2D)             (None, None, None,   245760      ['average_pooling2d_7
[0][0]']
                                192)

 batch_normalization_76 (BatchN  (None, None, None,   960        ['conv2d_76[0][0]']
 ormalization)                  320)

 activation_78 (Activation)     (None, None, None,   0          ['batch_normalization_7
8[0][0]']
                                384)

 activation_79 (Activation)     (None, None, None,   0          ['batch_normalization_7
9[0][0]']
                                384)

 activation_82 (Activation)     (None, None, None,   0          ['batch_normalization_8
2[0][0]']
                                384)

 activation_83 (Activation)     (None, None, None,   0          ['batch_normalization_8
3[0][0]']
                                384)

 batch_normalization_84 (BatchN  (None, None, None,   576        ['conv2d_84[0][0]']
 ormalization)                  192)

 activation_76 (Activation)     (None, None, None,   0          ['batch_normalization_7
6[0][0]']
                                320)

 mixed9_0 (Concatenate)         (None, None, None,   0          ['activation_78[0][0]',
                                768)                             'activation_79[0][0]']

 concatenate (Concatenate)      (None, None, None,   0          ['activation_82[0][0]',
                                768)                             'activation_83[0][0]']

 activation_84 (Activation)     (None, None, None,   0          ['batch_normalization_8
4[0][0]']
                                192)

 mixed9 (Concatenate)           (None, None, None,   0          ['activation_76[0][0]',
                                2048)                            'mixed9_0[0][0]',
                                                                 'concatenate[0][0]',
                                                                 'activation_84[0][0]']

 conv2d_89 (Conv2D)             (None, None, None,   917504      ['mixed9[0][0]']
                                448)

 batch_normalization_89 (BatchN  (None, None, None,   1344       ['conv2d_89[0][0]']
 ormalization)                  448)

 activation_89 (Activation)     (None, None, None,   0          ['batch_normalization_8
9[0][0]']
                                448)

 conv2d_86 (Conv2D)             (None, None, None,   786432      ['mixed9[0][0]']
                                384)

 conv2d_90 (Conv2D)             (None, None, None,   1548288     ['activation_89[0][0]']
                                384)

 batch_normalization_86 (BatchN  (None, None, None,   1152       ['conv2d_86[0][0]']
 ormalization)                  384)
```

```
 batch_normalization_90 (BatchN  (None, None, None,   1152       ['conv2d_90[0][0]']
 ormalization)                   384)

 activation_86 (Activation)     (None, None, None,   0          ['batch_normalization_8
6[0][0]']
                                384)

 activation_90 (Activation)     (None, None, None,   0          ['batch_normalization_9
0[0][0]']
                                384)

 conv2d_87 (Conv2D)             (None, None, None,   442368     ['activation_86[0][0]']
                                384)

 conv2d_88 (Conv2D)             (None, None, None,   442368     ['activation_86[0][0]']
                                384)

 conv2d_91 (Conv2D)             (None, None, None,   442368     ['activation_90[0][0]']
                                384)

 conv2d_92 (Conv2D)             (None, None, None,   442368     ['activation_90[0][0]']
                                384)

 average_pooling2d_8 (AveragePo  (None, None, None,   0          ['mixed9[0][0]']
 oling2D)                        2048)

 conv2d_85 (Conv2D)             (None, None, None,   655360     ['mixed9[0][0]']
                                320)

 batch_normalization_87 (BatchN  (None, None, None,   1152       ['conv2d_87[0][0]']
 ormalization)                   384)

 batch_normalization_88 (BatchN  (None, None, None,   1152       ['conv2d_88[0][0]']
 ormalization)                   384)

 batch_normalization_91 (BatchN  (None, None, None,   1152       ['conv2d_91[0][0]']
 ormalization)                   384)

 batch_normalization_92 (BatchN  (None, None, None,   1152       ['conv2d_92[0][0]']
 ormalization)                   384)

 conv2d_93 (Conv2D)             (None, None, None,   393216     ['average_pooling2d_8
[0][0]']
                                192)

 batch_normalization_85 (BatchN  (None, None, None,   960        ['conv2d_85[0][0]']
 ormalization)                   320)

 activation_87 (Activation)     (None, None, None,   0          ['batch_normalization_8
7[0][0]']
                                384)

 activation_88 (Activation)     (None, None, None,   0          ['batch_normalization_8
8[0][0]']
                                384)

 activation_91 (Activation)     (None, None, None,   0          ['batch_normalization_9
1[0][0]']
                                384)

 activation_92 (Activation)     (None, None, None,   0          ['batch_normalization_9
2[0][0]']
                                384)
```

```
 batch_normalization_93 (BatchN  (None, None, None,   576         ['conv2d_93[0][0]']
 ormalization)                  192)

 activation_85 (Activation)     (None, None, None,   0           ['batch_normalization_8
5[0][0]']
                                320)

 mixed9_1 (Concatenate)         (None, None, None,   0           ['activation_87[0][0]',
                                768)                             'activation_88[0][0]']

 concatenate_1 (Concatenate)    (None, None, None,   0           ['activation_91[0][0]',
                                768)                             'activation_92[0][0]']

 activation_93 (Activation)     (None, None, None,   0           ['batch_normalization_9
3[0][0]']
                                192)

 mixed10 (Concatenate)          (None, None, None,   0           ['activation_85[0][0]',
                                2048)                            'mixed9_1[0][0]',
                                                                 'concatenate_1[0][0]',
                                                                 'activation_93[0][0]']

==================================================================================
==========
Total params: 21,802,784
Trainable params: 21,768,352
Non-trainable params: 34,432
```

In [40]:

```
# extract features from each image in the dataset
img_features = {}
for image, image_path in tqdm(New_Img) :
  batch_features = image_features_extract_model(image)
  #squeeze out the features in a batch
  batch_features_flattened = tf.reshape(batch_features, (batch_features.shape[0], -1, b
  for batch_feat, path in zip(batch_features_flattened, image_path) :
    feature_path = path.numpy().decode('utf-8')
    img_features[feature_path] = batch_feat.numpy()
```

```
100%|██████████| 497/497 [02:53<00:00,  2.87it/s]
```

In [41]:

```
# batch_features
```

In [42]:

```
# batch_features_flattened
```

In [43]:

```
# batch_feat.shape
```

In [44]:

```
# #view top five items of img_features dict
# import more_itertools
# top_5 = more_itertools.take(5, img_features.items())
# top_5
```

In [45]:

```
#to provide, both images along with the captions as input
```

```
def map(image_name, caption):
    img_tensor = img_features[image_name.decode('utf-8')]
    return img_tensor, caption
```

In [46]:

```
# This function should transform the created dataset(img_path,cap) to (features,cap) us
BUFFER_SIZE = 1000
BATCH_SIZE = 64
def gen_dataset(img, capt):

    data = tf.data.Dataset.from_tensor_slices((img, capt))
    data = data.map(lambda ele1, ele2 : tf.numpy_function(map, [ele1, ele2], [tf.float3
                    num_parallel_calls = tf.data.experimental.AUTOTUNE)


    data = (data.shuffle(BUFFER_SIZE, reshuffle_each_iteration= True).batch(BATCH_SIZE,
    .prefetch(tf.data.experimental.AUTOTUNE))
    return data
```

In [47]:

```
train_dataset = gen_dataset(path_train,caption_train)
test_dataset = gen_dataset(path_test,caption_test)
```

In [48]:

```
sample_img_batch, sample_cap_batch = next(iter(train_dataset))
# print(sample_img_batch.shape)  #(batch_size, 8*8, 2048)
# print(sample_cap_batch.shape) #(batch_size,max_len)
```

Model Building

- Set the parameters
- Build the Encoder
- Visualise the top 30 occuring words in the captions
- Visualise the top 30 occuring words in the captions
- Create a list which contains all the captions & path

In [49]:

```
# Setting  parameters

embedding_dim = 256
units = 512

#top 5,000 words +1
vocab_size = 5001
train_num_steps = len(path_train) // BATCH_SIZE
test_num_steps = len(path_test) // BATCH_SIZE

max_length = 31
feature_shape = batch_feat.shape[1]
attention_feature_shape = batch_feat.shape[0]
```

Encoder (CNN)

- The encoder parts involve the convolution of the input image with the help of various convolution, max pooling, and fully connected layers.
- Since we are not dealing with the classification of the image, we have removed them from the end.
- The final output of the encoder part will be the generation of the feature vector.

In [50]:
```
tf.compat.v1.reset_default_graph()
print(tf.compat.v1.get_default_graph())
```

```
<tensorflow.python.framework.ops.Graph object at 0x7f1a5c25e910>
```

In [51]:
```
#Building Encoder using CNN Keras subclassing method

class Encoder(Model):
    def __init__(self,embed_dim):
        super(Encoder, self).__init__()
        self.dense = tf.keras.layers.Dense(embed_dim) #build your Dense layer with relu

    def call(self, features):
        features =  self.dense(features) # extract the features from the image shape: (
        features =  tf.keras.activations.relu(features, alpha=0.01, max_value=None, thr
        return features
```

In [52]:
```
encoder=Encoder(embedding_dim)
```

In [53]:
```
from keras.utils.vis_utils import plot_model
```

Important aspects of Encoder

- The CNN-based encoder produces the feature vector which is the encoded representation of the input image.
- The resulting **feature vector is static** and does not change at each timestamp.
- Therefore we need to pass this vector to the **attention model along with the hidden state of the decoder to create the context vector**.

Attention model

- Attention is an interface connecting the encoder and decoder that provides the decoder with information from every encoder hidden state.
- With this framework, the model is able to selectively focus on valuable parts of the input sequence and hence, learn the association between them.
- The attention model produces an output(context vector) that is fed to the decoder for predicting the word at that timestamp
- This output, i.e context vector is adaptive in nature and change for each timestamp
- It aims to overcome the limitation of traditional CNN-RNN based models. Using this, **instead of passing the complete input image to the RNN at every timestamp, we can pass different relevant parts of the image to it

In [54]:
```
class Attention_model(Model):
    def __init__(self, units):
        super(Attention_model, self).__init__()
        self.W1 = tf.keras.layers.Dense(units)
        self.W2 = tf.keras.layers.Dense(units)
        self.V = tf.keras.layers.Dense(1)
        self.units=units

    def call(self, features, hidden):
        hidden_with_time_axis = hidden[:, tf.newaxis]
        score = tf.keras.activations.tanh(self.W1(features) + self.W2(hidden_with_time_
        attention_weights = tf.keras.activations.softmax(self.V(score), axis=1)
        context_vector = attention_weights * features
        context_vector = tf.reduce_sum(context_vector, axis=1)
        return context_vector, attention_weights
```

Decoder

- Input to GRU : Context vector (from attention model) concatenated with embedded vector (embedding layer). Output of this concatenation layer is fed to GRU as input
- Embedding layer present inside the decoder takes the input sequence preprocessed and transformed such that all samples have equal sequence length - through padding followed by masking). Embedding layer transforms this into an embedded vector.
- Concatenation layer contains : Embedded vector (output of embedding layer) along with the Context vector (output of attention model)

In [55]:
```
class Decoder(Model):
    def __init__(self, embed_dim, units, vocab_size):
        super(Decoder, self).__init__()
        self.units=units
```

```
        self.attention = Attention_model(self.units) #iniitalise your Attention model w
        self.embed = tf.keras.layers.Embedding(vocab_size, embed_dim) #build your Embed
        self.gru = tf.keras.layers.GRU(self.units,return_sequences=True,return_state=Tr
        self.d1 = tf.keras.layers.Dense(self.units) #build your Dense layer
        self.d2 = tf.keras.layers.Dense(vocab_size) #build your Dense layer


    def call(self,x,features, hidden):
        context_vector, attention_weights = self.attention(features, hidden) #create yo
        embed = self.embed(x) # embed your input to shape: (batch_size, 1, embedding_di
        embed = tf.concat([tf.expand_dims(context_vector, 1), embed], axis = -1) # Conc
        output,state = self.gru(embed) # Extract the output & hidden state from GRU lay
        output = self.d1(output)
        output = tf.reshape(output, (-1, output.shape[2])) # shape : (batch_size * max_
        output = self.d2(output) # shape : (batch_size * max_length, vocab_size)

        return output, state, attention_weights

    def init_state(self, batch_size):
        return tf.zeros((batch_size, self.units))
```

In [56]:

```
decoder=Decoder(embedding_dim, units, vocab_size)
```

In [57]:

```
features=encoder(sample_img_batch)

hidden = decoder.init_state(batch_size=sample_cap_batch.shape[0])
dec_input = tf.expand_dims([tokenizer.word_index['<start>']] * sample_cap_batch.shape[0

predictions, hidden_out, attention_weights= decoder(dec_input, features, hidden)
# print('Feature shape from Encoder: {}'.format(features.shape)) #(batch, 8*8, embed_di
# print('Predcitions shape from Decoder: {}'.format(predictions.shape)) #(batch,vocab_s
# print('Attention weights shape from Decoder: {}'.format(attention_weights.shape)) #(b
```

Model training & optimization

- Set the optimizer & loss object
- Visualise both the images & text present in the dataset
- Create your checkpoint path
- Create your training & testing step functions
- Create your loss function for the test dataset

In [58]:

```
# optimizer = tf.keras.optimizers.Adam(learning_rate = 0.001)  #define the optimizer
# loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits = True, reduc
```

In [59]:

```
# def loss_function(real, pred):
#     mask = tf.math.logical_not(tf.math.equal(real, 0))
#     loss_ = loss_object(real, pred)
```

```
#     mask = tf.cast(mask, dtype=loss_.dtype)
#     loss_ *= mask
#     #loss is getting multiplied with mask to get an ideal shape

#     return tf.reduce_mean(loss_)
```

Why Masking ?

- Padding can result in a risk of adding penalty to the model.
- Once the padding is done, we need to apply 'masking'
- Without masking, the model will **consider the padded input at that timestep, which will contribute to an increased loss
- Through masking we need to inform the model to **ignore whenever a padded input is passed at a timestep**, hinting that this part of the input is padded.
- Create a list which contains all the captions & path

In [60]:
```
# checkpoint_path = "Flickr30K/checkpoint1"
# ckpt = tf.train.Checkpoint(encoder=encoder,
#                            decoder=decoder,
#                            optimizer = optimizer)
# ckpt_manager = tf.train.CheckpointManager(ckpt, checkpoint_path, max_to_keep=5)
```

In [61]:
```
# start_epoch = 0
# if ckpt_manager.latest_checkpoint:
#     start_epoch = int(ckpt_manager.latest_checkpoint.split('-')[-1])
```

While creating the training step for your model, we will apply Teacher forcing mechanism.

- Why Teacher Forcing ?
- There are multiple issues with with training recurrent neural networks that use output from prior time steps as input :
- Slow convergence
- Model instability
- Poor skill

- If the previous output is incorrect (by any chance), it will result in inaccurate input for the next time stamp which will further result in a different output than expected and the process will continue.
- As a result, the model will get off track and will get punished for every subsequent word it generates. This makes learning slower and the model unstable.
- To address this we Teacher Forcing

Teacher forcing is a fast and effective way to train a recurrent neural network, where the **target/real word (i.e ground truth) is passed as the next input to the decoder instead of previous prediciton or output.** Training with Teacher Forcing **converges faster.** At the early stages of training, the predictions of the model are very bad. If we do not use Teacher Forcing, the hidden states of the model will be updated by a sequence of wrong predictions, errors will accumulate, and it is difficult for the model to learn from that.

In [62]:
```
# @tf.function
# def train_step(img_tensor, target):
#     loss = 0
#     hidden = decoder.init_state(batch_size=target.shape[0])
#     dec_input = tf.expand_dims([tokenizer.word_index['<start>']] * target.shape[0], 1

#     with tf.GradientTape() as tape:

#         encoder_op = encoder(img_tensor)
#         for r in range(1, target.shape[1]) :
#           predictions, hidden, _ = decoder(dec_input, encoder_op, hidden)
#           loss = loss + loss_function(target[:, r], predictions)
#           dec_input = tf.expand_dims(target[:, r], 1)

#     avg_loss = (loss/ int(target.shape[1])) #avg loss per batch
#     trainable_vars = encoder.trainable_variables + decoder.trainable_variables
#     grad = tape.gradient (loss, trainable_vars)
#     optimizer.apply_gradients(zip(grad, trainable_vars))

#     return loss, avg_loss
```

In [63]:
```
# @tf.function
# def test_step(img_tensor, target):
#     loss = 0
#     hidden = decoder.init_state(batch_size = target.shape[0])
#     dec_input = tf.expand_dims([tokenizer.word_index['<start>']] * target.shape[0], 1
#     with tf.GradientTape() as tape:
#       encoder_op = encoder(img_tensor)
#       for r in range(1, target.shape[1]) :
#         predictions, hidden, _ = decoder(dec_input, encoder_op, hidden)
#         loss = loss + loss_function(target[:, r], predictions)
#         dec_input = tf.expand_dims(target[: , r], 1)
#     avg_loss = (loss/ int(target.shape[1])) #avg loss per batch
#     trainable_vars = encoder.trainable_variables + decoder.trainable_variables
#     grad = tape.gradient (loss, trainable_vars)
```

```
#     optimizer.apply_gradients(zip(grad, trainable_vars))
#     return loss, avg_loss
```

In [64]:

```
# def test_loss_cal(test_dataset):
#     total_loss = 0
#     for (batch, (img_tensor, target)) in enumerate(test_dataset) :
#       batch_loss, t_loss = test_step(img_tensor, target)
#       total_loss = total_loss + t_loss
#       avg_test_loss = total_loss/ test_num_steps

#     return avg_test_loss
```

In [65]:

```
# loss_plot = []
# test_loss_plot = []
# EPOCHS = 30
# best_test_loss=100
# for epoch in tqdm(range(0, EPOCHS)):
#     start = time.time()
#     total_loss = 0
#     for (batch, (img_tensor, target)) in enumerate(train_dataset):
#         batch_loss, t_loss = train_step(img_tensor, target)
#         total_loss += t_loss
#         avg_train_loss=total_loss / train_num_steps
#     loss_plot.append(avg_train_loss)
#     test_loss = test_loss_cal(test_dataset)
#     test_loss_plot.append(test_loss)
#     print ('For epoch: {}, the train loss is {:.3f}, & test loss is {:.3f}'.format(ep
#     print ('Time taken for 1 epoch {} sec\n'.format(time.time() - start))
#     if test_loss < best_test_loss:
#         print('Test loss has been reduced from %.3f to %.3f' % (best_test_loss, test_
#         best_test_loss = test_loss
#         ckpt_manager.save()
```

In [66]:

```
# from matplotlib.pyplot import figure
# figure(figsize=(12, 8))
# plt.plot(loss_plot, color='orange', label = 'training_loss_plot')
# plt.plot(test_loss_plot, color='green', label = 'test_loss_plot')
# plt.xlabel('Epochs', fontsize = 15, color = 'red')
# plt.ylabel('Loss', fontsize = 15, color = 'red')
# plt.title('Loss Plot', fontsize = 20, color = 'red')
# plt.legend()
# plt.show()
```

Model Evaluation

- Define your evaluation function using greedy search
- Define your evaluation function using beam search (optional)
- Test it on a sample data using BLEU score

Greedy Search

- This method is a simple approximation technique which calculates the probability of the words according to their occurrence in the English vocabulary.
- It takes the sample of the words, finds the probability of each of the words, and then outputs the word with the highest probability.
- Greedy Search will always consider only one best alternative and this makes the computational speed of the model fast, but the accuracy might not be up to the mark.

In [67]:

```
def evaluate(image):
    attention_plot = np.zeros((max_length, attention_feature_shape))

    hidden = decoder.init_state(batch_size=1)

    temp_input = tf.expand_dims(load_images(image)[0], 0)
    img_tensor_val = image_features_extract_model(temp_input)
    img_tensor_val = tf.reshape(img_tensor_val, (img_tensor_val.shape[0], -1, img_tenso

    features = encoder (img_tensor_val)

    dec_input = tf.expand_dims([tokenizer.word_index['<start>']], 0)
    result = []

    for i in range(max_length):
        predictions, hidden, attention_weights = decoder(dec_input, features, hidden)
        attention_plot[i] = tf.reshape(attention_weights, (-1, )).numpy()

        predicted_id = tf.argmax(predictions[0]).numpy()
        result.append (tokenizer.index_word[predicted_id])

        if tokenizer.index_word[predicted_id] == '<end>':
            return result, attention_plot,predictions

        dec_input = tf.expand_dims([predicted_id], 0)

    attention_plot = attention_plot[:len(result), :]
    return result, attention_plot,predictions
```

In [68]:

```
def plot_attention_map (caption, weights, image) :

  fig = plt.figure(figsize = (20, 20))
  temp_img = np.array(Image.open(image))

  if (caption[-1]== '<end>'):
    caption.pop()

  cap_len = len(caption)

  for cap in range(cap_len) :
    weights_img = np.reshape(weights[cap], (8,8))
    wweights_img = np.array(Image.fromarray(weights_img).resize((224,224), Image.LANCZO

    ax = fig.add_subplot(cap_len//2, cap_len//2, cap+1)
    ax.set_title(caption[cap], fontsize = 14, color = 'red')
```

```
    img = ax.imshow(temp_img)

    ax.imshow(weights_img, cmap='gist_heat', alpha=0.6, extent=img.get_extent())
    ax.axis('off')
  plt.subplots_adjust(hspace=0.2, wspace=0.2)
  plt.show()
```

In [69]:
```
from nltk.translate.bleu_score import sentence_bleu
```

In [70]:
```
def filt_text(text):
    filt=['<start>','<unk>','<end>']
    temp= text.split()
    [temp.remove(j) for k in filt for j in temp if k==j]
    text=' '.join(temp)
    return text
```

In [71]:
```
# image_test = path_test.copy()
```

In [72]:
```
# def pred_caption_audio(random, autoplay=False, weights=(0.5, 0.5, 0, 0)) :

#     cap_test_data = caption_test.copy()
#     rid = np.random.randint(0, random)
#     test_image = image_test[rid]
#     real_caption = ' '.join([tokenizer.index_word[i] for i in cap_test_data[rid] if i
#     result, attention_plot, pred_test = evaluate(test_image)
#     real_caption=filt_text(real_caption)
#     pred_caption=' '.join(result).rsplit(' ', 1)[0]
#     real_appn = []
#     real_appn.append(real_caption.split())
#     reference = real_appn
#     candidate = pred_caption.split()
#     score = sentence_bleu(reference, candidate, weights=weights)#set your weights
#     print(f"BLEU score: {score*100}")
#     print ('Real Caption:', real_caption)
#     print ('Prediction Caption:', pred_caption)
#     plot_attention_map(result, attention_plot, test_image)
#     speech = gTTS('Predicted Caption : ' + pred_caption, lang = 'en', slow = False)
#     speech.save('voice.mp3')
#     audio_file = 'voice.mp3'

#     display.display(display.Audio(audio_file, rate = None, autoplay = autoplay))

#     return test_image
```

Test Images

- BLEU is a well-acknowledged metric to measure similarity of one hypothesis sentence to multiple reference sentences. Given a single hypothesis sentence and multiple reference sentences, it returns a value between 0 and 1

- The metric close to 1 means that the two are very similar.
- We use the BLEU measure to evaluate the result of the test set generated captions. The BLEU is simply taking the fraction of n-grams in the predicted sentence that appears in the ground-truth

In [73]:
```
# encoder.save_weights('./outputs_encoder_weights.h5')
# decoder.save_weights('./outputs_decoder_weights.h5')
```

In [74]:
```
encoder.load_weights('./outputs_encoder_weights.h5')
decoder.load_weights('./outputs_decoder_weights.h5')
```

In [75]:
```
def test_audio(path, autoplay=False, weights=(0.5, 0.5, 0, 0)) :

    test_image = path
    #real_caption = ' '.join([tokenizer.index_word[i] for i in cap_test_data[rid] if i
    result, attention_plot, pred_test = evaluate(test_image)
    #real_caption=filt_text(real_caption)
    pred_caption=' '.join(result).rsplit(' ', 1)[0]
    #real_appn = []
    #real_appn.append(real_caption.split())
    #reference = real_appn
    candidate = pred_caption.split()
    #score = sentence_bleu(reference, candidate, weights=weights)#set your weights
    #print(f"BLEU score: {score*100}")
    #print ('Real Caption:', real_caption)
    print ('Prediction Caption:', pred_caption)
    plot_attention_map(result, attention_plot, test_image)
    speech = gTTS('Predicted Caption : ' + pred_caption, lang = 'en', slow = False)
    speech.save('voice.mp3')
    audio_file = 'voice.mp3'

    display.display(display.Audio(audio_file, rate = None, autoplay = autoplay))

    return test_image
```

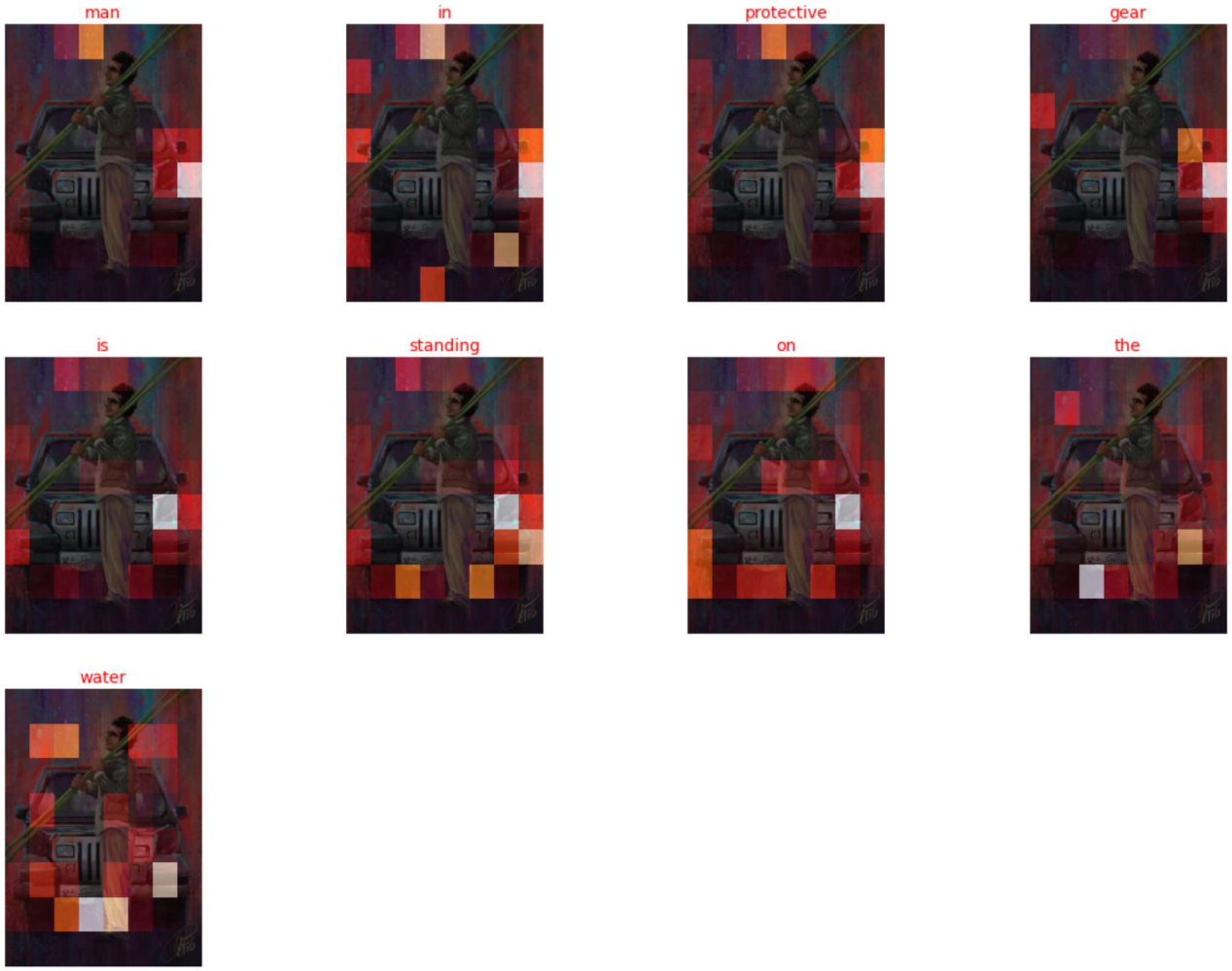
In [92]:
```
path_list = glob.glob("/home/ubuntu/trainContent/try_me/*")
```

In [97]:
```
path_list[2]
```

Out[97]:
```
'/home/ubuntu/trainContent/try_me/atef.jpg'
```

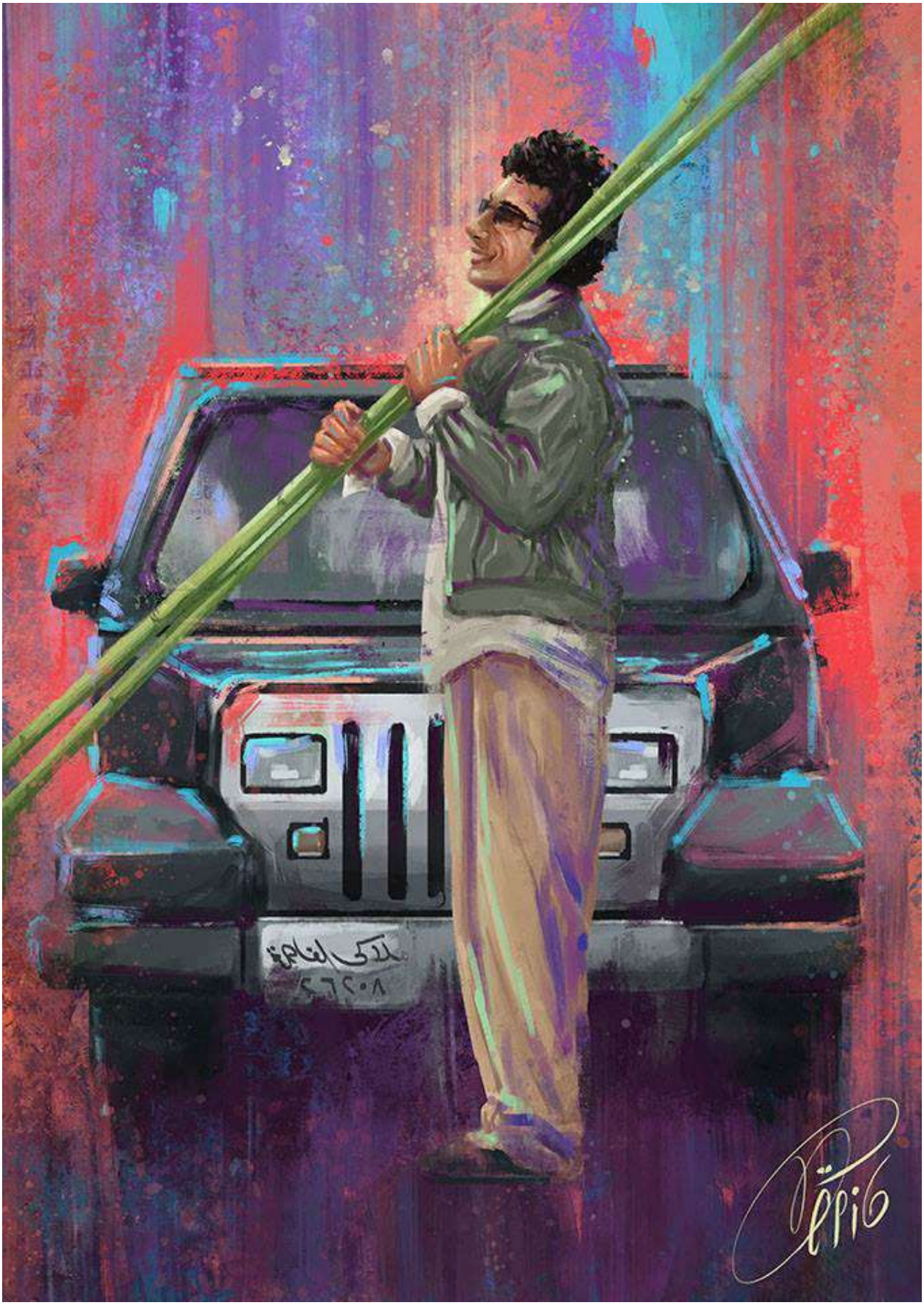
In [98]:
```
test_image = test_audio(path_list[2], False, weights = (0.25, 0.25, 0, 0))
Image.open(test_image)
```

```
Prediction Caption: man in protective gear is standing on the water
```



0:00 / 0:05

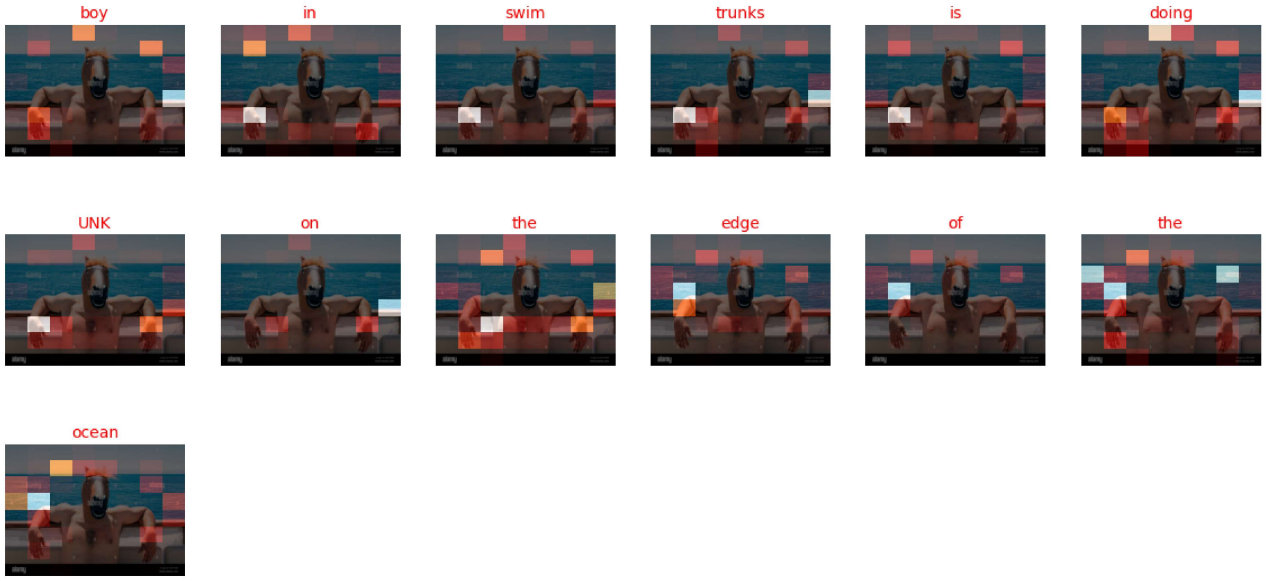
Out[98]:



In [95]:

```
for i in range(len(path_list)):
    test_image = test_audio(path_list[i], False, weights = (0.25, 0.25, 0, 0))
    Image.open(path_list[i])
```

```
Prediction Caption: boy in swim trunks is doing UNK on the edge of the ocean
```

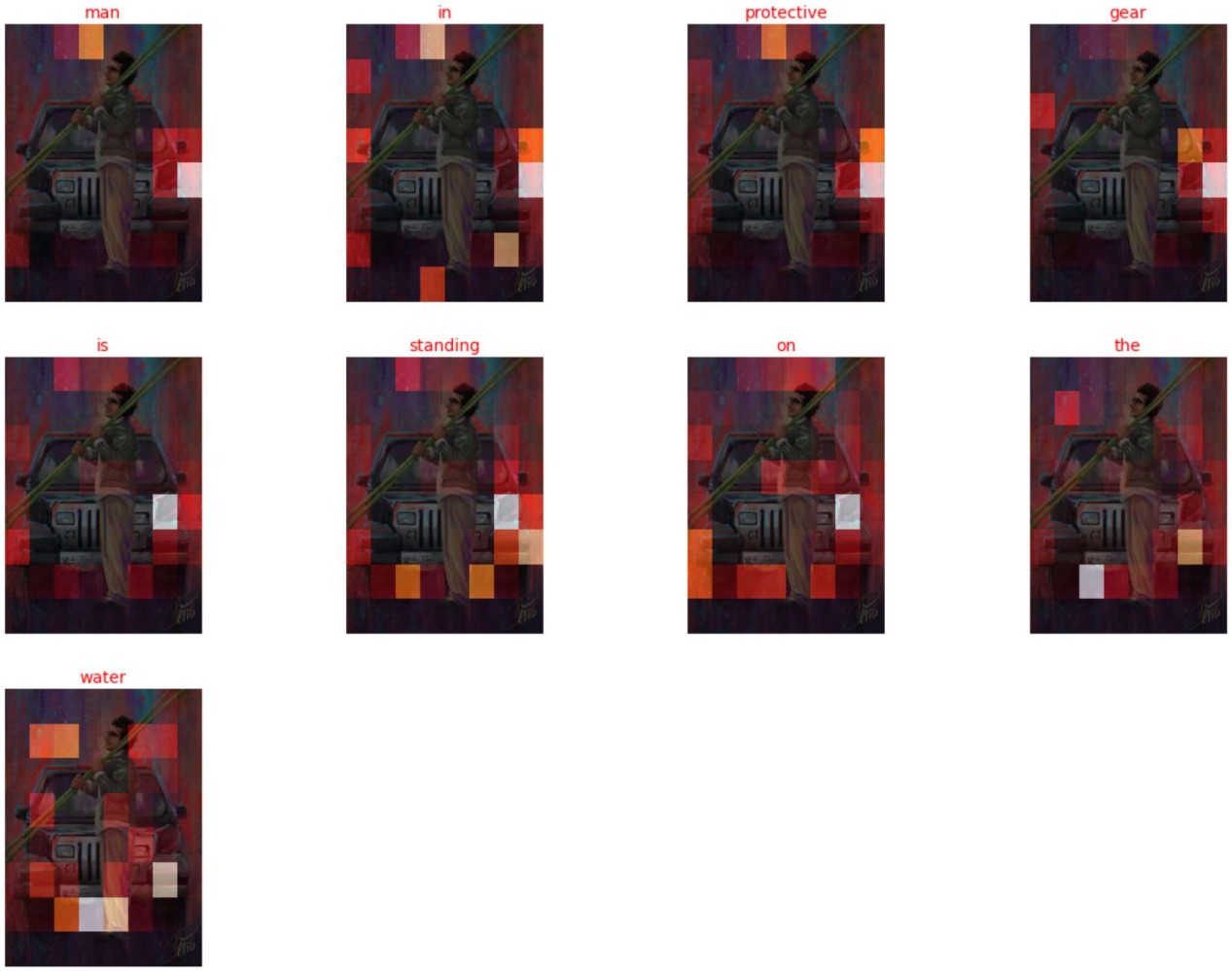


0:00 / 0:05

```
Prediction Caption: man is performing tricks
```

0:00 / 0:03

```
Prediction Caption: man in protective gear is standing on the water
```

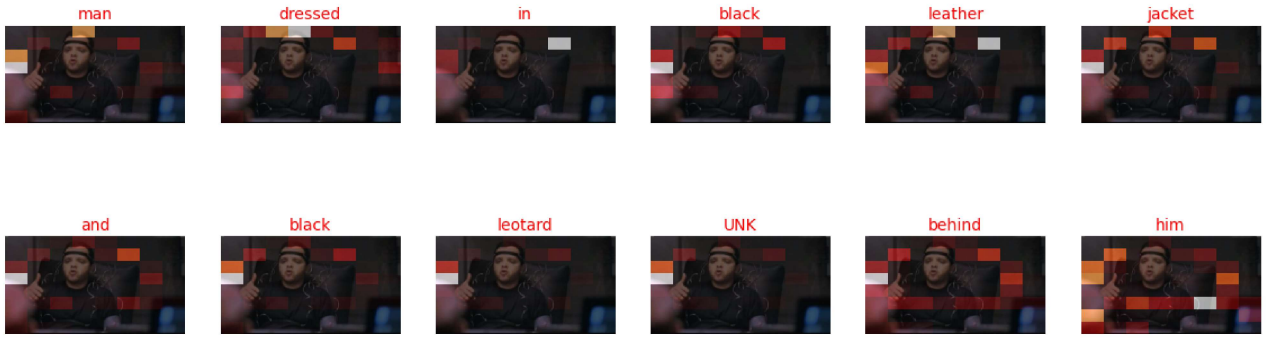


0:00 / 0:05

```
Prediction Caption: boy in swim trunks is doing UNK on the edge of the ocean
```

0:00 / 0:05

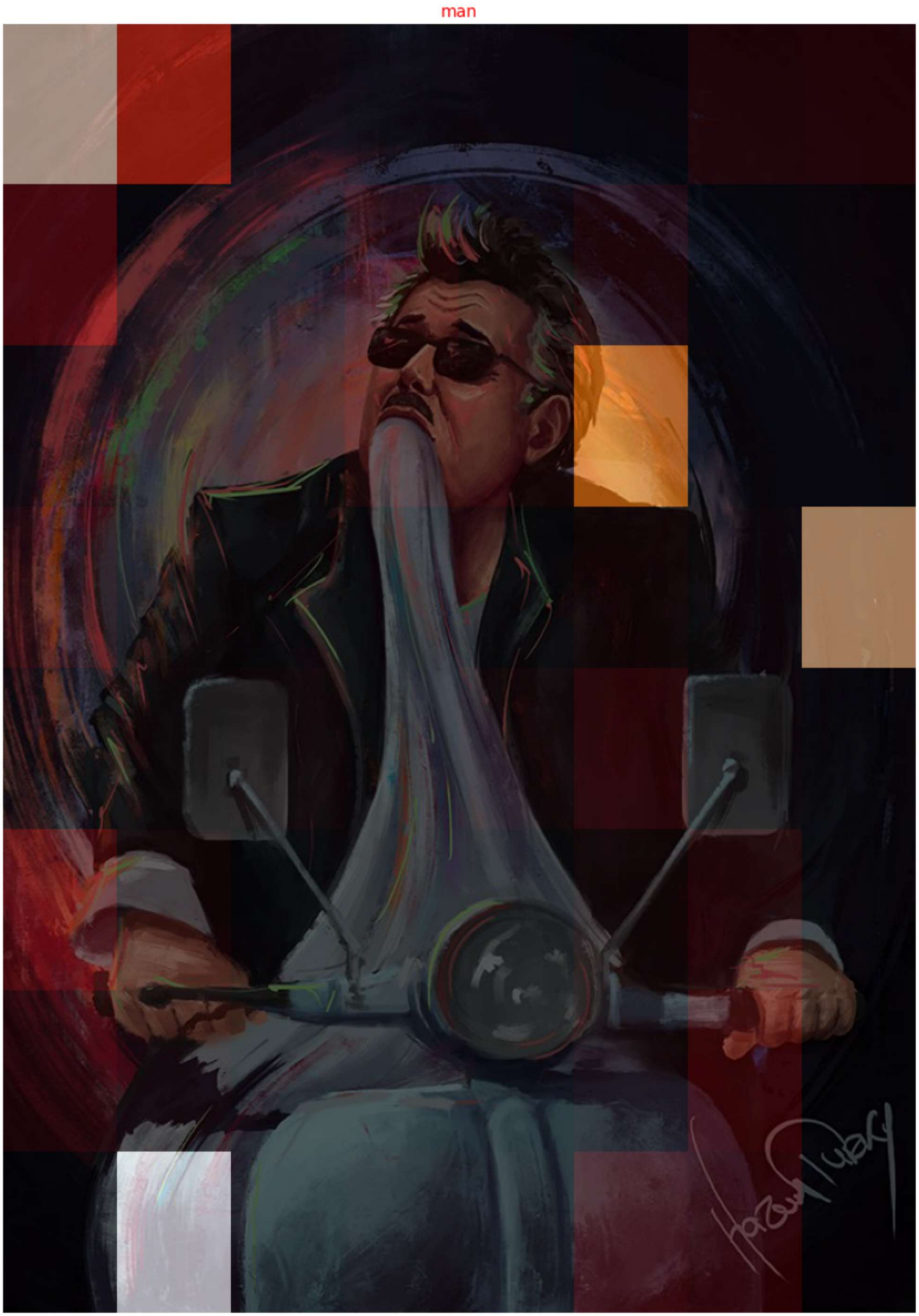
```
Prediction Caption: man dressed in black leather jacket and black leotard UNK behind him
```



0:00 / 0:06

```
Prediction Caption: man in tunnel
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipykernel_19509/3586871901.py in <module>
      1 for i in range(len(path_list)):
----> 2     test_image = test_audio(path_list[i], False, weights = (0.25, 0.25, 0, 0))
      3     Image.open(path_list[i])

/tmp/ipykernel_19509/3919787099.py in test_audio(path, autoplay, weights)
     14     #print ('Real Caption:', real_caption)
     15     print ('Prediction Caption:', pred_caption)
---> 16     plot_attention_map(result, attention_plot, test_image)
     17     speech = gTTS('Predicted Caption : ' + pred_caption, lang = 'en', slow = Fa
lse)
     18     speech.save('voice.mp3')

/tmp/ipykernel_19509/3522479041.py in plot_attention_map(caption, weights, image)
     13     wweights_img = np.array(Image.fromarray(weights_img).resize((224,224), Imag
e.LANCZOS))
     14
---> 15     ax = fig.add_subplot(cap_len//2, cap_len//2, cap+1)
     16     ax.set_title(caption[cap], fontsize = 14, color = 'red')
     17

~/anaconda3/envs/tensorflow2_p38/lib/python3.8/site-packages/matplotlib/figure.py in add
_subplot(self, *args, **kwargs)
    782             projection_class, pkw = self._process_projection_requirements(
    783                 *args, **kwargs)
--> 784             ax = subplot_class_factory(projection_class)(self, *args, **pkw)
    785             key = (projection_class, pkw)
    786         return self._add_axes_internal(ax, key)

~/anaconda3/envs/tensorflow2_p38/lib/python3.8/site-packages/matplotlib/axes/_subplots.p
y in __init__(self, fig, *args, **kwargs)
     36         self._axes_class.__init__(self, fig, [0, 0, 1, 1], **kwargs)
     37         # This will also update the axes position.
---> 38         self.set_subplotspec(SubplotSpec._from_subplot_args(fig, args))
     39
     40     def __reduce__(self):

~/anaconda3/envs/tensorflow2_p38/lib/python3.8/site-packages/matplotlib/gridspec.py in _
from_subplot_args(figure, args)
    651                     num = int(num)
    652                 if num < 1 or num > rows*cols:
--> 653                     raise ValueError(
    654                         f"num must be 1 <= num <= {rows*cols}, not {num}")
    655                 i = j = num
```

```
ValueError: num must be 1 <= num <= 1, not 2
```



In [ ]: