English -Malayalam Multimodal Machine Translation-5000 images

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
import cv2
import matplotlib.image as mp
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
     Mounted at /content/drive
```

```
with open('/content/drive/My Drive/Main/train.mn.txt') as file:
    mal_txt = file.read().split('\n')
with open('/content/drive/My Drive/Main/train.en.txt') as file:
    eng_txt = file.read().split('\n')
with open('/content/drive/My Drive/Main/train_images.txt') as file:
    train_images = file.read().split('\n')
```

```
train_images[-1]
```

```
     ''
```

```
#removing last elements which containing special characters
mal_txt.pop()
mal_txt.pop()
eng_txt.pop()
eng_txt.pop()
train_images.pop()
print(len(mal_txt))
print(len(eng_txt))
print(len(train_images))
img_path=[]
for s in train_images:
    img_path.append("/content/drive/My Drive/Main/trainimages/train/"+s)
```

```
     28930
     28931
     28931
```

```
print(mal_txt[1])
print(eng_txt[1])
```

```
ഇത് ഒരു ഇൻഡോർ രംഗമാണ്
it is an indoor scene
```

```
mal_txt[0:15]
```

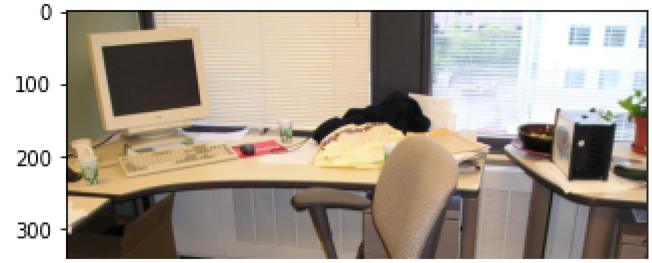
```
['ശാന്തമായ  കടലിൽ സർഫിങ് നടത്തുന്ന പുരുഷ സർഫർ',
 'ഇത് ഒരു ഇൻഡോർ രംഗമാണ്',
 'കമ്പ്യൂട്ടർ സ്ക്രീനുകൾ ഓണാക്കി',
 'മനുഷ്യന് ചെറിയ മുടിയുണ്ട്',
 'ഫോട്ടോ ആൽബം മുതിർന്നവരുടെ മടിയിൽ തുറക്കുന്നു',
 'കറുത്ത കാറിനടുത്ത് ഒരു കൂട്ടം പെൺകുട്ടികളുണ്ട്',
 'ഒരു ഉന്തുവണ്ടിയിലെ കുട്ടി',
 'ഉയരമുള്ള മെറ്റൽ ലൈറ്റ്പോസ്റ്റ്',
 'മതിൽ വെളുത്ത ചായം പൂശി',
 'ചാരനിറത്തിലുള്ള റോഡിന്റെ വശങ്ങളിൽ പച്ച പുല്ലിന്റെ സ്ട്രിപ്പുകൾ',
 'സമുദ്രം അഭിമുഖീകരിക്കുന്ന സ്ത്രീ',
 'ഇതൊരു ഓഫീസ് രേഖാചിത്രം',
 'നാല് ലോഹത്തിന്റെ കസേരകൾ',
 'കോലാഹലം ഒരു മേശപ്പുറത്താണ്',
 'ഒരു വെളുത്ത മൈക്രോവേവ് ഓവൻ']
```

```
eng_txt[0:15]
```

```
['Male surfer surfing in still in the ocean',
 'it is an indoor scene\t\t\t\t\t\t',
 'Computer screens turned on\t\t\t\t\t\t',
 'man has short hair\t\t\t\t\t\t',
 "photo album open on an adult's lap\t\t\t\t\t\t",
 'there is a group of girls beside the black car\t\t\t\t\t\t',
 'Child in a stroller\t\t\t\t\t\t',
 'Tall metal lightpost\t\t\t\t\t\t',
 'wall is painted white\t\t\t\t\t\t',
 'there are several pictures on the wall\t\t\t\t\t\t',
 'woman facing the ocean\t\t\t\t\t\t',
 'this is an office layout\t\t\t\t\t\t',
 'four metallic chairs\t\t\t\t\t\t',
 'Clutter is on a table\t\t\t\t\t\t',
 'a white microwave oven\t\t\t\t\t\t']
```

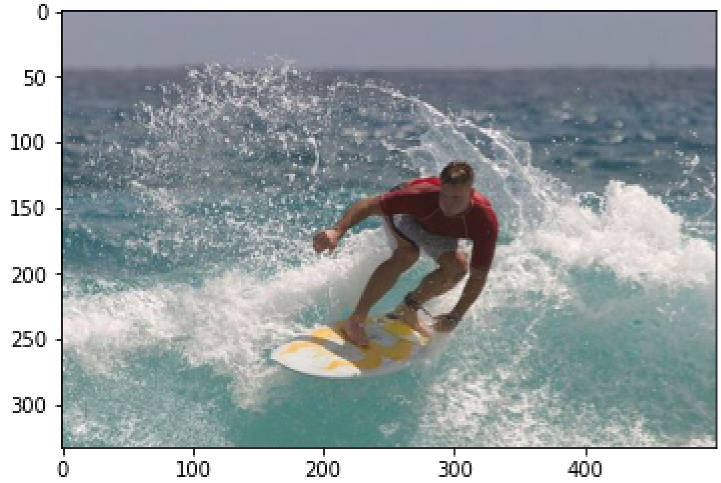
```
im=mp.imread(img_path[1])
plt.imshow(im)
print(img_path[1])
print("mal:"+mal_txt[1])
print("eng:"+eng_txt[1])
```

```
/content/drive/My Drive/Main/trainimages/train/11.jpg
mal:ഇത് ഒരു ഇൻഡോർ രംഗമാണ്
eng:it is an indoor scene
```



```
im=mp.imread(img_path[0])
plt.imshow(im)
print(img_path[0])
print("mal:"+mal_txt[0])
print("eng:"+eng_txt[0])
```

```
/content/drive/My Drive/Main/trainimages/train/10.jpg
mal:ശാന്തമായ  കടലിൽ സർഫിങ് നടത്തുന്ന പുരുഷ സർഫർ
eng:Male surfer surfing in still in the ocean
```



```
#Manually splitting data for training and texting-due to presence of 3 inputs split using Skl
splits=10000
mal_train=mal_txt[:splits]
eng_train=eng_txt[:splits]
```

```
mal_df = pd.DataFrame(mal_train, columns=['Malayalam'])
eng_df = pd.DataFrame(eng_train, columns=['English'])
```

```
mal_df.head(10)
```

	Malayalam
0	ശാന്തമായ കടലിൽ സർഫിങ് നടത്തുന്ന പുരുഷ സർഫർ
1	ഇത് ഒരു ഇൻഡോർ രംഗമാണ്
2	കമ്പ്യൂട്ടർ സ്ക്രീനുകൾ ഓണാക്കി
3	മനുഷ്യന് ചെറിയ മുടിയുണ്ട്
4	ഫോട്ടോ ആൽബം മുതിർന്നവരുടെ മടിയിൽ തുറക്കുന്നു
5	കറുത്ത കാറിനടുത്ത് ഒരു കൂട്ടം പെൺകുട്ടികളുണ്ട്
6	ഒരു ഉന്തുവണ്ടിയിലെ കുട്ടി
7	ഉയരമുള്ള മെറ്റൽ ലൈറ്റ്പോസ്റ്റ്
8	മതിൽ വെളുത്ത ചായം പൂശി

```
eng_df.head(10)
```

	English
0	Male surfer surfing in still in the ocean
1	it is an indoor scene\t\t\t\t\t\t
2	Computer screens turned on\t\t\t\t\t\t
3	man has short hair\t\t\t\t\t\t
4	photo album open on an adult's lap\t\t\t\t\t\t
5	there is a group of girls beside the black car...
6	Child in a stroller\t\t\t\t\t\t
7	Tall metal lightpost\t\t\t\t\t\t
8	wall is painted white\t\t\t\t\t\t
9	there are several pictures on the wall\t\t\t\t...

```
#Datacleaning by removing special characters
import re
def clean_text(text):
    text = text.lower()
    text = re.sub(r" ‘’", "", text)
    text = re.sub(r" ‘", "", text)
    text = re.sub(r"-", " ", text)
    text = re.sub(r"<5>", "5", text)
    text = re.sub(r"“ ", "", text)
    text = re.sub(r" ”", "", text)
    text = re.sub(r"[+\.\!\/_,$%^*(+\"\']+|[+——！，〈〉《》。｜？?、．%~@#￥%……&*（）’]", "", t
    text=text.rstrip()
    return text
```

```
mal_text1 = mal_df["Malayalam"].apply(clean_text)
eng_text1 = eng_df["English"].apply(clean_text)
mal_text2 = list(mal_text1.values)
eng_text2 = list(eng_text1.values)
```

```
#cleaned Malayalm data
mal_text1
```

```
0       ശാന്തമായ  കടലിൽ സർഫിങ് നടത്തുന്ന പുരുഷ സർഫർ
1                            ഇത് ഒരു ഇൻഡോർ രംഗമാണ്
2                        കമ്പ്യൂട്ടർ സ്ക്രീനുകൾ ഓണാക്കി
3                            മനുഷ്യന് ചെറിയ മുടിയുണ്ട്
4         ഫോട്ടോ ആൽബം മുതിർന്നവരുടെ മടിയിൽ തുറക്കുന്നു
                              ...
4995             ഫ്രിസ്ബീ ഉള്ള പുൽത്തകിടിയിൽ ഒരു നായ
4996        ഒരു ഉദ്യാനത്തിൽ ചാരനിറത്തിലുള്ള ലോഹ വേലി
4997                 തവിട്ടുനിറമുള്ള മുടിയുള്ള മനുഷ്യൻ
4998                         ഒരു ജിറാഫ് പുല്ല് തിന്നുന്നു
4999                          മുൻവശത്തുള്ള ഒരു പെൺകുട്ടി
Name: Malayalam, Length: 5000, dtype: object
```

```
#cleaned English data
eng_text1
```

```
0       male surfer surfing in still in the ocean
1                           it is an indoor scene
2                       computer screens turned on
3                               man has short hair
4                photo album open on an adults lap
                          ...
4995                a dog on a lawn with a frisbee
4996                  a gray metal fence in a park
4997                            a brown haired man
4998                            the floor is tiled
4999              a young girl in the foreground
Name: English, Length: 5000, dtype: object
```

```
mal_text2[1:5]
```

```
['ഇത് ഒരു ഇൻഡോർ രംഗമാണ്',
 'കമ്പ്യൂട്ടർ സ്ക്രീനുകൾ ഓണാക്കി',
 'മനുഷ്യന് ചെറിയ മുടിയുണ്ട്',
 'ഫോട്ടോ ആൽബം മുതിർന്നവരുടെ മടിയിൽ തുറക്കുന്നു']
```

```
eng_text2[1:5]
```

```
['it is an indoor scene',
 'computer screens turned on',
 'man has short hair',
 'photo album open on an adults lap']
```

```
#Adding starting and ending tokens
mal_temp=[]
for s in mal_text2:
    temp="sos "+s+" eos"
    mal_temp.append(temp)
#text2=[]
mal_text2=mal_temp
mal_text2[1:10]
```

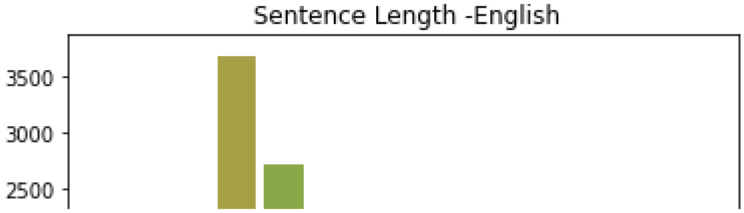
```
['sos ഇത് ഒരു ഇൻഡോർ രംഗമാണ് eos',
 'sos കമ്പ്യൂട്ടർ സ്ക്രീനുകൾ ഓണാക്കി eos',
 'sos മനുഷ്യന് ചെറിയ മുടിയുണ്ട് eos',
 'sos ഫോട്ടോ ആൽബം മുതിർന്നവരുടെ മടിയിൽ തുറക്കുന്നു eos',
 'sos കറുത്ത കാറിനടുത്ത് ഒരു കൂട്ടം പെൺകുട്ടികളുണ്ട് eos',
 'sos ഒരു ഉന്തുവണ്ടിയിലെ കുട്ടി eos',
 'sos ഉയരമുള്ള മെറ്റൽ ലൈറ്റ്പോസ്റ്റ് eos',
 'sos മതിൽ വെളുത്ത ചായം പൂശി eos',
 'sos ചാരനിറത്തിലുള്ള റോഡിന്റെ വശങ്ങളിൽ പച്ച പുല്ലിന്റെ സ്ട്രിപ്പുകൾ eos']
```

```
import seaborn as sn
import matplotlib.pyplot as plt
malayalam_words = []
for i in mal_text2:
    malayalam_words.append(len(i.split()))
sn.countplot(malayalam_words).set(title=' Sentence Length -Malayalam')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
```

```
english_words = []
for j in eng_text2:
    english_words.append(len(j.split()))
sn.countplot(english_words).set(title=' Sentence Length -English')
plt.show()
```

```
     /usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass tł
       FutureWarning
```



```
maxlen_malayalam = max(malayalam_words)
maxlen_english = max(english_words)
print('Maximum sentence length-Malayalam :',maxlen_malayalam)
print('Maximum sentence length-English :',maxlen_english)
```

```
     Maximum sentence length-Malayalam : 13
     Maximum sentence length-English : 14
```

```
#splitting training data into training and validation data
x_tr=eng_text2[:splits-500]
y_tr=mal_text2[:splits-500]
x_val=eng_text2[splits-500:]
y_val=mal_text2[splits-500:]
```

```
x_tr[1:5]
```

```
     ['it is an indoor scene',
      'computer screens turned on',
      'man has short hair',
      'photo album open on an adults lap']
```

```
y_tr[1:5]
```

```
     ['sos ഇത് ഒരു ഇൻഡോർ രംഗമാണ് eos',
      'sos കമ്പ്യൂട്ടർ സ്ക്രീനുകൾ ഓണാക്കി eos',
      'sos മനുഷ്യന് ചെറിയ മുടിയുണ്ട് eos',
      'sos ഫോട്ടോ ആൽബം മുതിർന്നവരുടെ മടിയിൽ തുറക്കുന്നു eos']
```

```
len(x_tr)
```

```
     9500
```

```
len(x_val)
```

```
     500
```

```
#Tokening the sentences using Keras tokenizer -Malayalam data
from keras.preprocessing.text import Tokenizer
x_tokens = Tokenizer()
```

```
x_tokens.fit_on_texts(x_tr)
x_tr = x_tokens.texts_to_sequences(x_tr)
x_val = x_tokens.texts_to_sequences(x_val)
print('x_tr:',x_tr)
print('x_val:',x_val)
```

```
x_tr: [[463, 238, 230, 6, 817, 6, 2, 239], [149, 5, 32, 1173, 209], [118, 1604, 464, 3],
x_val: [[2, 18, 6, 2, 262], [188, 6, 30], [26, 64, 3, 55], [1, 326, 4, 1, 33], [341, 3,
```

```
#padding with post (appending zeros at the end to equalize sentence length)
from keras.preprocessing.sequence import pad_sequences
x_tr = pad_sequences(x_tr,maxlen = maxlen_english,padding = 'post')
x_val = pad_sequences(x_val,maxlen = maxlen_english,padding = 'post')

# +1 for padding
x_voc_size   =  len(x_tokens.word_index) +1
print("No of unique words in English",x_voc_size)
```

```
No of unique words in English 3031
```

```
# English data preprocessing
from keras.preprocessing.text import Tokenizer
y_tokens = Tokenizer()
y_tokens.fit_on_texts(y_tr)

y_tr = y_tokens.texts_to_sequences(y_tr)
y_val = y_tokens.texts_to_sequences(y_val)

from keras.preprocessing.sequence import pad_sequences
y_tr = pad_sequences(y_tr,maxlen = maxlen_malayalam,padding = 'post')
y_val = pad_sequences(y_val,maxlen = maxlen_malayalam,padding = 'post')

# +1 for padding
y_voc_size   =  len(y_tokens.word_index) +1
print("No of unique words in Malyalam",y_voc_size)
```

```
No of unique words in Malyalam 5674
```

```
pip install keras-applications
```

```
Collecting keras-applications
  Downloading Keras_Applications-1.0.8-py3-none-any.whl (50 kB)
     |████████████████████████████████| 50 kB 4.0 MB/s
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages (from kera
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-package
Installing collected packages: keras-applications
Successfully installed keras-applications-1.0.8
```

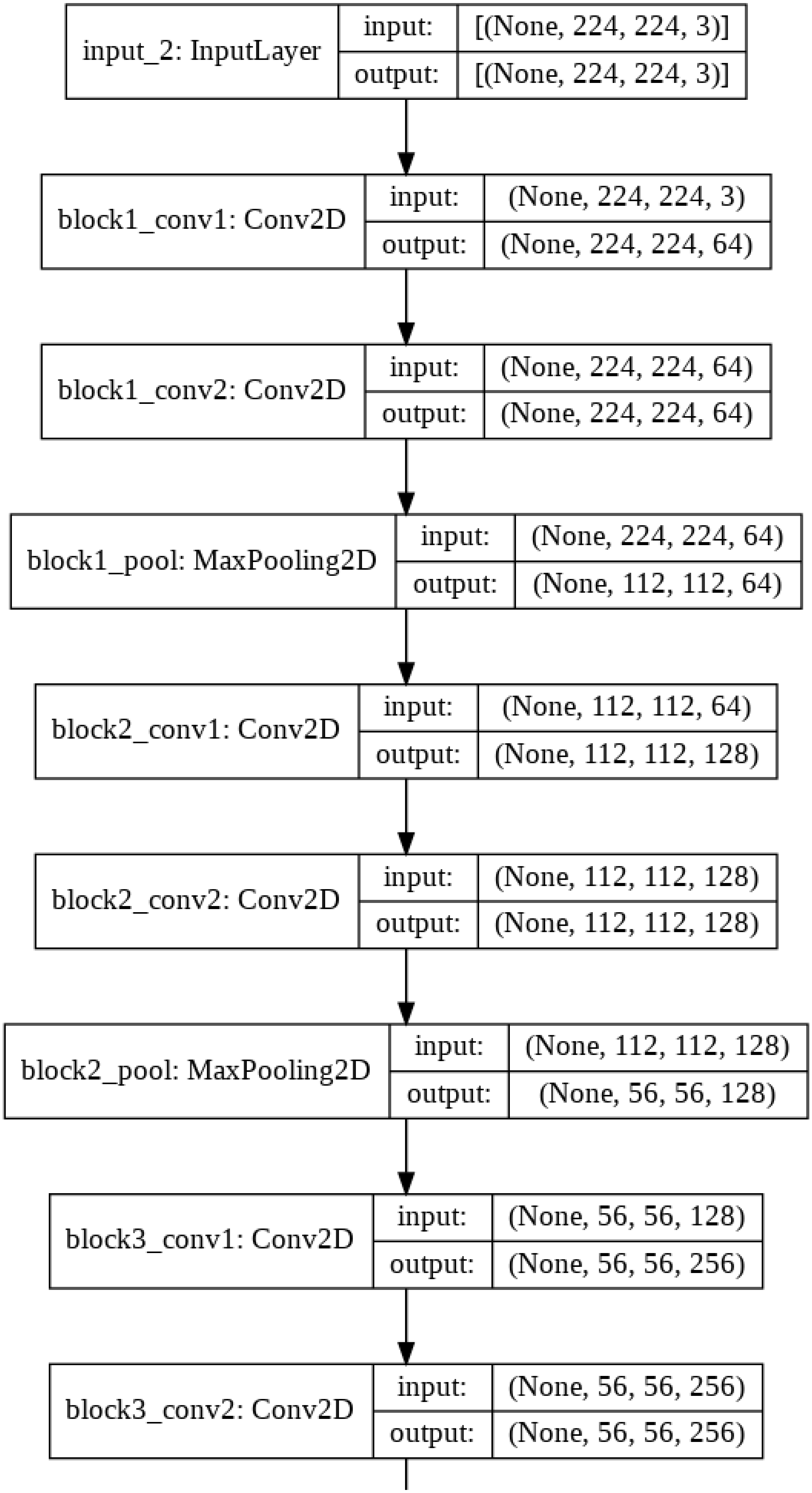
```
import pandas as pd
import pickle
import numpy as np
import os
import keras
import tensorflow
from keras_applications.resnet import ResNet50
from tensorflow.keras.optimizers import Adam
from keras.layers import Dense, GlobalAveragePooling2D,BatchNormalization,Flatten,Input, Conv
from keras.models import Sequential, Model
from keras.utils import np_utils
import random
from keras.preprocessing import image, sequence
import matplotlib.pyplot as plt
import keras
from keras import backend as K
import gensim
from numpy import *
import numpy as np
import pandas as pd
import re
from tensorflow.keras.applications.vgg16 import VGG16
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from nltk.corpus import stopwords
from tensorflow.keras.layers import Input, LSTM, Embedding, Dense, Concatenate, TimeDistribut
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping
import warnings
```

```
#Loading VGG model for Feature Extraction-Removing classification layers from memory
modelvgg = VGG16(include_top=True,weights="imagenet")
modelvgg.layers.pop()
modelvgg = Model(inputs=modelvgg.inputs, outputs=modelvgg.layers[-2].output)
modelvgg.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg1
553467904/553467096 [==============================] - 3s 0us/step
553476096/553467096 [==============================] - 3s 0us/step
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
```

```
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
=================================================================
Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.utils.vis_utils import plot_model
import tensorflow as tf


tf.keras.utils.plot_model(
    modelvgg,
    to_file='model.png',
    show_shapes=True,
    show_layer_names=True,
    rankdir='TB'
)
```



input_2: InputLayer | input: [(None, 224, 224, 3)] | output: [(None, 224, 224, 3)]

block1_conv1: Conv2D | input: (None, 224, 224, 3) | output: (None, 224, 224, 64)

block1_conv2: Conv2D | input: (None, 224, 224, 64) | output: (None, 224, 224, 64)

block1_pool: MaxPooling2D | input: (None, 224, 224, 64) | output: (None, 112, 112, 64)

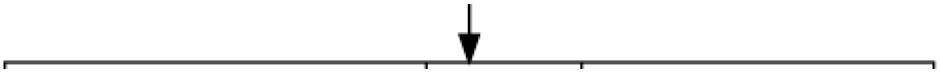
block2_conv1: Conv2D | input: (None, 112, 112, 64) | output: (None, 112, 112, 128)

block2_conv2: Conv2D | input: (None, 112, 112, 128) | output: (None, 112, 112, 128)

block2_pool: MaxPooling2D | input: (None, 112, 112, 128) | output: (None, 56, 56, 128)

block3_conv1: Conv2D | input: (None, 56, 56, 128) | output: (None, 56, 56, 256)

block3_conv2: Conv2D | input: (None, 56, 56, 256) | output: (None, 56, 56, 256)



```
pip install cv
```

```
Collecting cv
  Downloading cv-1.0.0-py3-none-any.whl (7.3 kB)
Installing collected packages: cv
Successfully installed cv-1.0.0
```

```
import cv2
import cv
```

```
ERROR:root:Error disabling cv.imshow().
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/google/colab/_import_hooks/_cv2.py", line
    cv_module.imshow,
AttributeError: module 'cv' has no attribute 'imshow'
```

```
#tRY RESNET
```

```
#Resizing image and converting grey scale images into RGB images
#split=5000
imagedata=np.zeros(shape=(splits,224,224,3))
for i in range(splits):
    temp=mp.imread(img_path[i])
    if (len(temp.shape)==3):
       temp=cv2.resize(temp,(224,224))
       imagedata[i]=temp
    elif (len(temp.shape)<3):
       #plt.imshow(temp)
       temp=cv2.cvtColor(temp, cv2.COLOR_BGR2RGB)
       temp=cv2.resize(temp,(224,224))
       imagedata[i]=temp
imagedata=imagedata/255
imagedata=imagedata.astype(np.float16)
```

```
imagedata[1:10]
```

```
array([[[[0.4626  , 0.443   , 0.3254  ],
         [0.4548  , 0.4353  , 0.3176  ],
         [0.4666  , 0.447   , 0.3293  ],
         ...,
         [0.902   , 0.933   , 0.9453  ],
         [0.906   , 0.937   , 0.949   ],
         [0.906   , 0.937   , 0.949   ]],

        [[0.4666  , 0.447   , 0.3293  ],
         [0.4587  , 0.4392  , 0.3215  ],
         [0.4707  , 0.451   , 0.3333  ],
         ...,
         [0.949   , 0.9805  , 0.9883  ],
         [0.9453  , 0.9766  , 0.9883  ],
         [0.9453  , 0.9766  , 0.9883  ]],

        [[0.4707  , 0.451   , 0.3333  ],
         [0.4626  , 0.443   , 0.3254  ],
         [0.4785  , 0.4587  , 0.341   ],
         ...,
         [0.961   , 0.992   , 1.      ],
         [0.9453  , 0.9766  , 0.992   ],
         [0.9453  , 0.9766  , 0.992   ]],

        ...,

        [[0.3098  , 0.2864  , 0.2393  ],
         [0.341   , 0.3098  , 0.2666  ],
         [0.3647  , 0.3254  , 0.2864  ],
         ...,
         [0.1686  , 0.153   , 0.1059  ],
         [0.1647  , 0.149   , 0.102   ],
         [0.1569  , 0.1412  , 0.0941  ]],

        [[0.3098  , 0.2864  , 0.2393  ],
         [0.3215  , 0.2903  , 0.2471  ],
         [0.3608  , 0.3176  , 0.2783  ],
         ...,
         [0.1608  , 0.1451  , 0.098   ],
         [0.1647  , 0.149   , 0.102   ],
         [0.153   , 0.1372  , 0.0902  ]],

        [[0.3176  , 0.298   , 0.2471  ],
         [0.3215  , 0.2903  , 0.2471  ],
         [0.349   , 0.3098  , 0.2705  ],
         ...,
         [0.153   , 0.1372  , 0.0902  ],
         [0.1686  , 0.153   , 0.1059  ],
         [0.149   , 0.1333  , 0.0863  ]]],


       [[[0.51    , 0.502   , 0.506   ],
         [0.506   , 0.506   , 0.51    ],
         [0.51    , 0.506   , 0.5215  ],
         ...,
         [0.392   , 0.408   , 0.447   ],
         [0.3843  , 0.408   , 0.4548  ],
         [0.3765  , 0.408   , 0.4587  ]],
```

```
with open('/content/drive/My Drive/Main/imagedatas.txt', 'w') as writefile:
    writefile.write("imagedata")
```

```
#preprocessing images
from keras.preprocessing.image import load_img, img_to_array
from keras.applications.vgg16 import preprocess_input
from collections import OrderedDict
```

```
jpgs=img_path[:splits]
images_new = OrderedDict()
npix = 224
target_size = (npix,npix,3)
for i,name in enumerate(jpgs):
    filename = name
    image = load_img(filename, target_size=target_size)
    # convert the image pixels to a numpy array
    image = img_to_array(image)
    nimage = preprocess_input(image)
    y_pred = modelvgg.predict(nimage.reshape( (1,) + nimage.shape[:3]))
    images_new [name] = y_pred.flatten()
    if i%200==0:
        print(i,filename)
```

```
0 /content/drive/My Drive/Main/trainimages/train/10.jpg
200 /content/drive/My Drive/Main/trainimages/train/739.jpg
400 /content/drive/My Drive/Main/trainimages/train/1529.jpg
600 /content/drive/My Drive/Main/trainimages/train/2238.jpg
800 /content/drive/My Drive/Main/trainimages/train/2970.jpg
1000 /content/drive/My Drive/Main/trainimages/train/3693.jpg
1200 /content/drive/My Drive/Main/trainimages/train/4450.jpg
1400 /content/drive/My Drive/Main/trainimages/train/150275.jpg
1600 /content/drive/My Drive/Main/trainimages/train/497937.jpg
1800 /content/drive/My Drive/Main/trainimages/train/713300.jpg
2000 /content/drive/My Drive/Main/trainimages/train/1159284.jpg
2200 /content/drive/My Drive/Main/trainimages/train/1160083.jpg
2400 /content/drive/My Drive/Main/trainimages/train/1592294.jpg
2600 /content/drive/My Drive/Main/trainimages/train/1592957.jpg
2800 /content/drive/My Drive/Main/trainimages/train/2315779.jpg
3000 /content/drive/My Drive/Main/trainimages/train/2316528.jpg
3200 /content/drive/My Drive/Main/trainimages/train/2317337.jpg
3400 /content/drive/My Drive/Main/trainimages/train/2318048.jpg
3600 /content/drive/My Drive/Main/trainimages/train/2318796.jpg
3800 /content/drive/My Drive/Main/trainimages/train/2319593.jpg
4000 /content/drive/My Drive/Main/trainimages/train/2320436.jpg
4200 /content/drive/My Drive/Main/trainimages/train/2321247.jpg
4400 /content/drive/My Drive/Main/trainimages/train/2322001.jpg
4600 /content/drive/My Drive/Main/trainimages/train/2322763.jpg
4800 /content/drive/My Drive/Main/trainimages/train/2323538.jpg
5000 /content/drive/My Drive/Main/trainimages/train/2324400.jpg
5200 /content/drive/My Drive/Main/trainimages/train/2325137.jpg
5400 /content/drive/My Drive/Main/trainimages/train/2325912.jpg
5600 /content/drive/My Drive/Main/trainimages/train/2326777.jpg
5800 /content/drive/My Drive/Main/trainimages/train/2327564.jpg
6000 /content/drive/My Drive/Main/trainimages/train/2328291.jpg
6200 /content/drive/My Drive/Main/trainimages/train/2329112.jpg
6400 /content/drive/My Drive/Main/trainimages/train/2329883.jpg
6600 /content/drive/My Drive/Main/trainimages/train/2330601.jpg
6800 /content/drive/My Drive/Main/trainimages/train/2331402.jpg
7000 /content/drive/My Drive/Main/trainimages/train/2332176.jpg
7200 /content/drive/My Drive/Main/trainimages/train/2333032.jpg
7400 /content/drive/My Drive/Main/trainimages/train/2333847.jpg
7600 /content/drive/My Drive/Main/trainimages/train/2334602.jpg
7800 /content/drive/My Drive/Main/trainimages/train/2335353.jpg
```

```
8000 /content/drive/My Drive/Main/trainimages/train/2336144.jpg
8200 /content/drive/My Drive/Main/trainimages/train/2336923.jpg
8400 /content/drive/My Drive/Main/trainimages/train/2337672.jpg
8600 /content/drive/My Drive/Main/trainimages/train/2338520.jpg
8800 /content/drive/My Drive/Main/trainimages/train/2339266.jpg
9000 /content/drive/My Drive/Main/trainimages/train/2339991.jpg
9200 /content/drive/My Drive/Main/trainimages/train/2340756.jpg
9400 /content/drive/My Drive/Main/trainimages/train/2341498.jpg
9600 /content/drive/My Drive/Main/trainimages/train/2342276.jpg
9800 /content/drive/My Drive/Main/trainimages/train/2343119.jpg
```

```
print(list(images_new.values())[1])
```

```
[0.6324536 1.3856603 0.        ... 0.        0.        2.182263 ]
```

```
#storing image pixels sepearetely
vgg_feature=np.zeros(shape=(len(jpgs),4096))
for i in range(len(jpgs)):
    vgg_feature[i]=images_new[jpgs[i]]
vgg_feature[1:10]
```

```
array([[0.63245362, 1.38566029, 0.        , ..., 0.        , 0.        ,
         2.1822629 ],
       [0.54209262, 0.        , 0.        , ..., 0.        , 0.        ,
         0.93312246],
       [1.5215044 , 0.        , 0.        , ..., 0.        , 0.        ,
         0.        ],
       ...,
       [0.11626244, 1.3521657 , 0.05647588, ..., 0.        , 2.70661497,
         0.        ],
       [2.4945612 , 1.68215179, 0.        , ..., 0.        , 0.        ,
         0.        ],
       [0.26692578, 4.36360025, 0.        , ..., 0.        , 0.        ,
         0.        ]])
```

```
#splitting image pixels for training and validation
vgg_train_=vgg_feature[:splits-500]
vgg_val=vgg_feature[splits-500:]
```

```
#Generating a repeat vector from image pixels
img_inputs=Input(shape=(4096,))
d_1=Dense(512, activation='relu')(img_inputs)
r_1=RepeatVector(maxlen_english)(d_1)
vf_model = Model(img_inputs, r_1)
vf_model.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 4096)]            0
_________________________________________________________________
```

```
dense (Dense)                (None, 512)               2097664
_________________________________________________________________
repeat_vector (RepeatVector) (None, 14, 512)           0
=================================================================
Total params: 2,097,664
Trainable params: 2,097,664
Non-trainable params: 0
_________________________________________________________________
```

```
x_voc=x_voc_size
y_voc=y_voc_size
```

```
#Model
x_voc=x_voc_size
y_voc=y_voc_size
latent_dim = 512
embedding_dim=512
#Encoder
encoder_inputs = Input(shape=(maxlen_english,))
#The model will take as input an integer matrix of size (batch,input_length)and the largest i
enc_emb =  Embedding(x_voc, embedding_dim,trainable=True)(encoder_inputs)
print(encoder_inputs.get_shape)
print(enc_emb.get_shape)
```

```
<bound method KerasTensor.get_shape of <KerasTensor: shape=(None, 14) dtype=float32 (cre
<bound method KerasTensor.get_shape of <KerasTensor: shape=(None, 14, 512) dtype=float3
```

```
#encoder LSTM Layer 1
encoder_lstm1 = LSTM(latent_dim,return_sequences=True,return_state=True,dropout=0.4,recurrent
#The dimension of each state equals to the LSTM unit number
encoder_output1, state_h1, state_c1 = encoder_lstm1(enc_emb)
print(encoder_lstm1.output_shape)
```

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the crite
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the crite
[(None, 14, 512), (None, 512), (None, 512)]
```

```
#LSTM layer 2
encoder_lstm2 = LSTM(latent_dim,return_sequences=True,return_state=True,dropout=0.4,recurrent
encoder_output2, state_h2, state_c2 = encoder_lstm2(encoder_output1)
print(encoder_lstm2.output_shape)
```

```
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet the cri
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet the cri
[(None, 14, 512), (None, 512), (None, 512)]
```

```
#Concatenating image features with text input
encoder_output2=Concatenate(axis=-1)([encoder_output2,r_1])


#LSTM layer 3
encoder_lstm3=LSTM(latent_dim, return_state=True, return_sequences=True,dropout=0.4,recurrent
encoder_outputs, state_h, state_c= encoder_lstm3(encoder_output2)
print(encoder_lstm3.output_shape)
```

```
WARNING:tensorflow:Layer lstm_2 will not use cuDNN kernels since it doesn't meet the cri
WARNING:tensorflow:Layer lstm_2 will not use cuDNN kernels since it doesn't meet the cri
[(None, 14, 512), (None, 512), (None, 512)]
```

```
#Decoder
# Set up the decoder, using `encoder_states` as initial state.
decoder_inputs = Input(shape=(None,))
#embedding layer
dec_emb_layer = Embedding(y_voc, embedding_dim,trainable=True)
dec_emb = dec_emb_layer(decoder_inputs)
print(decoder_inputs.get_shape)
print(dec_emb.get_shape)
```

```
<bound method KerasTensor.get_shape of <KerasTensor: shape=(None, None) dtype=float32 (
<bound method KerasTensor.get_shape of <KerasTensor: shape=(None, None, 512) dtype=floa
```

```
#Decoder LSTM layer1
decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True,dropout=0.4,recurren
decoder_outputs,decoder_fwd_state, decoder_back_state = decoder_lstm(dec_emb,initial_state=[s
print(decoder_lstm.output_shape)
```

```
WARNING:tensorflow:Layer lstm_3 will not use cuDNN kernels since it doesn't meet the cri
WARNING:tensorflow:Layer lstm_3 will not use cuDNN kernels since it doesn't meet the cri
[(None, None, 512), (None, 512), (None, 512)]
```

```
#dense layer
decoder_dense =  TimeDistributed(Dense(y_voc, activation='softmax'))
decoder_outputs = decoder_dense(decoder_outputs)
print(decoder_dense.output_shape)
```

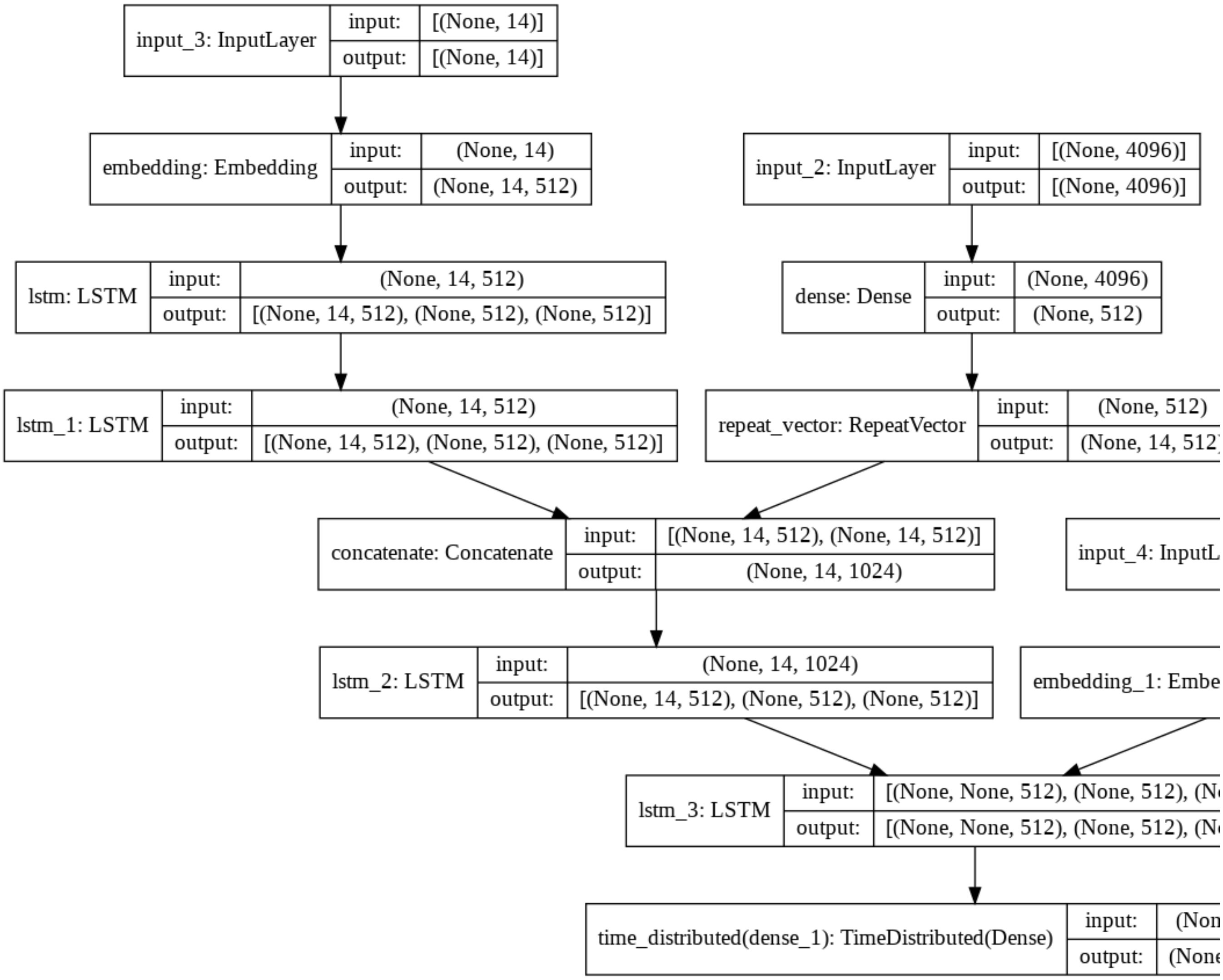
```
(None, None, 5674)
```

```
model = Model([encoder_inputs,decoder_inputs,img_inputs], decoder_outputs)
model.summary()
```

```
Model: "model_2"
_________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
```

```
==================================================================================================
input_3 (InputLayer)            [(None, 14)]         0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 14, 512)      1551872     input_3[0][0]
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 4096)]       0
__________________________________________________________________________________________________
lstm (LSTM)                     [(None, 14, 512), (N 2099200     embedding[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 512)          2097664     input_2[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   [(None, 14, 512), (N 2099200     lstm[0][0]
__________________________________________________________________________________________________
repeat_vector (RepeatVector)    (None, 14, 512)      0           dense[0][0]
__________________________________________________________________________________________________
input_4 (InputLayer)            [(None, None)]       0
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 14, 1024)     0           lstm_1[0][0]
                                                                 repeat_vector[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, None, 512)    2905088     input_4[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   [(None, 14, 512), (N 3147776     concatenate[0][0]
__________________________________________________________________________________________________
lstm_3 (LSTM)                   [(None, None, 512),  2099200     embedding_1[0][0]
                                                                 lstm_2[0][1]
                                                                 lstm_2[0][2]
__________________________________________________________________________________________________
time_distributed (TimeDistribut (None, None, 5674)   2910762     lstm_3[0][0]
==================================================================================================
Total params: 18,910,762
Trainable params: 18,910,762
Non-trainable params: 0
__________________________________________________________________________________________________
```

```
from keras.utils.vis_utils import plot_model
import tensorflow as tf


tf.keras.utils.plot_model(
    model,
    to_file='model.png',
    show_shapes=True,
    show_layer_names=True,
    rankdir='TB'
)
```



```
in
```

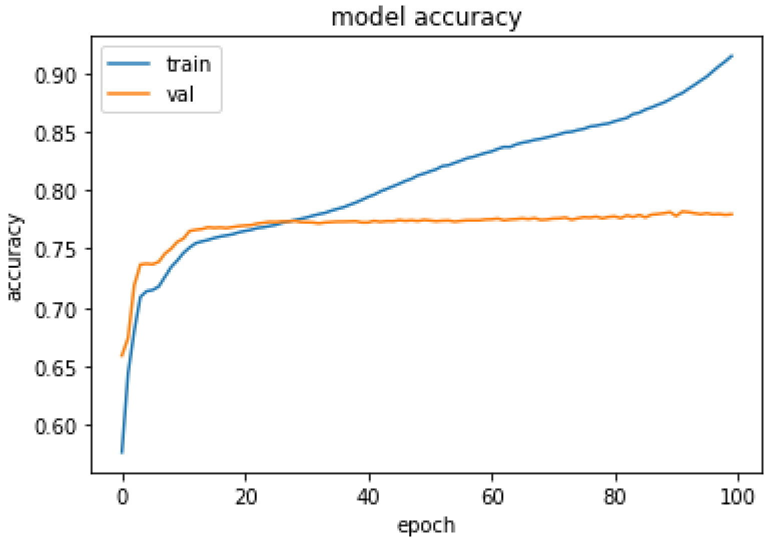
```
#compiling model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',metrics=['accuracy'])
history=model.fit([x_tr,y_tr[:,:-1],vgg_train_], y_tr.reshape(y_tr.shape[0],y_tr.shape[1], 1)
```

```
    Epoch 1/100
    19/19 [==============================] - 15s 324ms/step - loss: 4.0751 - accuracy: 0.
    Epoch 2/100
    19/19 [==============================] - 5s 271ms/step - loss: 2.4583 - accuracy: 0.6
    Epoch 3/100
    19/19 [==============================] - 5s 263ms/step - loss: 2.2810 - accuracy: 0.6
```

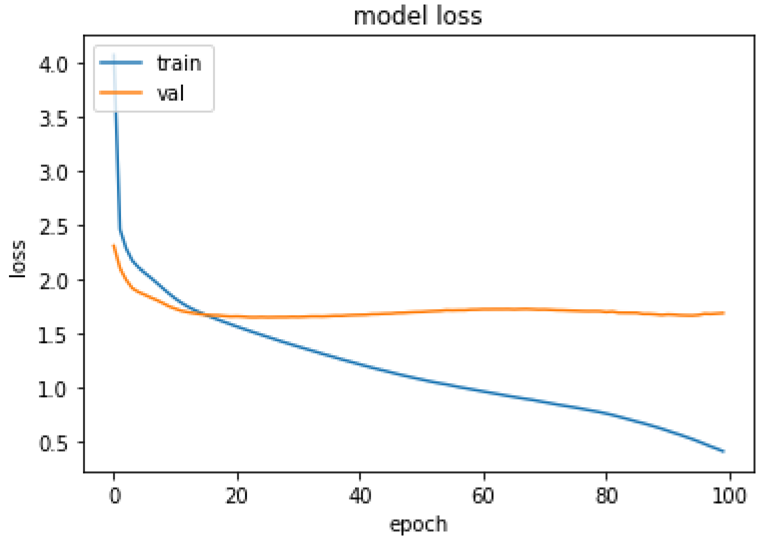
```
Epoch 4/100
19/19 [==============================] - 5s 274ms/step - loss: 2.1676 - accuracy: 0.7
Epoch 5/100
19/19 [==============================] - 5s 270ms/step - loss: 2.1056 - accuracy: 0.7
Epoch 6/100
19/19 [==============================] - 5s 268ms/step - loss: 2.0594 - accuracy: 0.7
Epoch 7/100
19/19 [==============================] - 5s 263ms/step - loss: 2.0141 - accuracy: 0.7
Epoch 8/100
19/19 [==============================] - 5s 273ms/step - loss: 1.9649 - accuracy: 0.7
Epoch 9/100
19/19 [==============================] - 5s 276ms/step - loss: 1.9147 - accuracy: 0.7
Epoch 10/100
19/19 [==============================] - 5s 276ms/step - loss: 1.8653 - accuracy: 0.7
Epoch 11/100
19/19 [==============================] - 5s 272ms/step - loss: 1.8208 - accuracy: 0.7
Epoch 12/100
19/19 [==============================] - 5s 273ms/step - loss: 1.7808 - accuracy: 0.7
Epoch 13/100
19/19 [==============================] - 5s 271ms/step - loss: 1.7465 - accuracy: 0.7
Epoch 14/100
19/19 [==============================] - 5s 274ms/step - loss: 1.7173 - accuracy: 0.7
Epoch 15/100
19/19 [==============================] - 5s 271ms/step - loss: 1.6913 - accuracy: 0.7
Epoch 16/100
19/19 [==============================] - 5s 274ms/step - loss: 1.6674 - accuracy: 0.7
Epoch 17/100
19/19 [==============================] - 5s 279ms/step - loss: 1.6450 - accuracy: 0.7
Epoch 18/100
19/19 [==============================] - 5s 279ms/step - loss: 1.6233 - accuracy: 0.7
Epoch 19/100
19/19 [==============================] - 5s 279ms/step - loss: 1.6023 - accuracy: 0.7
Epoch 20/100
19/19 [==============================] - 5s 275ms/step - loss: 1.5819 - accuracy: 0.7
Epoch 21/100
19/19 [==============================] - 5s 270ms/step - loss: 1.5611 - accuracy: 0.7
Epoch 22/100
19/19 [==============================] - 5s 281ms/step - loss: 1.5411 - accuracy: 0.7
Epoch 23/100
19/19 [==============================] - 5s 267ms/step - loss: 1.5216 - accuracy: 0.7
Epoch 24/100
19/19 [==============================] - 5s 282ms/step - loss: 1.5021 - accuracy: 0.7
Epoch 25/100
19/19 [==============================] - 5s 270ms/step - loss: 1.4835 - accuracy: 0.7
Epoch 26/100
19/19 [==============================] - 5s 270ms/step - loss: 1.4647 - accuracy: 0.7
Epoch 27/100
19/19 [==============================] - 5s 277ms/step - loss: 1.4463 - accuracy: 0.7
Epoch 28/100
19/19 [==============================] - 5s 270ms/step - loss: 1.4275 - accuracy: 0.7
Epoch 29/100
19/19 [==============================] - 5s 275ms/step - loss: 1.4097 - accuracy: 0.7
```

```
import keras
from matplotlib import pyplot as plt
```

```
#history = model1.fit(train_x, train_y,validation_split = 0.1, epochs=50, batch_size=4)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



```
reverse_target_word_index=y_tokens.index_word
```

```
reverse_target_word_index
```

```
{1: 'sos',
 2: 'eos',
 3: 'ഒരു',
 4: 'കറുത്ത',
 5: 'വെളുത്ത',
 6: 'സ്ത്രീ',
 7: 'ധരിച്ച',
 8: 'മനുഷ്യൻ',
 9: 'പച്ച',
 10: 'വലിയ',
 11: 'നീല',
 12: 'ബാഗ്',
 13: 'വർണ്ണാഭമായ',
 14: 'ചുവന്ന',
 15: 'ചാരനിറത്തിലുള്ള',
 16: 'സ്കാർഫ്',
 17: 'ട്രാഷ്',
 18: 'റോഡിന്റെ',
 19: 'വിൻഡോ',
 20: 'പുല്ലിന്റെ',
 21: 'മഞ്ഞ',
 22: 'വശങ്ങളിൽ',
 23: 'സ്ട്രിപ്പുകൾ',
 24: 'വ്യക്തി',
 25: 'ഇതൊരു',
 26: 'രണ്ട്',
 27: 'ചെറിയ',
 28: 'കെട്ടിടത്തിലെ',
 29: 'നിലത്ത്',
 30: 'ആന',
 31: 'വെള്ള',
 32: 'ഒരാൾ',
 33: 'ഉള്ള',
 34: 'ചുവപ്പ്',
 35: 'തവിട്ട്',
 36: 'കുഞ്ഞ്',
 37: 'ജിറാഫ്',
 38: 'വെള്ളയും',
 39: 'ഓറഞ്ച്',
 40: 'മുകളിൽ',
 41: 'പുല്ല്',
 42: 'എഞ്ചിൻ',
 43: 'ചിഹ്നം',
 44: 'കാർ',
 45: 'കെട്ടിടത്തിന്റെ',
 46: 'മേശപ്പുറത്ത്',
 47: 'പുള്ളി',
 48: 'ആളുകൾ',
 49: 'നിറമുള്ള',
 50: 'ടെന്നീസ്',
 51: 'രംഗം',
 52: 'ട്രെയിൻ',
 53: 'തലം',
 54: 'ഇത്',
 55: 'ബീച്ച്',
 56: 'നിൽക്കുന്നു',
 57: 'തിന്നുന്നു',
 58: 'നിൽക്കുന്ന',
 59: 'ദ്വീപ്',
```

```
reverse_source_word_index=x_tokens.index_word
reverse_source_word_index
```

```
    {1: 'a',
     2: 'the',
     3: 'on',
     4: 'of',
     5: 'is',
     6: 'in',
     7: 'white',
     8: 'man',
     9: 'and',
     10: 'black',
     11: 'with',
     12: 'red',
     13: 'this',
     14: 'person',
     15: 'blue',
     16: 'building',
     17: 'wall',
     18: 'woman',
     19: 'brown',
     20: 'wearing',
     21: 'green',
     22: 'window',
     23: 'yellow',
     24: 'head',
     25: 'sign',
     26: 'two',
     27: 'train',
     28: 'street',
     29: 'water',
     30: 'sky',
     31: 'side',
     32: 'an',
     33: 'table',
     34: 'car',
     35: 'standing',
     36: 'light',
     37: 'large',
     38: 'clock',
     39: 'people',
     40: 'shirt',
     41: 'sitting',
     42: 'holding',
     43: 'are',
     44: 'small',
     45: 'plate',
     46: 'has',
     47: 'bus',
     48: 'road',
     49: 'to',
     50: 'dog',
     51: 'grass',
     52: 'orange',
     53: 'tennis',
     54: 'top',
     55: 'ground',
     56: 'silver',
     57: 'cat',
     58: 'plane',
     59: 'at',
```

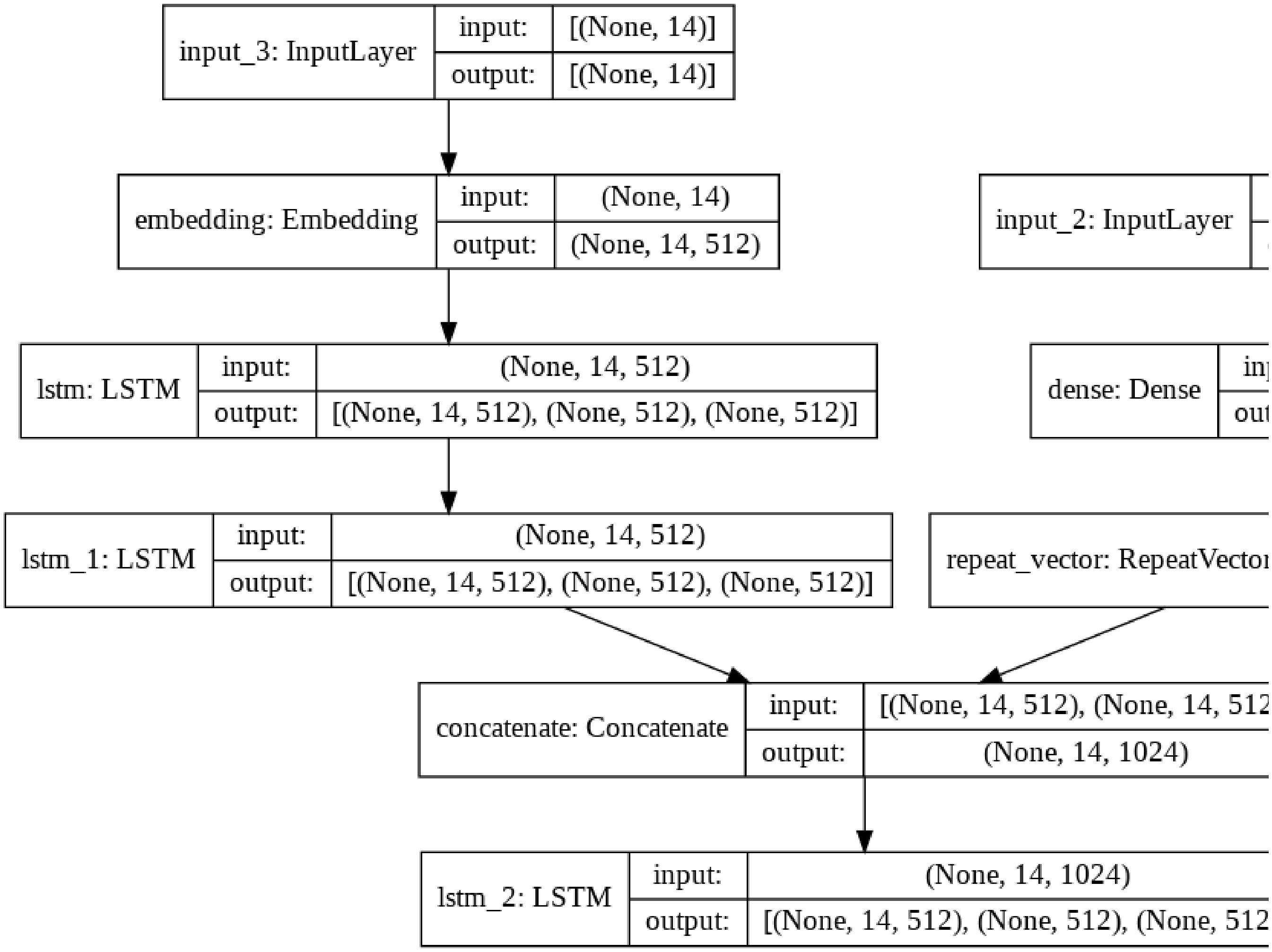
```
target_word_index=y_tokens.word_index
target_word_index
```

```
    {'sos': 1,
     'eos': 2,
     'ഒരു': 3,
     'കറുത്ത': 4,
     'വെളുത്ത': 5,
     'സ്ത്രീ': 6,
     'ധരിച്ച': 7,
     'മനുഷ്യൻ': 8,
     'പച്ച': 9,
     'വലിയ': 10,
     'നീല': 11,
     'ബാഗ്': 12,
     'വർണ്ണാഭമായ': 13,
     'ചുവന്ന': 14,
     'ചാരനിറത്തിലുള്ള': 15,
     'സ്കാർഫ്': 16,
     'ട്രാഷ്': 17,
     'റോഡിന്റെ': 18,
     'വിൻഡോ': 19,
     'പുല്ലിന്റെ': 20,
     'മഞ്ഞ': 21,
     'വശങ്ങളിൽ': 22,
     'സ്ട്രിപ്പുകൾ': 23,
     'വ്യക്തി': 24,
     'ഇതൊരു': 25,
     'രണ്ട്': 26,
     'ചെറിയ': 27,
     'കെട്ടിടത്തിലെ': 28,
     'നിലത്ത്': 29,
     'ആന': 30,
     'വെള്ള': 31,
     'ഒരാൾ': 32,
     'ഉള്ള': 33,
     'ചുവപ്പ്': 34,
     'തവിട്ട്': 35,
     'കുഞ്ഞ്': 36,
     'ജിറാഫ്': 37,
     'വെള്ളയും': 38,
     'ഓറഞ്ച്': 39,
     'മുകളിൽ': 40,
     'പുല്ല്': 41,
     'എഞ്ചിൻ': 42,
     'ചിഹ്നം': 43,
     'കാർ': 44,
     'കെട്ടിടത്തിന്റെ': 45,
     'മേശപ്പുറത്ത്': 46,
     'പുള്ളി': 47,
     'ആളുകൾ': 48,
     'നിറമുള്ള': 49,
     'ടെന്നീസ്': 50,
     'രംഗം': 51,
     'ട്രെയിൻ': 52,
     'തലം': 53,
     'ഇത്': 54,
     'ബീച്ച്': 55,
     'നിൽക്കുന്നു': 56,
     'തിന്നുന്നു': 57,
     'നിൽക്കുന്ന': 58,
     'ദ്വീപ്': 59,
```

```
# Encode the input sequence to get the feature vector
encoder_model = Model(inputs=[encoder_inputs,img_inputs],outputs=[encoder_outputs, state_h, s
encoder_model.summary()
```

```
Model: "model_3"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            [(None, 14)]         0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 14, 512)      1551872     input_3[0][0]
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 4096)]       0
__________________________________________________________________________________________________
lstm (LSTM)                     [(None, 14, 512), (N 2099200     embedding[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 512)          2097664     input_2[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   [(None, 14, 512), (N 2099200     lstm[0][0]
__________________________________________________________________________________________________
repeat_vector (RepeatVector)    (None, 14, 512)      0           dense[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 14, 1024)     0           lstm_1[0][0]
                                                                 repeat_vector[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   [(None, 14, 512), (N 3147776     concatenate[0][0]
==================================================================================================
Total params: 10,995,712
Trainable params: 10,995,712
Non-trainable params: 0
__________________________________________________________________________________________________
```

```
tf.keras.utils.plot_model(
        encoder_model,
        to_file='model.png',
        show_shapes=True,
        show_layer_names=True,
        rankdir='TB'
    )
```

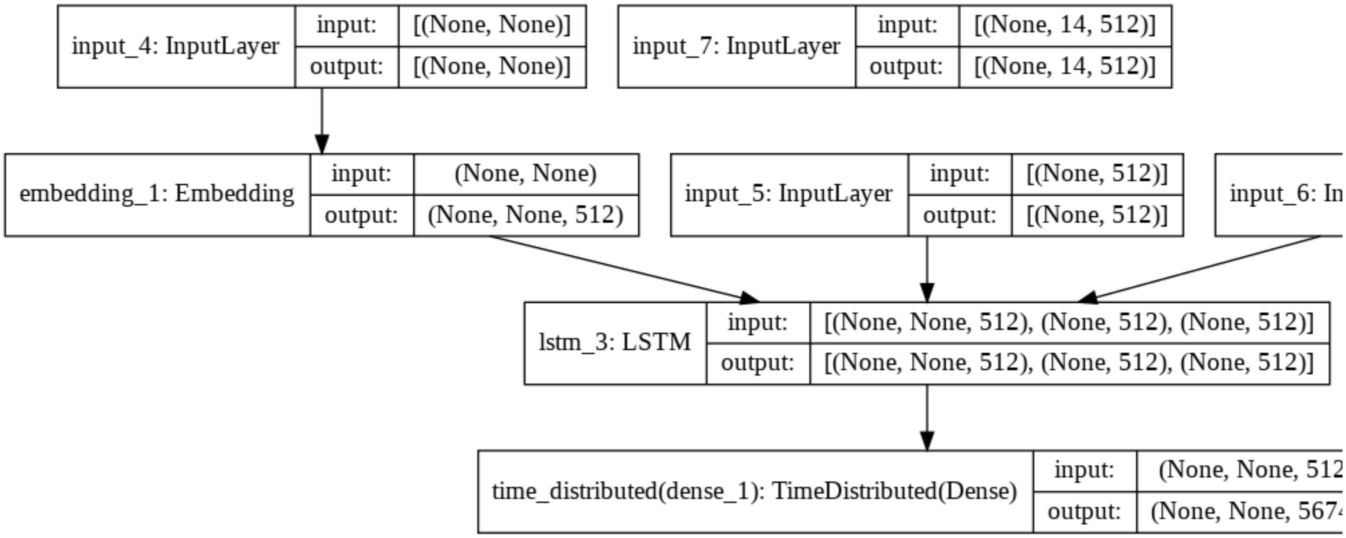


```
# Decoder setup
# Below tensors will hold the states of the previous time step
decoder_state_input_h = Input(shape=(latent_dim,))
decoder_state_input_c = Input(shape=(latent_dim,))
decoder_hidden_state_input = Input(shape=(maxlen_english,latent_dim))
print(decoder_inputs.get_shape)
#print(dec_emb.get_shape)
```

```
<bound method KerasTensor.get_shape of <KerasTensor: shape=(None, None) dtype=float32 (
# Get the embeddings of the decoder sequence
dec_emb2= dec_emb_layer(decoder_inputs)
# To predict the next word in the sequence, set the initial states to the states from the pre
decoder_outputs2, state_h2, state_c2 = decoder_lstm(dec_emb2, initial_state=[decoder_state_in

# A dense softmax layer to generate prob dist. over the target vocabulary
decoder_outputs2 = decoder_dense(decoder_outputs2)

# Final decoder model
decoder_model = Model(
    [decoder_inputs] + [decoder_hidden_state_input,decoder_state_input_h, decoder_state_input
    [decoder_outputs2] + [state_h2, state_c2])
decoder_model.summary()
```

```
Model: "model_4"
_________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
=========================================================================================
input_4 (InputLayer)            [(None, None)]       0
_________________________________________________________________________________________
embedding_1 (Embedding)         (None, None, 512)    2905088     input_4[0][0]
_________________________________________________________________________________________
input_5 (InputLayer)            [(None, 512)]        0
_________________________________________________________________________________________
input_6 (InputLayer)            [(None, 512)]        0
_________________________________________________________________________________________
lstm_3 (LSTM)                   [(None, None, 512),  2099200     embedding_1[1][0]
                                                                 input_5[0][0]
                                                                 input_6[0][0]
_________________________________________________________________________________________
input_7 (InputLayer)            [(None, 14, 512)]    0
_________________________________________________________________________________________
time_distributed (TimeDistribut (None, None, 5674)   2910762     lstm_3[1][0]
=========================================================================================
Total params: 7,915,050
Trainable params: 7,915,050
Non-trainable params: 0
_________________________________________________________________________________________
```

```
tf.keras.utils.plot_model(
    decoder_model,
    to_file='model.png',
    show_shapes=True,
    show_layer_names=True,
    rankdir='TB'
)
```



```
def decode_sequence(input_seq,img):
    img=img[np.newaxis,:]
    # Encode the input as state vectors.
    e_out, e_h, e_c = encoder_model.predict([input_seq,img])

    # Generate empty target sequence of length 1.
    target_seq = np.zeros((1,1))

    # Populate the first word of target sequence with the start word.
    target_seq[0, 0] = target_word_index['sos']

    stop_condition = False
    decoded_sentence = ''
    while not stop_condition:

        output_tokens, h, c = decoder_model.predict([target_seq] + [e_out, e_h, e_c])

        # Sample a token
        sampled_token_index = np.argmax(output_tokens[0, -1, :])
        sampled_token = reverse_target_word_index[sampled_token_index]

        if(sampled_token!='eos'):
            decoded_sentence += ' '+sampled_token

        # Exit condition: either hit max length or find stop word.
        if (sampled_token == 'eos'  or len(decoded_sentence.split()) >= (maxlen_malayalam -1)
            stop_condition = True

        # Update the target sequence (of length 1).
        target_seq = np.zeros((1,1))
        target_seq[0, 0] = sampled_token_index
```

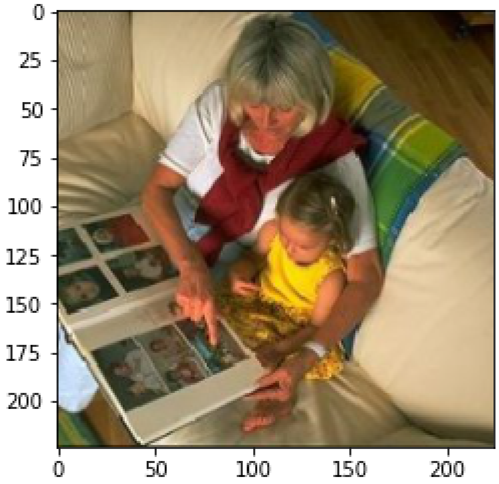
```
            # Update internal states
            e_h, e_c = h, c

        return decoded_sentence


    def seq2summary(input_seq):
        newString=''
        for i in input_seq:
            if((i!=0 and i!=target_word_index['sos']) and i!=target_word_index['eos']):
                newString=newString+reverse_target_word_index[i]+' '
        return newString

    def seq2text(input_seq):
        newString=''
        for i in input_seq:
            if(i!=0):
                newString=newString+reverse_source_word_index[i]+' '
        return newString

    for i in range(5):
        print("Review:",seq2text(x_tr[i]))
        print("Original summary:",seq2summary(y_tr[i]))
        print("Predicted summary:",decode_sequence(x_tr[i].reshape(1,maxlen_english),vgg_train_[i
        print("\n")
```

```
Review: male surfer surfing in still in the ocean
Original summary: ശാന്തമായ കടലിൽ സർഫിങ് നടത്തുന്ന പുരുഷ സർഫർ
Predicted summary:  തിരമാലയിൽ കയറുന്ന ഒരാൾ


Review: it is an indoor scene
Original summary: ഇത് ഒരു ഇൻഡോർ രംഗമാണ്
Predicted summary:  മനുഷ്യന്റെ തലയിൽ കറുത്ത മാസ്ക്


Review: computer screens turned on
Original summary: കമ്പ്യൂട്ടർ സ്ക്രീനുകൾ ഓണാക്കി
Predicted summary:  കമ്പ്യൂട്ടർ സ്ക്രീനുകൾ ഓണാക്കി


Review: man has short hair
Original summary: മനുഷ്യന് ചെറിയ മുടിയുണ്ട്
Predicted summary:  മനുഷ്യൻ സ്റ്റ യിൽ നിൽക്കുന്നു


Review: photo album open on an adults lap
Original summary: ഫോട്ടോ ആൽബം മുതിർന്നവരുടെ മടിയിൽ തുറക്കുന്നു
Predicted summary:  ഫോട്ടോ ആൽബം മുതിർന്നവരുടെ മടിയിൽ തുറക്കുന്നു
```

```
i=4
```

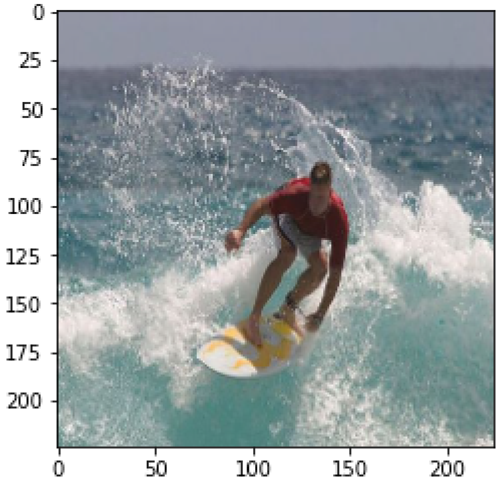
```
print("Review:",seq2text(x_tr[i]))
#print("Original summary:",seq2summary(y_tr[i]))
print("Predicted summary:",decode_sequence(x_tr[i].reshape(1,maxlen_english),vgg_train_[i]))
plt.imshow(imagedata[i].astype(np.float32))
```

```
    Review: photo album open on an adults lap
    Predicted summary:  ഫോട്ടോ ആൽബം മുതിർന്നവരുടെ മടിയിൽ തുറക്കുന്നു
    <matplotlib.image.AxesImage at 0x7ff53f2a7150>
```



```
i=0
print("Review:",seq2text(x_tr[i]))
print("Original summary:",seq2summary(y_tr[i]))
print("Predicted summary:",decode_sequence(x_tr[i].reshape(1,maxlen_english),vgg_train_[i]))
plt.imshow(imagedata[i].astype(np.float32))
```

```
    Review: male surfer surfing in still in the ocean
    Original summary: ശാന്തമായ കടലിൽ സർഫിങ് നടത്തുന്ന പുരുഷ സർഫർ
    Predicted summary:  തിരമാലയിൽ കയറുന്ന ഒരാൾ
    <matplotlib.image.AxesImage at 0x7ff53edf4e10>
```



```
!pip install sacrebleu
import sacrebleu
import random
```

```
Collecting sacrebleu
  Downloading sacrebleu-2.0.0-py3-none-any.whl (90 kB)
     |████████████████████████████████| 90 kB 5.7 MB/s
Collecting colorama
  Downloading colorama-0.4.4-py2.py3-none-any.whl (16 kB)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: regex in /usr/local/lib/python3.7/dist-packages (from sac
Collecting portalocker
  Downloading portalocker-2.3.2-py2.py3-none-any.whl (15 kB)
Requirement already satisfied: tabulate>=0.8.9 in /usr/local/lib/python3.7/dist-packages
Installing collected packages: portalocker, colorama, sacrebleu
Successfully installed colorama-0.4.4 portalocker-2.3.2 sacrebleu-2.0.0
```

```
temp_o=[]
temp_p=[]
for i in range(50):
    s=random.randint(0,len(y_tr)-1)
    temp_o.append(seq2summary(y_tr[s]))
    temp_p.append(decode_sequence(x_tr[s].reshape(1,maxlen_english),vgg_train_[s]))

bleu = sacrebleu.corpus_bleu(temp_o, [temp_p],lowercase=True, tokenize='intl')
print(bleu.score)
```

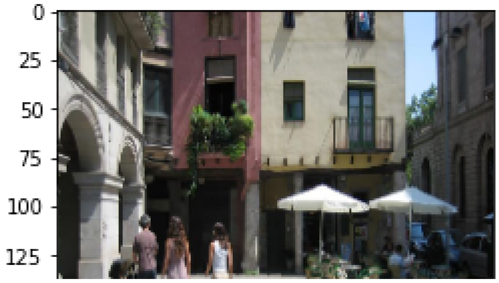
```
25.403027683651814
```

```
temp_o=[]
temp_p=[]
for i in range(10000):
    s=random.randint(0,len(y_tr)-1)
    temp_o.append(seq2summary(y_tr[s]))
    temp_p.append(decode_sequence(x_tr[s].reshape(1,maxlen_english),vgg_train_[s]))

bleu = sacrebleu.corpus_bleu(temp_o, [temp_p],lowercase=True, tokenize='intl')
print(bleu.score)
```

```
32.396260550602484
```

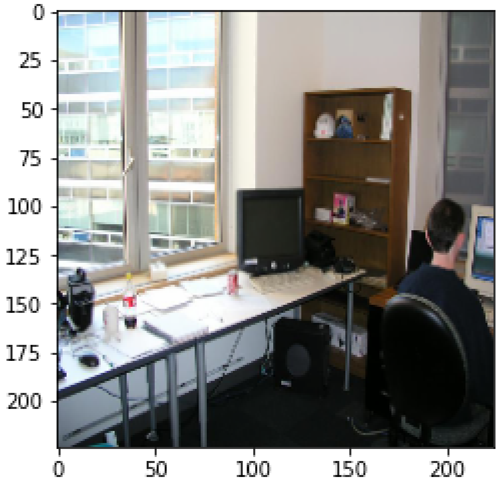
```
i=6
print("Review:",seq2text(x_tr[i]))
print("Original summary:",seq2summary(y_tr[i]))
print("Predicted summary:",decode_sequence(x_tr[i].reshape(1,maxlen_english),vgg_train_[i]))
plt.imshow(imagedata[i].astype(np.float32))
```

```
Review: child in a stroller
Original summary: ഒരു ഉന്തുവണ്ടിയിലെ കുട്ടി
Predicted summary:  ഒരു കറുത്ത ട്രാഷ് ബാഗ്
<matplotlib.image.AxesImage at 0x7ff53f6d18d0>
```



```
i=3
print("Review:",seq2text(x_tr[i]))
print("Original summary:",seq2summary(y_tr[i]))
print("Predicted summary:",decode_sequence(x_tr[i].reshape(1,maxlen_english),vgg_train_[i]))
plt.imshow(imagedata[i].astype(np.float32))
```

```
Review: man has short hair
Original summary: മനുഷ്യന് ചെറിയ മുടിയുണ്ട്
Predicted summary:  മനുഷ്യൻ സ്റ്റ യിൽ നിൽക്കുന്നു
<matplotlib.image.AxesImage at 0x7ff53f43a850>
```



● ✕