In [1]:

```
# https://www.kaggle.com/hsankesara/flickr-image-dataset

! wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.149 Safari/537.36" --header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-US,en;q=0.9" --header="Referer: https://www.kaggle.com/" "https://storage.googleapis.com/kaggle-data-sets/31296/39911/bundle/archive.zip?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1586244948&Signature=QER5siwrR2gk5SuOZ%2BpKcAbSmNh0fCrUqIzPXsjWy6goXUk6ULHMqHZFmkZHoheiloVWbmF0J6tEk2rpvRBU3sWk2xmVRUfCPvlWdeGt6lkHKZb%2BjYuxn7%2ByOy%2FsdrmrcXm4zOWkqawWN1JHAywoltoV6kYj%2BcksVHerq8RSuLW95tlyIWe08jcjxHp3PQu1KrAx1aN59TfLGm6KAof%2B4v3yGtGkiCOuG7Yj6WwJ4OTbaE8ZJhQIu1tPGE28t5RNzYQAhfTB3Mah5adSkGAYqiCht0F4BJ6U7PR56jGyEbX25RCAPqV6JZRIf06QX1gU1%2BBpUNP4cl0iCkifGerN2A%3D%3D&response-content-disposition=attachment%3B+filename%3Dflickr-image-dataset.zip" -c -O 'flickr-image-dataset.zip'
```

```
--2020-04-04 18:32:54--  https://storage.googleapis.com/kaggle-data-sets/31296/39911/bundle/archive.zip?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1586244948&Signature=QER5siwrR2gk5SuOZ%2BpKcAbSmNh0fCrUqIzPXsjWy6goXUk6ULHMqHZFmkZHoheiloVWbmF0J6tEk2rpvRBU3sWk2xmVRUfCPvlWdeGt6lkHKZb%2BjYuxn7%2ByOy%2FsdrmrcXm4zOWkqawWN1JHAywoltoV6kYj%2BcksVHerq8RSuLW95tlyIWe08jcjxHp3PQu1KrAx1aN59TfLGm6KAof%2B4v3yGtGkiCOuG7Yj6WwJ4OTbaE8ZJhQIu1tPGE28t5RNzYQAhfTB3Mah5adSkGAYqiCht0F4BJ6U7PR56jGyEbX25RCAPqV6JZRIf06QX1gU1%2BBpUNP4cl0iCkifGerN2A%3D%3D&response-content-disposition=attachment%3B+filename%3Dflickr-image-dataset.zip
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.201.128, 2607:f8b0:4001:c16::80
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.201.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8765396518 (8.2G) [application/zip]
Saving to: ‘flickr-image-dataset.zip’

flickr-image-datase 100%[===================>]   8.16G  45.7MB/s    in 3m 9s

2020-04-04 18:36:03 (44.2 MB/s) - ‘flickr-image-dataset.zip’ saved [8765396518/8765396518]
```

In [2]:

```
import datetime
import time

start= time.time()

import zipfile
with zipfile.ZipFile("/content/flickr-image-dataset.zip","r") as zip_ref:
  zip_ref.extractall()

print("Time Taken is: " + str(time.time() - start))
```

```
Time Taken is: 217.12104773521423
```

In [3]:

```
!pip3 install contractions
```

```
Collecting contractions
  Downloading https://files.pythonhosted.org/packages/85/41/c3dfd5feb91a8d587ed1a59f553f07c05f95ad4e5d00ab78702fbf8fe48a/contractions-0.0.24-py2.py3-none-any.whl
Collecting textsearch
  Downloading https://files.pythonhosted.org/packages/42/a8/03407021f9555043de5492a2bd7a35c56cc03c2510092b5ec018cae1bbf1/textsearch-0.0.17-py2.py3-none-any.whl
Collecting pyahocorasick
  Downloading https://files.pythonhosted.org/packages/f4/9f/f0d8e8850e12829eea2e778f1c90e3c53a9a799b7f412082a5d21cd19ae1/pyahocorasick-1.4.0.tar.gz (312kB)
     |████████████████████████████████| 317kB 4.6MB/s
Collecting Unidecode
  Downloading https://files.pythonhosted.org/packages/d0/42/d9edfed04228bacea2d824904cae367ee9efd05e6cce7ceaaedd0b0ad964/Unidecode-1.1.1-py2.py3-none-any.whl (238kB)
     |████████████████████████████████| 245kB 10.2MB/s
Building wheels for collected packages: pyahocorasick
  Building wheel for pyahocorasick (setup.py) ... done
  Created wheel for pyahocorasick: filename=pyahocorasick-1.4.0-cp36-cp36m-linux_x86_64.whl size=81707 sha256=2a8fb1765e60592defcf071a8d0a1609caac7029ddf4d0df888c6c8e142c00e2
  Stored in directory: /root/.cache/pip/wheels/0a/90/61/87a55f5b459792fbb2b7ba6b31721b06ff5cf6bde541b40994
Successfully built pyahocorasick
Installing collected packages: pyahocorasick, Unidecode, textsearch, contractions
Successfully installed Unidecode-1.1.1 contractions-0.0.24 pyahocorasick-1.4.0 textsearch-0.0.17
```

In [0]:

```
# We import necessary Libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
from PIL import Image
import glob
import pickle
from time import time
from bs4 import BeautifulSoup
import contractions
import re

import tensorflow as tf
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Embedding, TimeDistributed, Dense, RepeatVector,\
                                    Activation, Flatten, Reshape, concatenate, Dropout, \
                                    BatchNormalization, Bidirectional
from tensorflow.keras.optimizers import Adam, RMSprop
from tensorflow.keras.applications.inception_v3 import InceptionV3, preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model
from tensorflow.keras import Input, layers
from tensorflow.keras import optimizers
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
```

In [5]:

```
# https://stackoverflow.com/a/18129082/10219869

df= pd.read_csv(r'/content/flickr30k_images/results.csv', error_bad_lines=False, sep='|')
print(df.shape)
df.head(10)
```

(158915, 3)

Out[5]:

	image_name	comment_number	comment
0	1000092795.jpg	0	Two young guys with shaggy hair look at their...
1	1000092795.jpg	1	Two young , White males are outside near many...
2	1000092795.jpg	2	Two men in green shirts are standing in a yard .
3	1000092795.jpg	3	A man in a blue shirt standing in a garden .
4	1000092795.jpg	4	Two friends enjoy time spent together .
5	10002456.jpg	0	Several men in hard hats are operating a gian...
6	10002456.jpg	1	Workers look down from up above on a piece of...
7	10002456.jpg	2	Two men working on a machine wearing hard hats .
8	10002456.jpg	3	Four men on top of a tall structure .
9	10002456.jpg	4	Three men on a large rig .

In [6]:

```
# To check the names of columns
df.columns
```

Out[6]:

```
Index(['image_name', ' comment_number', ' comment'], dtype='object')
```

In [7]:

```
df[' comment'][:5]
```

Out[7]:

```
0    Two young guys with shaggy hair look at their...
1    Two young , White males are outside near many...
```

```
2     Two men in green shirts are standing in a yard .
3         A man in a blue shirt standing in a garden .
4              Two friends enjoy time spent together .
Name:  comment, dtype: object
```

In [8]:

```
# for every five consecutive indexes, there is same image hence same name.
df['image_name'][:5]
```

Out[8]:

```
0    1000092795.jpg
1    1000092795.jpg
2    1000092795.jpg
3    1000092795.jpg
4    1000092795.jpg
Name: image_name, dtype: object
```

In [9]:

```
# Check for any nan values

print('Image_name:\n', df[df['image_name'].isnull()])
print('Comment_number:\n', df[df[' comment_number'].isnull()])
print('Comment:\n', df[df[' comment'].isnull()])
```

```
Image_name:
 Empty DataFrame
Columns: [image_name,  comment_number,  comment]
Index: []
Comment_number:
 Empty DataFrame
Columns: [image_name,  comment_number,  comment]
Index: []
Comment:
           image_name                    comment_number  comment
19999  2199200615.jpg   4   A dog runs across the grass .      NaN
```

In [0]:

```
# As we see there is aproblem with index number 19999 and we shall solve it by replacing these values
# https://stackoverflow.com/a/37725243/10219869

df.loc[19999, ' comment_number'] = 4
df.loc[19999, ' comment'] = 'A dog runs across the grass .'
```

In [11]:

```
# Now it is good
df[19999:20000]
```

Out[11]:

	image_name	comment_number	comment
19999	2199200615.jpg	4	A dog runs across the grass .

In [12]:

```
# Considering 30,000 captions => 6000 Images

new_df= df[:30000]
new_df.tail(10)
```

Out[12]:

	image_name	comment_number	comment
29990	244829722.jpg	0	A woman in a white t-shirt is in a swing that...
29991	244829722.jpg	1	A girl going on a ride in a circus that spins...
29992	244829722.jpg	2	A young woman rides by herself on a swinging ...

	image_name	comment_number	comment
29993	244829722.jpg	3	Girl in a swing ride at an amusement park .
29994	244829722.jpg	4	A woman is on a carnival swing ride .
29995	2448393373.jpg	0	A young boy wearing blue is holding a blue ba...
29996	2448393373.jpg	1	A boy with a plastic bat , looking skyward , ...
29997	2448393373.jpg	2	A small boy playing in the grass with a blue ...
29998	2448393373.jpg	3	A little boy plays baseball with himself .
29999	2448393373.jpg	4	A boy plays baseball .

In [13]:

```
# We need the words for corpus and we need to clean the sentences based on the below conditions

def clean_sentence(text):
  text = BeautifulSoup(text, 'lxml').get_text() # removes html tags such as <br />
  text = ''.join([i for i in text if not i.isdigit()]) # removes numbers
  text = text.lower() # converts text to lower case
  text = contractions.fix(text) # converts (don't) to (do not)
  text = re.sub('\W+',' ',text) # removes all special chars, punc
  text = text.split(' ')
  return text

# Clean the individual sentences and then obtain a set of word corpus
sentences= [clean_sentence(i) for i in new_df[' comment']]
print('Length of sentences:', len(sentences))
```

Length of sentences: 30000

In [14]:

```
# We observed a space at the end of every string hence we need to remove it
new_sentences= []
for i in sentences:
  del i[-1]
  new_sentences.append(i)

print(len(new_sentences))
```

30000

In [0]:

```
"""Now, we create a dictionary named “desc” which contains the name of the image (without the .jpg extension)
   as keys and a list of the 5 captions for the corresponding image as values."""

# We remove the jpg extension from name of image and then remove the 5 duplicate names to 1 name
image= [i.replace('.jpg','') for i in new_df['image_name']]

# https://stackoverflow.com/a/7961390/10219869
from collections import OrderedDict
key= list(OrderedDict.fromkeys(image))

comment= [' '.join(i) for i in sentences] # sentences, because we have already cleaned them.

# https://stackoverflow.com/a/312464/10219869
# 5 captions for a single image
value= [comment[i:i+5] for i in range(0, len(comment), 5)]

desc= dict(zip(key, value))
```

In [16]:

```
# Sample output before
desc['1000092795']
```

Out[16]:

['two young guys with shaggy hair look at their hands while hanging out in the yard',
 'two young white males are outside near many bushes',
 'two men in green shirts are standing in a yard',
 'a man in a blue shirt standing in a garden',
 'two friends enjoy time spent together']

In [0]:

```
# We split the data into train and test at 95% train (5700) and 5% test (300).

train_desc= dict(list(desc.items())[0:5700])
test_desc= dict(list(desc.items())[5700:])
```

In [0]:

```
# We append the 'startseq' and 'endseq' for each caption in order for data generator
# 'startseq' -> This is a start sequence token which will be added at the start of every caption.
# 'endseq' -> This is an end sequence token which will be added at the end of every caption.

train_key= list(train_desc.keys())
values= []
for i in list(train_desc.values()):
  v=[]
  for j in i:
    a= 'startseq ' + j + ' endseq'
    v.append(a)
  values.append(v)

new_desc= dict(zip(train_key, values))
```

In [19]:

```
# A sample check

values[0]
```

Out[19]:

```
['startseq two young guys with shaggy hair look at their hands while hanging out in the yard endseq',
 'startseq two young white males are outside near many bushes endseq',
 'startseq two men in green shirts are standing in a yard endseq',
 'startseq a man in a blue shirt standing in a garden endseq',
 'startseq two friends enjoy time spent together endseq']
```

In [20]:

```
# all words (new_desc is train data) into this list
word_corpus=[]

for i in list(new_desc.values()):
  for j in i:
    j= j.split(' ')
    for k in j:
      word_corpus.append(k)

print('Total words before Duplicates:', len(word_corpus))
words_corpus= set(word_corpus)
print('Total words after removing duplicates:',len(words_corpus))
```

```
Total words before Duplicates: 400159
Total words after removing duplicates: 8937
```

In [21]:

```
# We consider those words which occurred more than 5 times in a corpus
'''
Since we are creating a predictive model, we would not like to have all the words present in our vocabulary but the words
which are more likely to occur or which are common. This helps the model become more robust to outliers and make less mistakes.
'''
# https://stackoverflow.com/a/26773321/10219869

from collections import Counter
cnt= Counter(word_corpus)
new_word_corpus= [i for i in cnt if cnt[i] > 5]

print('Total number of words appearing more than 5 times :',len(new_word_corpus))

# for zero padding we consider + 1 to the above count => 2831
```

```
Total number of words appearing more than 5 times : 2830
```

In [22]:

```
# Now we combine all the captions in new_desc (train data)

train_captions= []
for i in list(new_desc.values()):
  for j in i:
    train_captions.append(j)

print(len(train_captions))
```

28500

In [23]:

```
# Word mapping to integers and vice versa for new_corpus_words (words got repeated > 5 times)

new_keys= list(range(1, 2831))

ix_to_word= dict(zip(new_keys, new_word_corpus))
word_to_ix= dict(zip(new_word_corpus, new_keys))

# Sample answers
print(list(ix_to_word.items())[:10])

print(list(word_to_ix.items())[:10])
```

[(1, 'startseq'), (2, 'two'), (3, 'young'), (4, 'guys'), (5, 'with'), (6, 'shaggy'), (7, 'hair'), (8, 'look'), (9, 'at'), (10, 'their')]
[('startseq', 1), ('two', 2), ('young', 3), ('guys', 4), ('with', 5), ('shaggy', 6), ('hair', 7), ('look', 8), ('at', 9), ('their', 10)]

In [24]:

```
word_to_ix['startseq']
```

Out[24]:

1

In [25]:

```
ix_to_word[1]
```

Out[25]:

'startseq'

In [26]:

```
vocab_size = len(ix_to_word) + 1  #1 for appended zeros
vocab_size
```

Out[26]:

2831

In [27]:

```
# To find the max length  of a description so that we can easily pad sequences
l= []
for i in train_captions:
  i= i.split()
  l.append(len(i))

print('Max length of caption:', max(l))
```

Max length of caption: 80

In [28]:

```
# This is the caption which has maximum length of words.
# l.index(80)
print(train_captions[16050])
```

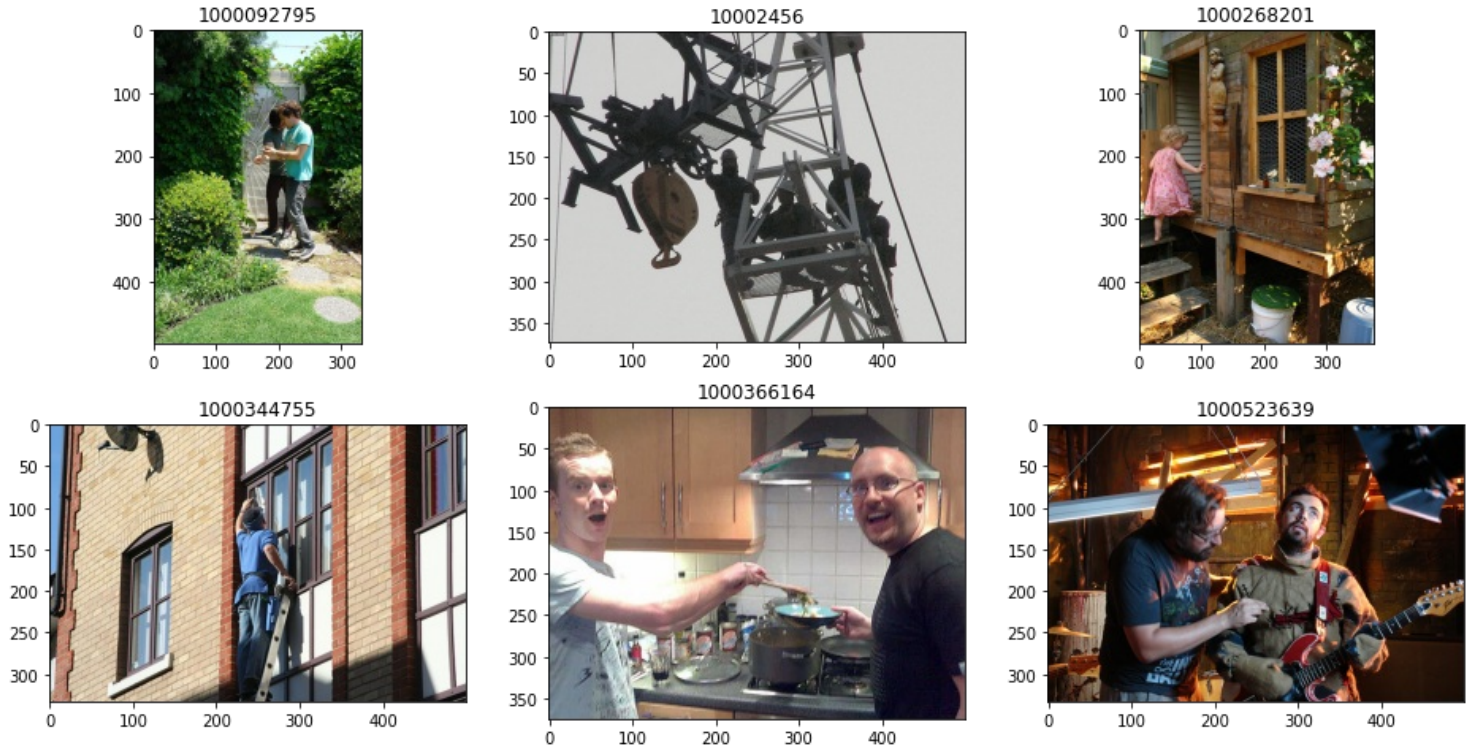
startseq a man wearing a helmet red pants with white stripes going down the sides and a white and red shirt is on a small bicycle using only his hand
s while his legs are up in the air while another man wearing a light blue shirt with dark blue trim and black pants with red stripes going up the sides is

s while his legs are up in the air while another man wearing a light blue shirt with dark blue trim and black pants with red stripes going up the sides is standing nearby gesturing toward the first man and holding a small figurine of one of the seven dwarves endseq

In [29]:

```
# We now check sample images

path= '/content/flickr30k_images/flickr30k_images/'   # the path is provided

image=[]
for i in key[:6]:
  image.append(path+i+'.jpg')                    # we append the name of image to path along with .jpg string

title= key[:6]
plt.figure(figsize= (16, 8))

for i in range(6):
  plt.subplot(2, 3, i+1)
  j= plt.imread(image[i])
  print(j.shape)
  plt.imshow(j)
  plt.title(title[i])

plt.show()
```

```
(500, 333, 3)
(374, 500, 3)
(500, 375, 3)
(333, 500, 3)
(375, 500, 3)
(333, 500, 3)
```



In [30]:

```
# We create new list and preserve the images
import time
start= time.time()

Images= [path+i+'.jpg' for i in key]
print('No of Images: ', len(Images))

# We check the max and min shape of images
width=[]
height=[]

for i in Images:
  j= plt.imread(i)
  w, h, channel= j.shape
  width.append(w)
  height.append(h)
```

```
print('Max Width: ', max(width))
print('Min Width: ', min(width))
print('Max Height: ', max(height))
print('Min Height: ', min(height))

print("Time Taken is: " + str(time.time() - start))
```

```
No of Images:  6000
Max Width:  500
Min Width:  157
Max Height:  500
Min Height:  200
Time Taken is: 26.211413621902466
```

In [31]:

```
# Just to ensure the sequence to dictionary matches with sequence of images
a= dict(zip(list(train_desc.keys())[:10], Images[:10]))
a
```

Out[31]:

```
{'1000092795': '/content/flickr30k_images/flickr30k_images/1000092795.jpg',
 '10002456': '/content/flickr30k_images/flickr30k_images/10002456.jpg',
 '1000268201': '/content/flickr30k_images/flickr30k_images/1000268201.jpg',
 '1000344755': '/content/flickr30k_images/flickr30k_images/1000344755.jpg',
 '1000366164': '/content/flickr30k_images/flickr30k_images/1000366164.jpg',
 '1000523639': '/content/flickr30k_images/flickr30k_images/1000523639.jpg',
 '1000919630': '/content/flickr30k_images/flickr30k_images/1000919630.jpg',
 '10010052': '/content/flickr30k_images/flickr30k_images/10010052.jpg',
 '1001465944': '/content/flickr30k_images/flickr30k_images/1001465944.jpg',
 '1001545525': '/content/flickr30k_images/flickr30k_images/1001545525.jpg'}
```

In [32]:

```
# To split the images into train and test based on 95% ratio of 30200 and 5% 1583 totalled to 31783.

train_images= Images[:5700]
test_images= Images[5700:]
print(len(train_images))
print(len(test_images))
```

```
5700
300
```

In [33]:

```
base_model = InceptionV3(weights = 'imagenet')
base_model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels.
h5
96116736/96112376 [==============================] - 1s 0us/step
Model: "inception_v3"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 299, 299, 3) 0
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 149, 149, 32) 864         input_1[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 149, 149, 32) 96          conv2d[0][0]
__________________________________________________________________________________________________
activation (Activation)         (None, 149, 149, 32) 0           batch_normalization[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 147, 147, 32) 9216        activation[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 147, 147, 32) 96          conv2d_1[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 147, 147, 32) 0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 147, 147, 64) 18432       activation_1[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 147, 147, 64) 192         conv2d_2[0][0]
__________________________________________________________________________________________________
activation_2 (Activation)       (None, 147, 147, 64) 0           batch_normalization_2[0][0]
__________________________________________________________________________________________________
```

max_pooling2d (MaxPooling2D) (None, 73, 73, 64) 0 activation_2[0][0]

conv2d_3 (Conv2D) (None, 73, 73, 80) 5120 max_pooling2d[0][0]

batch_normalization_3 (BatchNor (None, 73, 73, 80) 240 conv2d_3[0][0]

activation_3 (Activation) (None, 73, 73, 80) 0 batch_normalization_3[0][0]

conv2d_4 (Conv2D) (None, 71, 71, 192) 138240 activation_3[0][0]

batch_normalization_4 (BatchNor (None, 71, 71, 192) 576 conv2d_4[0][0]

activation_4 (Activation) (None, 71, 71, 192) 0 batch_normalization_4[0][0]

max_pooling2d_1 (MaxPooling2D) (None, 35, 35, 192) 0 activation_4[0][0]

conv2d_8 (Conv2D) (None, 35, 35, 64) 12288 max_pooling2d_1[0][0]

batch_normalization_8 (BatchNor (None, 35, 35, 64) 192 conv2d_8[0][0]

activation_8 (Activation) (None, 35, 35, 64) 0 batch_normalization_8[0][0]

conv2d_6 (Conv2D) (None, 35, 35, 48) 9216 max_pooling2d_1[0][0]

conv2d_9 (Conv2D) (None, 35, 35, 96) 55296 activation_8[0][0]

batch_normalization_6 (BatchNor (None, 35, 35, 48) 144 conv2d_6[0][0]

batch_normalization_9 (BatchNor (None, 35, 35, 96) 288 conv2d_9[0][0]

activation_6 (Activation) (None, 35, 35, 48) 0 batch_normalization_6[0][0]

activation_9 (Activation) (None, 35, 35, 96) 0 batch_normalization_9[0][0]

average_pooling2d (AveragePooli (None, 35, 35, 192) 0 max_pooling2d_1[0][0]

conv2d_5 (Conv2D) (None, 35, 35, 64) 12288 max_pooling2d_1[0][0]

conv2d_7 (Conv2D) (None, 35, 35, 64) 76800 activation_6[0][0]

conv2d_10 (Conv2D) (None, 35, 35, 96) 82944 activation_9[0][0]

conv2d_11 (Conv2D) (None, 35, 35, 32) 6144 average_pooling2d[0][0]

batch_normalization_5 (BatchNor (None, 35, 35, 64) 192 conv2d_5[0][0]

batch_normalization_7 (BatchNor (None, 35, 35, 64) 192 conv2d_7[0][0]

batch_normalization_10 (BatchNo (None, 35, 35, 96) 288 conv2d_10[0][0]

batch_normalization_11 (BatchNo (None, 35, 35, 32) 96 conv2d_11[0][0]

activation_5 (Activation) (None, 35, 35, 64) 0 batch_normalization_5[0][0]

activation_7 (Activation) (None, 35, 35, 64) 0 batch_normalization_7[0][0]

activation_10 (Activation) (None, 35, 35, 96) 0 batch_normalization_10[0][0]

activation_11 (Activation) (None, 35, 35, 32) 0 batch_normalization_11[0][0]

mixed0 (Concatenate) (None, 35, 35, 256) 0 activation_5[0][0]
activation_7[0][0]
activation_10[0][0]
activation_11[0][0]

conv2d_15 (Conv2D) (None, 35, 35, 64) 16384 mixed0[0][0]

batch_normalization_15 (BatchNo (None, 35, 35, 64) 192 conv2d_15[0][0]

activation_15 (Activation) (None, 35, 35, 64) 0 batch_normalization_15[0][0]

conv2d_13 (Conv2D) (None, 35, 35, 48) 12288 mixed0[0][0]

conv2d_16 (Conv2D) (None, 35, 35, 96) 55296 activation_15[0][0]

batch_normalization_13 (BatchNo (None, 35, 35, 48) 144 conv2d_13[0][0]

batch_normalization_16 (BatchNo (None, 35, 35, 96) 288 conv2d_16[0][0]

activation_13 (Activation) (None, 35, 35, 48) 0 batch_normalization_13[0][0]

activation_16 (Activation) (None, 35, 35, 96) 0 batch_normalization_16[0][0]

average_pooling2d_1 (AveragePoo	(None, 35, 35, 256)	0	mixed0[0][0]
conv2d_12 (Conv2D)	(None, 35, 35, 64)	16384	mixed0[0][0]
conv2d_14 (Conv2D)	(None, 35, 35, 64)	76800	activation_13[0][0]
conv2d_17 (Conv2D)	(None, 35, 35, 96)	82944	activation_16[0][0]
conv2d_18 (Conv2D)	(None, 35, 35, 64)	16384	average_pooling2d_1[0][0]
batch_normalization_12 (BatchNo	(None, 35, 35, 64)	192	conv2d_12[0][0]
batch_normalization_14 (BatchNo	(None, 35, 35, 64)	192	conv2d_14[0][0]
batch_normalization_17 (BatchNo	(None, 35, 35, 96)	288	conv2d_17[0][0]
batch_normalization_18 (BatchNo	(None, 35, 35, 64)	192	conv2d_18[0][0]
activation_12 (Activation)	(None, 35, 35, 64)	0	batch_normalization_12[0][0]
activation_14 (Activation)	(None, 35, 35, 64)	0	batch_normalization_14[0][0]
activation_17 (Activation)	(None, 35, 35, 96)	0	batch_normalization_17[0][0]
activation_18 (Activation)	(None, 35, 35, 64)	0	batch_normalization_18[0][0]
mixed1 (Concatenate)	(None, 35, 35, 288)	0	activation_12[0][0] activation_14[0][0] activation_17[0][0] activation_18[0][0]
conv2d_22 (Conv2D)	(None, 35, 35, 64)	18432	mixed1[0][0]
batch_normalization_22 (BatchNo	(None, 35, 35, 64)	192	conv2d_22[0][0]
activation_22 (Activation)	(None, 35, 35, 64)	0	batch_normalization_22[0][0]
conv2d_20 (Conv2D)	(None, 35, 35, 48)	13824	mixed1[0][0]
conv2d_23 (Conv2D)	(None, 35, 35, 96)	55296	activation_22[0][0]
batch_normalization_20 (BatchNo	(None, 35, 35, 48)	144	conv2d_20[0][0]
batch_normalization_23 (BatchNo	(None, 35, 35, 96)	288	conv2d_23[0][0]
activation_20 (Activation)	(None, 35, 35, 48)	0	batch_normalization_20[0][0]
activation_23 (Activation)	(None, 35, 35, 96)	0	batch_normalization_23[0][0]
average_pooling2d_2 (AveragePoo	(None, 35, 35, 288)	0	mixed1[0][0]
conv2d_19 (Conv2D)	(None, 35, 35, 64)	18432	mixed1[0][0]
conv2d_21 (Conv2D)	(None, 35, 35, 64)	76800	activation_20[0][0]
conv2d_24 (Conv2D)	(None, 35, 35, 96)	82944	activation_23[0][0]
conv2d_25 (Conv2D)	(None, 35, 35, 64)	18432	average_pooling2d_2[0][0]
batch_normalization_19 (BatchNo	(None, 35, 35, 64)	192	conv2d_19[0][0]
batch_normalization_21 (BatchNo	(None, 35, 35, 64)	192	conv2d_21[0][0]
batch_normalization_24 (BatchNo	(None, 35, 35, 96)	288	conv2d_24[0][0]
batch_normalization_25 (BatchNo	(None, 35, 35, 64)	192	conv2d_25[0][0]
activation_19 (Activation)	(None, 35, 35, 64)	0	batch_normalization_19[0][0]
activation_21 (Activation)	(None, 35, 35, 64)	0	batch_normalization_21[0][0]
activation_24 (Activation)	(None, 35, 35, 96)	0	batch_normalization_24[0][0]
activation_25 (Activation)	(None, 35, 35, 64)	0	batch_normalization_25[0][0]
mixed2 (Concatenate)	(None, 35, 35, 288)	0	activation_19[0][0] activation_21[0][0] activation_24[0][0] activation_25[0][0]
conv2d_27 (Conv2D)	(None, 35, 35, 64)	18432	mixed2[0][0]
batch_normalization_27 (BatchNo	(None, 35, 35, 64)	192	conv2d_27[0][0]

batch_normalization_27 (BatchNo (None, 35, 35, 64) 192 conv2d_27[0][0]

activation_27 (Activation) (None, 35, 35, 64) 0 batch_normalization_27[0][0]

conv2d_28 (Conv2D) (None, 35, 35, 96) 55296 activation_27[0][0]

batch_normalization_28 (BatchNo (None, 35, 35, 96) 288 conv2d_28[0][0]

activation_28 (Activation) (None, 35, 35, 96) 0 batch_normalization_28[0][0]

conv2d_26 (Conv2D) (None, 17, 17, 384) 995328 mixed2[0][0]

conv2d_29 (Conv2D) (None, 17, 17, 96) 82944 activation_28[0][0]

batch_normalization_26 (BatchNo (None, 17, 17, 384) 1152 conv2d_26[0][0]

batch_normalization_29 (BatchNo (None, 17, 17, 96) 288 conv2d_29[0][0]

activation_26 (Activation) (None, 17, 17, 384) 0 batch_normalization_26[0][0]

activation_29 (Activation) (None, 17, 17, 96) 0 batch_normalization_29[0][0]

max_pooling2d_2 (MaxPooling2D) (None, 17, 17, 288) 0 mixed2[0][0]

mixed3 (Concatenate) (None, 17, 17, 768) 0 activation_26[0][0]
activation_29[0][0]
max_pooling2d_2[0][0]

conv2d_34 (Conv2D) (None, 17, 17, 128) 98304 mixed3[0][0]

batch_normalization_34 (BatchNo (None, 17, 17, 128) 384 conv2d_34[0][0]

activation_34 (Activation) (None, 17, 17, 128) 0 batch_normalization_34[0][0]

conv2d_35 (Conv2D) (None, 17, 17, 128) 114688 activation_34[0][0]

batch_normalization_35 (BatchNo (None, 17, 17, 128) 384 conv2d_35[0][0]

activation_35 (Activation) (None, 17, 17, 128) 0 batch_normalization_35[0][0]

conv2d_31 (Conv2D) (None, 17, 17, 128) 98304 mixed3[0][0]

conv2d_36 (Conv2D) (None, 17, 17, 128) 114688 activation_35[0][0]

batch_normalization_31 (BatchNo (None, 17, 17, 128) 384 conv2d_31[0][0]

batch_normalization_36 (BatchNo (None, 17, 17, 128) 384 conv2d_36[0][0]

activation_31 (Activation) (None, 17, 17, 128) 0 batch_normalization_31[0][0]

activation_36 (Activation) (None, 17, 17, 128) 0 batch_normalization_36[0][0]

conv2d_32 (Conv2D) (None, 17, 17, 128) 114688 activation_31[0][0]

conv2d_37 (Conv2D) (None, 17, 17, 128) 114688 activation_36[0][0]

batch_normalization_32 (BatchNo (None, 17, 17, 128) 384 conv2d_32[0][0]

batch_normalization_37 (BatchNo (None, 17, 17, 128) 384 conv2d_37[0][0]

activation_32 (Activation) (None, 17, 17, 128) 0 batch_normalization_32[0][0]

activation_37 (Activation) (None, 17, 17, 128) 0 batch_normalization_37[0][0]

average_pooling2d_3 (AveragePoo (None, 17, 17, 768) 0 mixed3[0][0]

conv2d_30 (Conv2D) (None, 17, 17, 192) 147456 mixed3[0][0]

conv2d_33 (Conv2D) (None, 17, 17, 192) 172032 activation_32[0][0]

conv2d_38 (Conv2D) (None, 17, 17, 192) 172032 activation_37[0][0]

conv2d_39 (Conv2D) (None, 17, 17, 192) 147456 average_pooling2d_3[0][0]

batch_normalization_30 (BatchNo (None, 17, 17, 192) 576 conv2d_30[0][0]

batch_normalization_33 (BatchNo (None, 17, 17, 192) 576 conv2d_33[0][0]

batch_normalization_38 (BatchNo (None, 17, 17, 192) 576 conv2d_38[0][0]

batch_normalization_39 (BatchNo (None, 17, 17, 192) 576 conv2d_39[0][0]

activation_30 (Activation) (None, 17, 17, 192) 0 batch_normalization_30[0][0]

activation_33 (Activation)	(None, 17, 17, 192)	0	batch_normalization_33[0][0]
activation_38 (Activation)	(None, 17, 17, 192)	0	batch_normalization_38[0][0]
activation_39 (Activation)	(None, 17, 17, 192)	0	batch_normalization_39[0][0]
mixed4 (Concatenate)	(None, 17, 17, 768)	0	activation_30[0][0] activation_33[0][0] activation_38[0][0] activation_39[0][0]
conv2d_44 (Conv2D)	(None, 17, 17, 160)	122880	mixed4[0][0]
batch_normalization_44 (BatchNo	(None, 17, 17, 160)	480	conv2d_44[0][0]
activation_44 (Activation)	(None, 17, 17, 160)	0	batch_normalization_44[0][0]
conv2d_45 (Conv2D)	(None, 17, 17, 160)	179200	activation_44[0][0]
batch_normalization_45 (BatchNo	(None, 17, 17, 160)	480	conv2d_45[0][0]
activation_45 (Activation)	(None, 17, 17, 160)	0	batch_normalization_45[0][0]
conv2d_41 (Conv2D)	(None, 17, 17, 160)	122880	mixed4[0][0]
conv2d_46 (Conv2D)	(None, 17, 17, 160)	179200	activation_45[0][0]
batch_normalization_41 (BatchNo	(None, 17, 17, 160)	480	conv2d_41[0][0]
batch_normalization_46 (BatchNo	(None, 17, 17, 160)	480	conv2d_46[0][0]
activation_41 (Activation)	(None, 17, 17, 160)	0	batch_normalization_41[0][0]
activation_46 (Activation)	(None, 17, 17, 160)	0	batch_normalization_46[0][0]
conv2d_42 (Conv2D)	(None, 17, 17, 160)	179200	activation_41[0][0]
conv2d_47 (Conv2D)	(None, 17, 17, 160)	179200	activation_46[0][0]
batch_normalization_42 (BatchNo	(None, 17, 17, 160)	480	conv2d_42[0][0]
batch_normalization_47 (BatchNo	(None, 17, 17, 160)	480	conv2d_47[0][0]
activation_42 (Activation)	(None, 17, 17, 160)	0	batch_normalization_42[0][0]
activation_47 (Activation)	(None, 17, 17, 160)	0	batch_normalization_47[0][0]
average_pooling2d_4 (AveragePoo	(None, 17, 17, 768)	0	mixed4[0][0]
conv2d_40 (Conv2D)	(None, 17, 17, 192)	147456	mixed4[0][0]
conv2d_43 (Conv2D)	(None, 17, 17, 192)	215040	activation_42[0][0]
conv2d_48 (Conv2D)	(None, 17, 17, 192)	215040	activation_47[0][0]
conv2d_49 (Conv2D)	(None, 17, 17, 192)	147456	average_pooling2d_4[0][0]
batch_normalization_40 (BatchNo	(None, 17, 17, 192)	576	conv2d_40[0][0]
batch_normalization_43 (BatchNo	(None, 17, 17, 192)	576	conv2d_43[0][0]
batch_normalization_48 (BatchNo	(None, 17, 17, 192)	576	conv2d_48[0][0]
batch_normalization_49 (BatchNo	(None, 17, 17, 192)	576	conv2d_49[0][0]
activation_40 (Activation)	(None, 17, 17, 192)	0	batch_normalization_40[0][0]
activation_43 (Activation)	(None, 17, 17, 192)	0	batch_normalization_43[0][0]
activation_48 (Activation)	(None, 17, 17, 192)	0	batch_normalization_48[0][0]
activation_49 (Activation)	(None, 17, 17, 192)	0	batch_normalization_49[0][0]
mixed5 (Concatenate)	(None, 17, 17, 768)	0	activation_40[0][0] activation_43[0][0] activation_48[0][0] activation_49[0][0]
conv2d_54 (Conv2D)	(None, 17, 17, 160)	122880	mixed5[0][0]
batch_normalization_54 (BatchNo	(None, 17, 17, 160)	480	conv2d_54[0][0]

activation_54 (Activation)	(None, 17, 17, 160)	0	batch_normalization_54[0][0]
conv2d_55 (Conv2D)	(None, 17, 17, 160)	179200	activation_54[0][0]
batch_normalization_55 (BatchNo	(None, 17, 17, 160)	480	conv2d_55[0][0]
activation_55 (Activation)	(None, 17, 17, 160)	0	batch_normalization_55[0][0]
conv2d_51 (Conv2D)	(None, 17, 17, 160)	122880	mixed5[0][0]
conv2d_56 (Conv2D)	(None, 17, 17, 160)	179200	activation_55[0][0]
batch_normalization_51 (BatchNo	(None, 17, 17, 160)	480	conv2d_51[0][0]
batch_normalization_56 (BatchNo	(None, 17, 17, 160)	480	conv2d_56[0][0]
activation_51 (Activation)	(None, 17, 17, 160)	0	batch_normalization_51[0][0]
activation_56 (Activation)	(None, 17, 17, 160)	0	batch_normalization_56[0][0]
conv2d_52 (Conv2D)	(None, 17, 17, 160)	179200	activation_51[0][0]
conv2d_57 (Conv2D)	(None, 17, 17, 160)	179200	activation_56[0][0]
batch_normalization_52 (BatchNo	(None, 17, 17, 160)	480	conv2d_52[0][0]
batch_normalization_57 (BatchNo	(None, 17, 17, 160)	480	conv2d_57[0][0]
activation_52 (Activation)	(None, 17, 17, 160)	0	batch_normalization_52[0][0]
activation_57 (Activation)	(None, 17, 17, 160)	0	batch_normalization_57[0][0]
average_pooling2d_5 (AveragePoo	(None, 17, 17, 768)	0	mixed5[0][0]
conv2d_50 (Conv2D)	(None, 17, 17, 192)	147456	mixed5[0][0]
conv2d_53 (Conv2D)	(None, 17, 17, 192)	215040	activation_52[0][0]
conv2d_58 (Conv2D)	(None, 17, 17, 192)	215040	activation_57[0][0]
conv2d_59 (Conv2D)	(None, 17, 17, 192)	147456	average_pooling2d_5[0][0]
batch_normalization_50 (BatchNo	(None, 17, 17, 192)	576	conv2d_50[0][0]
batch_normalization_53 (BatchNo	(None, 17, 17, 192)	576	conv2d_53[0][0]
batch_normalization_58 (BatchNo	(None, 17, 17, 192)	576	conv2d_58[0][0]
batch_normalization_59 (BatchNo	(None, 17, 17, 192)	576	conv2d_59[0][0]
activation_50 (Activation)	(None, 17, 17, 192)	0	batch_normalization_50[0][0]
activation_53 (Activation)	(None, 17, 17, 192)	0	batch_normalization_53[0][0]
activation_58 (Activation)	(None, 17, 17, 192)	0	batch_normalization_58[0][0]
activation_59 (Activation)	(None, 17, 17, 192)	0	batch_normalization_59[0][0]
mixed6 (Concatenate)	(None, 17, 17, 768)	0	activation_50[0][0] activation_53[0][0] activation_58[0][0] activation_59[0][0]
conv2d_64 (Conv2D)	(None, 17, 17, 192)	147456	mixed6[0][0]
batch_normalization_64 (BatchNo	(None, 17, 17, 192)	576	conv2d_64[0][0]
activation_64 (Activation)	(None, 17, 17, 192)	0	batch_normalization_64[0][0]
conv2d_65 (Conv2D)	(None, 17, 17, 192)	258048	activation_64[0][0]
batch_normalization_65 (BatchNo	(None, 17, 17, 192)	576	conv2d_65[0][0]
activation_65 (Activation)	(None, 17, 17, 192)	0	batch_normalization_65[0][0]
conv2d_61 (Conv2D)	(None, 17, 17, 192)	147456	mixed6[0][0]
conv2d_66 (Conv2D)	(None, 17, 17, 192)	258048	activation_65[0][0]
batch_normalization_61 (BatchNo	(None, 17, 17, 192)	576	conv2d_61[0][0]
batch_normalization_66 (BatchNo	(None, 17, 17, 192)	576	conv2d_66[0][0]

activation_61 (Activation)	(None, 17, 17, 192)	0	batch_normalization_61[0][0]
activation_66 (Activation)	(None, 17, 17, 192)	0	batch_normalization_66[0][0]
conv2d_62 (Conv2D)	(None, 17, 17, 192)	258048	activation_61[0][0]
conv2d_67 (Conv2D)	(None, 17, 17, 192)	258048	activation_66[0][0]
batch_normalization_62 (BatchNo	(None, 17, 17, 192)	576	conv2d_62[0][0]
batch_normalization_67 (BatchNo	(None, 17, 17, 192)	576	conv2d_67[0][0]
activation_62 (Activation)	(None, 17, 17, 192)	0	batch_normalization_62[0][0]
activation_67 (Activation)	(None, 17, 17, 192)	0	batch_normalization_67[0][0]
average_pooling2d_6 (AveragePoo	(None, 17, 17, 768)	0	mixed6[0][0]
conv2d_60 (Conv2D)	(None, 17, 17, 192)	147456	mixed6[0][0]
conv2d_63 (Conv2D)	(None, 17, 17, 192)	258048	activation_62[0][0]
conv2d_68 (Conv2D)	(None, 17, 17, 192)	258048	activation_67[0][0]
conv2d_69 (Conv2D)	(None, 17, 17, 192)	147456	average_pooling2d_6[0][0]
batch_normalization_60 (BatchNo	(None, 17, 17, 192)	576	conv2d_60[0][0]
batch_normalization_63 (BatchNo	(None, 17, 17, 192)	576	conv2d_63[0][0]
batch_normalization_68 (BatchNo	(None, 17, 17, 192)	576	conv2d_68[0][0]
batch_normalization_69 (BatchNo	(None, 17, 17, 192)	576	conv2d_69[0][0]
activation_60 (Activation)	(None, 17, 17, 192)	0	batch_normalization_60[0][0]
activation_63 (Activation)	(None, 17, 17, 192)	0	batch_normalization_63[0][0]
activation_68 (Activation)	(None, 17, 17, 192)	0	batch_normalization_68[0][0]
activation_69 (Activation)	(None, 17, 17, 192)	0	batch_normalization_69[0][0]
mixed7 (Concatenate)	(None, 17, 17, 768)	0	activation_60[0][0] activation_63[0][0] activation_68[0][0] activation_69[0][0]
conv2d_72 (Conv2D)	(None, 17, 17, 192)	147456	mixed7[0][0]
batch_normalization_72 (BatchNo	(None, 17, 17, 192)	576	conv2d_72[0][0]
activation_72 (Activation)	(None, 17, 17, 192)	0	batch_normalization_72[0][0]
conv2d_73 (Conv2D)	(None, 17, 17, 192)	258048	activation_72[0][0]
batch_normalization_73 (BatchNo	(None, 17, 17, 192)	576	conv2d_73[0][0]
activation_73 (Activation)	(None, 17, 17, 192)	0	batch_normalization_73[0][0]
conv2d_70 (Conv2D)	(None, 17, 17, 192)	147456	mixed7[0][0]
conv2d_74 (Conv2D)	(None, 17, 17, 192)	258048	activation_73[0][0]
batch_normalization_70 (BatchNo	(None, 17, 17, 192)	576	conv2d_70[0][0]
batch_normalization_74 (BatchNo	(None, 17, 17, 192)	576	conv2d_74[0][0]
activation_70 (Activation)	(None, 17, 17, 192)	0	batch_normalization_70[0][0]
activation_74 (Activation)	(None, 17, 17, 192)	0	batch_normalization_74[0][0]
conv2d_71 (Conv2D)	(None, 8, 8, 320)	552960	activation_70[0][0]
conv2d_75 (Conv2D)	(None, 8, 8, 192)	331776	activation_74[0][0]
batch_normalization_71 (BatchNo	(None, 8, 8, 320)	960	conv2d_71[0][0]
batch_normalization_75 (BatchNo	(None, 8, 8, 192)	576	conv2d_75[0][0]
activation_71 (Activation)	(None, 8, 8, 320)	0	batch_normalization_71[0][0]
activation_75 (Activation)	(None, 8, 8, 192)	0	batch_normalization_75[0][0]

activation_75 (Activation) (None, 8, 8, 192) 0 batch_normalization_75[0][0]

max_pooling2d_3 (MaxPooling2D) (None, 8, 8, 768) 0 mixed7[0][0]

mixed8 (Concatenate) (None, 8, 8, 1280) 0 activation_71[0][0]
activation_75[0][0]
max_pooling2d_3[0][0]

conv2d_80 (Conv2D) (None, 8, 8, 448) 573440 mixed8[0][0]

batch_normalization_80 (BatchNo (None, 8, 8, 448) 1344 conv2d_80[0][0]

activation_80 (Activation) (None, 8, 8, 448) 0 batch_normalization_80[0][0]

conv2d_77 (Conv2D) (None, 8, 8, 384) 491520 mixed8[0][0]

conv2d_81 (Conv2D) (None, 8, 8, 384) 1548288 activation_80[0][0]

batch_normalization_77 (BatchNo (None, 8, 8, 384) 1152 conv2d_77[0][0]

batch_normalization_81 (BatchNo (None, 8, 8, 384) 1152 conv2d_81[0][0]

activation_77 (Activation) (None, 8, 8, 384) 0 batch_normalization_77[0][0]

activation_81 (Activation) (None, 8, 8, 384) 0 batch_normalization_81[0][0]

conv2d_78 (Conv2D) (None, 8, 8, 384) 442368 activation_77[0][0]

conv2d_79 (Conv2D) (None, 8, 8, 384) 442368 activation_77[0][0]

conv2d_82 (Conv2D) (None, 8, 8, 384) 442368 activation_81[0][0]

conv2d_83 (Conv2D) (None, 8, 8, 384) 442368 activation_81[0][0]

average_pooling2d_7 (AveragePoo (None, 8, 8, 1280) 0 mixed8[0][0]

conv2d_76 (Conv2D) (None, 8, 8, 320) 409600 mixed8[0][0]

batch_normalization_78 (BatchNo (None, 8, 8, 384) 1152 conv2d_78[0][0]

batch_normalization_79 (BatchNo (None, 8, 8, 384) 1152 conv2d_79[0][0]

batch_normalization_82 (BatchNo (None, 8, 8, 384) 1152 conv2d_82[0][0]

batch_normalization_83 (BatchNo (None, 8, 8, 384) 1152 conv2d_83[0][0]

conv2d_84 (Conv2D) (None, 8, 8, 192) 245760 average_pooling2d_7[0][0]

batch_normalization_76 (BatchNo (None, 8, 8, 320) 960 conv2d_76[0][0]

activation_78 (Activation) (None, 8, 8, 384) 0 batch_normalization_78[0][0]

activation_79 (Activation) (None, 8, 8, 384) 0 batch_normalization_79[0][0]

activation_82 (Activation) (None, 8, 8, 384) 0 batch_normalization_82[0][0]

activation_83 (Activation) (None, 8, 8, 384) 0 batch_normalization_83[0][0]

batch_normalization_84 (BatchNo (None, 8, 8, 192) 576 conv2d_84[0][0]

activation_76 (Activation) (None, 8, 8, 320) 0 batch_normalization_76[0][0]

mixed9_0 (Concatenate) (None, 8, 8, 768) 0 activation_78[0][0]
activation_79[0][0]

concatenate (Concatenate) (None, 8, 8, 768) 0 activation_82[0][0]
activation_83[0][0]

activation_84 (Activation) (None, 8, 8, 192) 0 batch_normalization_84[0][0]

mixed9 (Concatenate) (None, 8, 8, 2048) 0 activation_76[0][0]
mixed9_0[0][0]
concatenate[0][0]
activation_84[0][0]

conv2d_89 (Conv2D) (None, 8, 8, 448) 917504 mixed9[0][0]

batch_normalization_89 (BatchNo (None, 8, 8, 448) 1344 conv2d_89[0][0]

activation_89 (Activation) (None, 8, 8, 448) 0 batch_normalization_89[0][0]

conv2d_86 (Conv2D) (None, 8, 8, 384) 786432 mixed9[0][0]

```
conv2d_90 (Conv2D)              (None, 8, 8, 384)    1548288     activation_89[0][0]
__________________________________________________________________________________________________
batch_normalization_86 (BatchNo (None, 8, 8, 384)    1152        conv2d_86[0][0]
__________________________________________________________________________________________________
batch_normalization_90 (BatchNo (None, 8, 8, 384)    1152        conv2d_90[0][0]
__________________________________________________________________________________________________
activation_86 (Activation)      (None, 8, 8, 384)    0           batch_normalization_86[0][0]
__________________________________________________________________________________________________
activation_90 (Activation)      (None, 8, 8, 384)    0           batch_normalization_90[0][0]
__________________________________________________________________________________________________
conv2d_87 (Conv2D)              (None, 8, 8, 384)    442368      activation_86[0][0]
__________________________________________________________________________________________________
conv2d_88 (Conv2D)              (None, 8, 8, 384)    442368      activation_86[0][0]
__________________________________________________________________________________________________
conv2d_91 (Conv2D)              (None, 8, 8, 384)    442368      activation_90[0][0]
__________________________________________________________________________________________________
conv2d_92 (Conv2D)              (None, 8, 8, 384)    442368      activation_90[0][0]
__________________________________________________________________________________________________
average_pooling2d_8 (AveragePoo (None, 8, 8, 2048)   0           mixed9[0][0]
__________________________________________________________________________________________________
conv2d_85 (Conv2D)              (None, 8, 8, 320)    655360      mixed9[0][0]
__________________________________________________________________________________________________
batch_normalization_87 (BatchNo (None, 8, 8, 384)    1152        conv2d_87[0][0]
__________________________________________________________________________________________________
batch_normalization_88 (BatchNo (None, 8, 8, 384)    1152        conv2d_88[0][0]
__________________________________________________________________________________________________
batch_normalization_91 (BatchNo (None, 8, 8, 384)    1152        conv2d_91[0][0]
__________________________________________________________________________________________________
batch_normalization_92 (BatchNo (None, 8, 8, 384)    1152        conv2d_92[0][0]
__________________________________________________________________________________________________
conv2d_93 (Conv2D)              (None, 8, 8, 192)    393216      average_pooling2d_8[0][0]
__________________________________________________________________________________________________
batch_normalization_85 (BatchNo (None, 8, 8, 320)    960         conv2d_85[0][0]
__________________________________________________________________________________________________
activation_87 (Activation)      (None, 8, 8, 384)    0           batch_normalization_87[0][0]
__________________________________________________________________________________________________
activation_88 (Activation)      (None, 8, 8, 384)    0           batch_normalization_88[0][0]
__________________________________________________________________________________________________
activation_91 (Activation)      (None, 8, 8, 384)    0           batch_normalization_91[0][0]
__________________________________________________________________________________________________
activation_92 (Activation)      (None, 8, 8, 384)    0           batch_normalization_92[0][0]
__________________________________________________________________________________________________
batch_normalization_93 (BatchNo (None, 8, 8, 192)    576         conv2d_93[0][0]
__________________________________________________________________________________________________
activation_85 (Activation)      (None, 8, 8, 320)    0           batch_normalization_85[0][0]
__________________________________________________________________________________________________
mixed9_1 (Concatenate)          (None, 8, 8, 768)    0           activation_87[0][0]
                                                                 activation_88[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 8, 8, 768)    0           activation_91[0][0]
                                                                 activation_92[0][0]
__________________________________________________________________________________________________
activation_93 (Activation)      (None, 8, 8, 192)    0           batch_normalization_93[0][0]
__________________________________________________________________________________________________
mixed10 (Concatenate)           (None, 8, 8, 2048)   0           activation_85[0][0]
                                                                 mixed9_1[0][0]
                                                                 concatenate_1[0][0]
                                                                 activation_93[0][0]
__________________________________________________________________________________________________
avg_pool (GlobalAveragePooling2 (None, 2048)         0           mixed10[0][0]
__________________________________________________________________________________________________
predictions (Dense)             (None, 1000)         2049000     avg_pool[0][0]
==================================================================================================
Total params: 23,851,784
Trainable params: 23,817,352
Non-trainable params: 34,432
__________________________________________________________________________________________________
```

In [0]:

```
model = Model(base_model.input, base_model.layers[-2].output)
```

In [35]:

```
from keras.preprocessing.image import load_img, img_to_array
def preprocess_img(img_path):
    # inception v3 excepts img in 299*299
    img = load_img(img_path, target_size = (299, 299))
    x = img_to_array(img)
    # Add one more dimension
    x = np.expand_dims(x, axis = 0)
```

```
    x = np.expand_dims(x, axis = 0)
    x = preprocess_input(x)
    return x
```

```
Using TensorFlow backend.
```

In [0]:

```
# function to encode an image into a vector
def encode(image):
  image = preprocess_img(image)
  vec = model.predict(image)
  vec = np.reshape(vec, (vec.shape[1]))
  return vec
```

In [37]:

```
# run the encode function on all train images in a dictionary {image: 2048 vector size}

start = time.time()

encoding_train = {}
for img in train_images:
  # img(len(path)): means to start from there (discard the path and hence remains image name)
  encoding_train[img[len(path):]] = encode(img)

print("Time Taken is: " + str(time.time() - start))
```

```
Time Taken is: 237.73764514923096
```

In [38]:

```
print('The length of encode vector which is value:', len(list(encoding_train.values())[0]))
```

```
The length of encode vector which is value: 2048
```

In [39]:

```
# Just to ensure the sequence to dictionary matches with sequence of images
a= list(encoding_train.items())[:2]
a
```

Out[39]:

```
[('1000092795.jpg',
  array([0.18249522, 0.16290566, 0.522696  , ..., 0.6769272 , 0.32858288,
         0.08636494], dtype=float32)),
 ('10002456.jpg',
  array([0.5403669 , 0.11955979, 0.03325099, ..., 1.0153631 , 0.02305902,
         0.7214722 ], dtype=float32))]
```

In [40]:

```
# test images

start = time.time()

encoding_test = {}
for img in test_images:
    encoding_test[img[len(path):]] = encode(img)

print("Time Taken is: " + str(time.time() - start))

print(len(list(encoding_test.values())[0]))
```

```
Time Taken is: 12.111755847930908
2048
```

In [41]:

```
print("Train image encodings: " + str(len(encoding_train)))
print("Test image encodings: " + str(len(encoding_test)))
```

```
Train image encodings: 5700
```

Train image encodings: 3700
Test image encodings: 300

In [42]:

```
start= time.time()

# Our Input shall be 2048 (dense) + [80 (word_to_ix padded) * 300 (w2v) = 24000] = 26048 and
# final size of the data matrix is 602629 * 26048= 1.57 * e^10 blocks hence below data generator function
# 47500 captions * 12.7 (avg words in a caption) = 602629
# output shall be scalar among 3693 (vocab)

X1, X2, y = list(), list(), list()

for key, des_list in new_desc.items():
    pic = encoding_train[key + '.jpg']
    for cap in des_list:
        seq = [word_to_ix[word] for word in cap.split(' ') if word in word_to_ix]
        for i in range(1, len(seq)):
            in_seq, out_seq = seq[:i], seq[i]
            in_seq = pad_sequences([in_seq], maxlen = max(l))[0]
            out_seq = to_categorical([out_seq], num_classes = vocab_size)[0]
            #store
            X1.append(pic)
            X2.append(in_seq)
            y.append(out_seq)

X1 = np.array(X1)
X2 = np.array(X2)
y = np.array(y)
print('The shape of the Image vector(output from Inception network): ', X1.shape)

print('The shape of the maximum length padded sequence vector: ', X2.shape)

print('The unique vocabulary: ', y.shape)

print("Time Taken is: " + str(time.time() - start))
```

The shape of the Image vector(output from Inception network): (359954, 2048)
The shape of the maximum length padded sequence vector: (359954, 80)
The unique vocabulary: (359954, 2831)
Time Taken is: 170.56092047691345

In [43]:

```
!wget --header="Host: downloads.cs.stanford.edu" --header="User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko ) Chrome/76.0.3809.132 Safari/537.36" --header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,a pplication/signed-exchange;v=b3" --header="Accept-Language: en-GB,en-US;q=0.9,en;q=0.8" --header="Cookie: _ga=GA1.2.875242895.15770231 56; _gid=GA1.2.2102497446.1577023156; _gat=1" --header="Connection: keep-alive" "http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip" -O "g love.6B.zip" -c
```

--2020-04-04 18:47:50-- http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)... 171.64.64.22
Connecting to downloads.cs.stanford.edu (downloads.cs.stanford.edu)|171.64.64.22|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 862182613 (822M) [application/zip]
Saving to: ‘glove.6B.zip’

glove.6B.zip   100%[===================>] 822.24M 1.99MB/s   in 6m 27s

2020-04-04 18:54:18 (2.12 MB/s) - ‘glove.6B.zip’ saved [862182613/862182613]

In [44]:

```
from zipfile import ZipFile
file_name = "glove.6B.zip"

# opening the zip file in READ mode
with ZipFile(file_name, 'r') as zip:
    # printing all the contents of the zip file
    zip.printdir()

    # extracting all the files
    print('Extracting all the files now...')
    zip.extractall()
    print('Done!')
```

```
File Name                                             Modified             Size
glove.6B.50d.txt                                    2014-08-04 13:15:00    171350079
glove.6B.100d.txt                                   2014-08-04 13:14:34    347116733
glove.6B.200d.txt                                   2014-08-04 13:14:44    693432828
glove.6B.300d.txt                                   2014-08-27 12:19:16   1037962819
Extracting all the files now...
Done!
```

In [45]:

```
#load glove vectors for embedding layer
embeddings_index = {}
glove = open('glove.6B.300d.txt', 'r', encoding = 'utf-8').read()
for line in glove.split("\n"):
    values = line.split(" ")
    word = values[0]
    indices = np.asarray(values[1: ], dtype = 'float32')
    embeddings_index[word] = indices
print('Total word vectors: ' + str(len(embeddings_index)))
```

```
Total word vectors: 400001
```

In [46]:

```
emb_vec = embeddings_index.get('two')
print(emb_vec.shape)
```

```
(300,)
```

In [47]:

```
emb_vec = embeddings_index.get('endseq')
print(emb_vec)
```

```
None
```

In [48]:

```
emb_dim = 300
emb_matrix = np.zeros((vocab_size, emb_dim))
emb_matrix.shape
```

Out[48]:

```
(2831, 300)
```

In [82]:

```
from tqdm import tqdm
for word, i in tqdm(enumerate(word_to_ix)):
  emb_vec = embeddings_index.get(word)
  if emb_vec is not None:
    emb_matrix[i] = emb_vec

emb_matrix.shape
```

```
2830it [00:00, 601554.85it/s]
```

Out[82]:

```
(2831, 300)
```

In [84]:

```
# a sample output
emb_matrix[10]
```

Out[84]:

```
array([-7.06010014e-02,  5.49090028e-01, -2.55360007e-01, -1.65439993e-01,
       -3.44640017e-02,  2.91489989e-01, -1.23630002e-01,  3.48080009e-01,
        1.24080002e-01, -1.72790003e+00, -4.13489997e-01, -8.40670019e-02,
       -5.36409974e-01, -2.92789996e-01, -3.71349990e-01,  2.18899995e-01,
```

```
 8.35399985e-01, -2.47569993e-01, -3.65240008e-01, -2.29939997e-01,
 2.10429996e-01, -1.47120003e-02,  9.84990001e-02,  5.19759990e-02,
 1.59189999e-01, -2.32639998e-01,  9.04410034e-02,  2.08460003e-01,
-4.00150001e-01,  2.23529994e-01, -1.22199997e-01,  2.10360005e-01,
-1.76829994e-01,  6.32990003e-02, -1.04779994e+00,  3.56709987e-01,
 4.15420011e-02,  2.11099997e-01,  3.39379996e-01, -3.03990006e-01,
 1.84699997e-01, -1.56599998e-01, -2.59579986e-01,  2.20599994e-01,
 3.11369985e-01,  6.27820015e-01, -1.40650004e-01,  5.00699997e-01,
-2.23810002e-02, -1.96040004e-01,  4.37559992e-01, -6.84360027e-01,
 4.73630009e-03, -4.14240003e-01,  1.22349998e-02,  3.91189992e-01,
 2.58179996e-02,  3.75290006e-01, -3.38189989e-01,  4.11850005e-01,
 1.62630007e-01, -2.97089994e-01,  1.79169998e-01,  4.65339988e-01,
 5.44469990e-02, -4.30830002e-01,  2.71919996e-01,  2.33199999e-01,
 1.96899995e-02, -3.54349986e-02,  1.04240000e-01,  1.91990003e-01,
 8.33010003e-02,  2.84370005e-01,  1.71159998e-01,  1.41980007e-01,
-5.37590012e-02, -8.26639980e-02, -8.65840018e-02, -3.43070000e-01,
-3.82789999e-01,  4.74209994e-01,  1.18119996e-02, -1.43409997e-01,
 1.62459999e-01,  2.80680005e-02, -2.40369998e-02, -4.34350014e-01,
 1.10979997e-01,  3.04500014e-01, -1.44470006e-01,  7.33219981e-02,
 3.14220011e-01,  9.15720034e-03, -3.66600007e-01,  9.21360031e-02,
-2.61909992e-01, -1.15670003e-01,  2.27300003e-01, -4.56189990e-01,
 2.60019988e-01,  3.35070007e-02,  5.22629991e-02,  2.67630011e-01,
-2.20559999e-01, -3.09729993e-01,  4.80399996e-01, -2.74730008e-02,
-6.69879988e-02, -1.46369994e-01,  1.38490006e-01, -7.78739974e-02,
-3.62630010e-01,  2.67120004e-01, -3.09570003e-02,  3.05029988e-01,
-1.71609998e-01,  4.08109985e-02,  4.50830013e-01, -5.00949979e-01,
-1.69379994e-01, -6.53289974e-01,  1.89590007e-01,  2.14800000e-01,
-2.78459996e-01,  1.35230005e-01, -3.00309986e-01, -5.41329980e-01,
-3.37640010e-03, -4.46639992e-02, -4.21550013e-02,  4.25570011e-01,
-3.94410007e-02,  3.04450002e-02,  2.37310007e-02,  2.31319994e-01,
 6.26420021e-01, -4.53960001e-01,  1.86829999e-01,  3.77680004e-01,
-1.80179998e-01, -1.04350001e-01,  2.76969999e-01,  1.55790001e-01,
 9.73519981e-02, -7.21589997e-02, -2.70240009e-02,  2.57939994e-02,
 3.09410006e-01,  1.03989998e-02,  5.66049993e-01,  7.16520008e-03,
 3.74930017e-02, -1.09389998e-01,  5.57579994e-01,  2.38550007e-01,
-5.11849999e-01, -1.19180001e-01,  3.26950014e-01, -2.86749989e-01,
 3.07960004e-01, -1.22599997e-01,  9.52610001e-02,  2.59339988e-01,
-1.31750003e-01,  4.76960003e-01, -2.56069988e-01, -2.78060008e-02,
 9.38249975e-02,  6.41900003e-02,  1.81510001e-01,  3.36569995e-01,
-8.32539976e-01,  1.80390000e-01, -1.54039994e-01, -6.06029993e-03,
-1.78510007e-02,  4.15010005e-01, -2.29359999e-01,  8.38620007e-01,
 4.44750004e-02, -3.14749986e-01,  3.64760011e-01,  7.65969992e-01,
-3.13910007e-01,  5.09720027e-01,  1.10500000e-01,  1.17720000e-01,
 5.00109971e-01, -3.87380004e-01, -4.28289995e-02, -3.71780008e-01,
-2.31710002e-01, -9.29820016e-02,  8.56669992e-02, -1.02219999e-01,
 2.17869997e-01,  1.73820004e-01,  3.20620000e-01,  1.99019998e-01,
 9.82400000e-01, -3.83690000e-01,  2.95760006e-01,  2.55190015e-01,
 3.82239997e-01, -2.95700014e-01,  2.05729995e-03,  4.52120006e-01,
-4.22939986e-01,  2.33339995e-01, -1.46109998e-01,  1.57350004e-01,
-3.42759997e-01,  3.97269994e-01,  1.96710005e-01,  2.60230005e-01,
 6.88820004e-01,  3.35680008e-01,  3.51560004e-02, -6.53769970e-02,
-4.26390022e-03, -1.94629997e-01, -3.98840010e-01,  7.47980028e-02,
-4.08210009e-01,  2.31889993e-01,  1.65639997e-01,  2.96970010e-01,
-6.12079978e-01, -3.12750012e-01, -1.25960007e-01,  2.04630002e-01,
-2.35699996e-01, -2.78019994e-01,  1.71849996e-01, -7.11570010e-02,
-3.90379988e-02,  1.13720000e-01, -6.31420016e-01,  2.72370011e-01,
 1.03100002e-01,  3.22759986e-01,  8.18480015e-01,  7.69369975e-02,
-1.18780005e+00,  1.33530006e-01,  2.67049998e-01,  2.23729998e-01,
 2.61189997e-01,  1.78169996e-01, -7.88210034e-02, -8.28550011e-02,
 4.15609986e-01, -1.83579996e-01,  3.59109998e-01,  4.79310006e-01,
 1.34069994e-01,  4.02920008e-01, -4.01849985e-01,  4.37079996e-01,
 1.26660004e-01, -4.28380013e-01,  2.95199990e-01,  7.80280009e-02,
-1.62760004e-01,  4.50029999e-01, -5.41360021e-01, -8.87409985e-01,
-2.62259990e-01, -9.61909965e-02, -5.85289998e-03,  2.19300002e-01,
 2.22540006e-01, -2.30959997e-01, -2.10400000e-01, -4.19349998e-01,
-2.96000004e+00,  1.96780004e-02,  7.01680005e-01,  1.48800001e-01,
-6.60009980e-02,  3.08880001e-01, -4.26970005e-01,  5.28330028e-01,
-3.86640012e-01,  1.35700002e-01,  3.60590011e-01,  2.07289994e-01,
 3.82279992e-01,  6.16619997e-02, -4.11179990e-01, -3.10889989e-01,
-4.38069999e-02, -3.24019998e-01, -7.70720020e-02,  5.47739983e-01,
-1.13389999e-01, -4.25399989e-01, -2.50389993e-01, -2.04359993e-01])
```

In [70]:

```
# Here we cannot use Sequential API and we use Functional API which allows us to merge models

# Defining the Model-1 (Image)
ip1 = Input(shape = (2048, ))

fe1 = Dropout(0.3)(ip1)
fe2 = Dense(512, activation = 'relu')(fe1)
```

```
# Defining Model-II (24048 embedding vectors), max(l)= 80
ip2 = Input(shape = (max(l), ))

# Masking= True works as 'ffill' when a zero is occurred in an array.
# https://stackoverflow.com/a/53470422/10219869
se1 = Embedding(vocab_size, emb_dim, mask_zero= True)(ip2)
se2 = Bidirectional(LSTM(256) )(se1)
se3 = Dropout(0.3)(se2)

# This is where we merge the models
decoder1 = tf.keras.layers.add([fe2, se3])
decoder2 = Dense(256, activation = 'relu')(decoder1)

outputs = Dense(vocab_size, activation = 'softmax')(decoder2)

model = Model(inputs = [ip1, ip2], outputs = outputs)
model.summary()
```

```
Model: "model_4"
__________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================================
input_13 (InputLayer)        [(None, 80)]              0
__________________________________________________________________________________________________
input_12 (InputLayer)        [(None, 2048)]            0
__________________________________________________________________________________________________
embedding_5 (Embedding)      (None, 80, 300)           849300      input_13[0][0]
__________________________________________________________________________________________________
dropout_10 (Dropout)         (None, 2048)              0           input_12[0][0]
__________________________________________________________________________________________________
bidirectional_3 (Bidirectional) (None, 512)            1140736     embedding_5[0][0]
__________________________________________________________________________________________________
dense_11 (Dense)             (None, 512)               1049088     dropout_10[0][0]
__________________________________________________________________________________________________
dropout_11 (Dropout)         (None, 512)               0           bidirectional_3[0][0]
__________________________________________________________________________________________________
add_5 (Add)                  (None, 512)               0           dense_11[0][0]
                                                                   dropout_11[0][0]
__________________________________________________________________________________________________
dense_12 (Dense)             (None, 256)               131328      add_5[0][0]
__________________________________________________________________________________________________
dense_13 (Dense)             (None, 2831)              727567      dense_12[0][0]
==================================================================================================
Total params: 3,898,019
Trainable params: 3,898,019
Non-trainable params: 0
__________________________________________________________________________________________________
```

In [52]:

```
model.layers[2]
```

Out[52]:

```
<tensorflow.python.keras.layers.embeddings.Embedding at 0x7fe77ddb2f60>
```

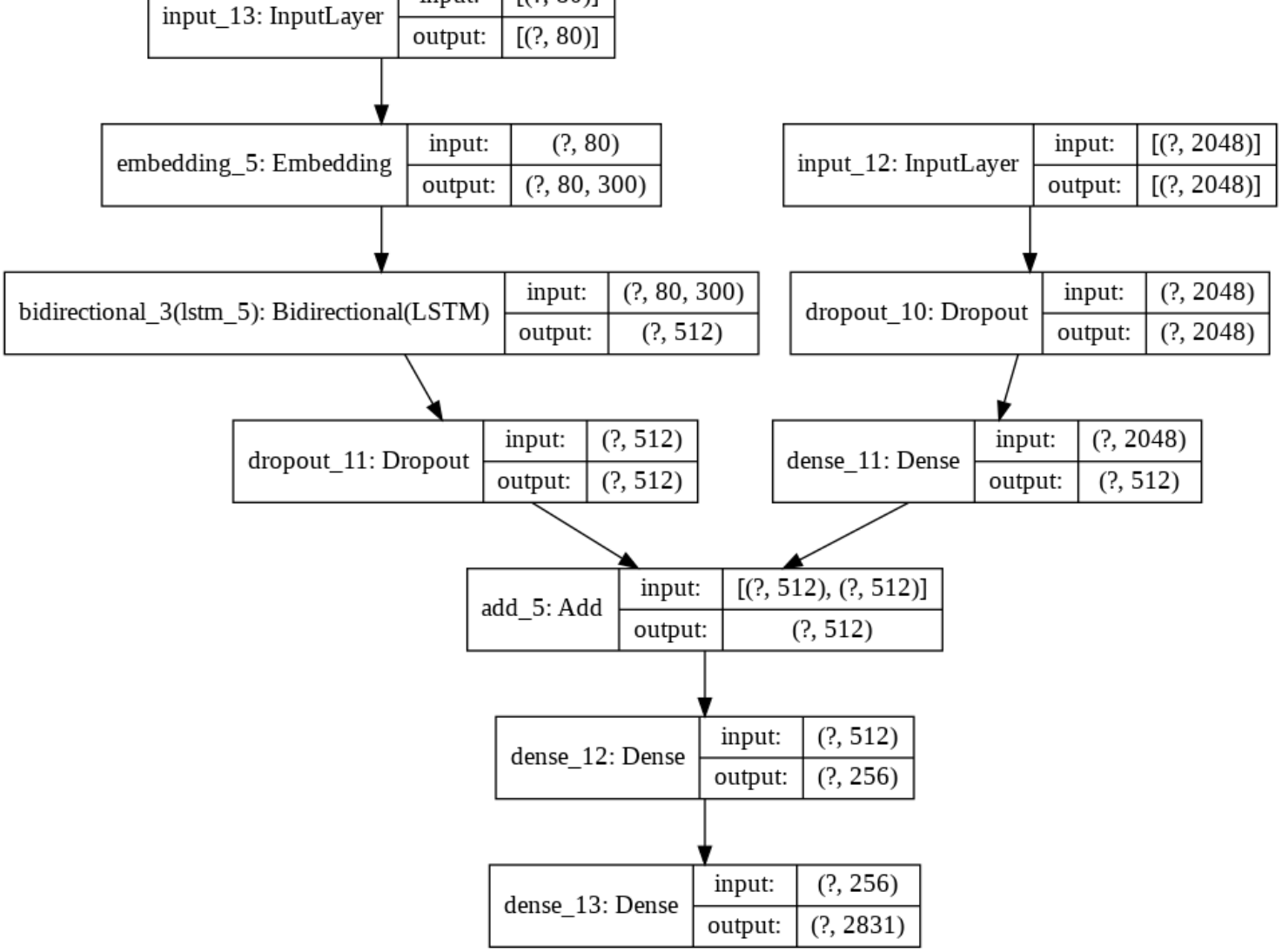
In [0]:

```
model.layers[2].set_weights([emb_matrix]) # we can see this in above summary [2] means third position
model.layers[2].trainable = False
adam=  tf.keras.optimizers.Adam(learning_rate = 0.0001)
model.compile(loss = 'categorical_crossentropy', optimizer = adam)
```

In [72]:

```
tf.keras.utils.plot_model(
    model,
    to_file='model_1.png',
    show_shapes=True,
    show_layer_names=True,
    rankdir='TB',
    expand_nested=False,
    dpi=96
)
```

Out[72]:



In [86]:

```
start= time.time()

for i in range(20):
    model.fit([X1, X2], y, epochs = 1, batch_size = 512)
    if(i%2 == 0):
        model.save_weights("image-caption-weights_new" + str(i) + ".h5")

print("Time Taken is: " + str(time.time() - start))
```

```
704/704 [==============================] - 186s 265ms/step - loss: 4.2954
704/704 [==============================] - 186s 264ms/step - loss: 3.8733
704/704 [==============================] - 187s 265ms/step - loss: 3.6368
704/704 [==============================] - 187s 265ms/step - loss: 3.5040
704/704 [==============================] - 187s 265ms/step - loss: 3.4126
704/704 [==============================] - 187s 266ms/step - loss: 3.3354
704/704 [==============================] - 187s 265ms/step - loss: 3.2743
704/704 [==============================] - 187s 266ms/step - loss: 3.2204
704/704 [==============================] - 187s 266ms/step - loss: 3.1773
704/704 [==============================] - 188s 267ms/step - loss: 3.1339
704/704 [==============================] - 189s 269ms/step - loss: 3.0964
704/704 [==============================] - 189s 268ms/step - loss: 3.0616
704/704 [==============================] - 189s 269ms/step - loss: 3.0294
704/704 [==============================] - 189s 269ms/step - loss: 3.0007
704/704 [==============================] - 190s 269ms/step - loss: 2.9734
704/704 [==============================] - 190s 269ms/step - loss: 2.9455
704/704 [==============================] - 189s 269ms/step - loss: 2.9214
704/704 [==============================] - 189s 269ms/step - loss: 2.8975
704/704 [==============================] - 190s 270ms/step - loss: 2.8742
704/704 [==============================] - 189s 268ms/step - loss: 2.8533
Time Taken is: 3774.62273645401
```

In [0]:

```
'''
The model generates a n-long vector(while 2831 - long vector in the original example) which is a probability distribution
across all the words in the vocabulary. For this reason we greedily select the word
with the maximum probability, given the feature vector and partial caption.
```

```
With the maximum probability, given the feature vector and partial caption.
This is called as Maximum Likelihood Estimation (MLE)
'''

def greedy_search(pic):
    start = 'startseq'
    for i in range(max(l)):
        seq = [word_to_ix[word] for word in start.split() if word in word_to_ix]
        seq = pad_sequences([seq], maxlen = max(l))
        yhat = model.predict([pic, seq])
        yhat = np.argmax(yhat)                    # max probability of 3693 words are preserved based on caption length words.
        word = ix_to_word[yhat]
        start += ' ' + word
        if word == 'endseq':
            break
    final = start.split()
    final = final[1:-1]
    final = ' '.join(final)
    return final
```

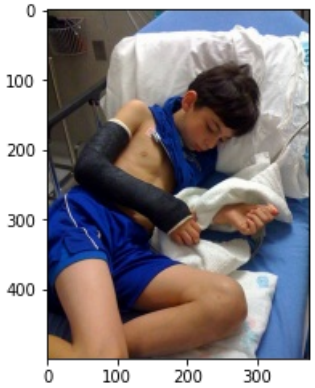
In [87]:

```
pic = list(encoding_test.keys())[202]
img = encoding_test[pic].reshape(1, 2048)
x = plt.imread(path + pic)
plt.imshow(x)
plt.show()
print(greedy_search(img))
```

a young boy in a red uniform is playing a soccer ball

In [88]:

```
pic = list(encoding_test.keys())[209]
img = encoding_test[pic].reshape(1, 2048)
x = plt.imread(path + pic)
plt.imshow(x)
plt.show()
print(greedy_search(img))
```



a young boy in a blue shirt and blue shorts is sleeping on a chair

In [92]:

```
pic = list(encoding_test.keys())[99]
img = encoding_test[pic].reshape(1, 2048)
```

```
x = plt.imread(path + pic)
plt.imshow(x)
plt.show()
print(greedy_search(img))
```

a group of people are sitting on a picnic

In [0]:

```
```