Medical Report Generation Case Study : Optimized Solution

```
!pip show tensorflow
!pip install plot_model
!pip install tensorboardcolab
%load_ext tensorboard
!rm -rf ./logs/
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
import re
import os
from nltk.tokenize import word_tokenize
import xml.etree.ElementTree as ET
from os import listdir
from os import path
import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Embedding
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import add
from tensorflow.keras.callbacks import ModelCheckpoint
import pickle
from tqdm import tqdm
import random
from numpy import argmax
from tensorflow.keras.models import load_model
from nltk.translate.bleu_score import corpus_bleu
from tensorboardcolab import *
from tensorflow.keras.callbacks import TensorBoard
from datetime import datetime, timedelta
from tensorflow.keras.utils import plot_model
from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot
from numpy import array
from prettytable import PrettyTable
```

```
Name: tensorflow
Version: 2.2.0rc4
Summary: TensorFlow is an open source machine learning framework for everyone.
Home-page: https://www.tensorflow.org/
Author: Google Inc.
Author-email: packages@tensorflow.org
License: Apache 2.0
Location: /usr/local/lib/python3.6/dist-packages
Requires: scipy, tensorflow-estimator, absl-py, tensorboard, grpcio, keras-preprocessing, six, google-pasta,
Required-by: fancyimpute
Collecting plot_model
  Downloading https://files.pythonhosted.org/packages/62/b8/0967e30391a7c07002c5e7bca868763dcfd2
Installing collected packages: plot-model
Successfully installed plot-model-0.20
Requirement already satisfied: tensorboardcolab in /usr/local/lib/python3.6/dist-packages (0.0.22)
Using TensorFlow backend.
```

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk

Enter your authorization code:
··········
Mounted at /content/drive
```

Here we Load all the images,preprocess it and extract feature vector using pretrained Xception n

```
# create image data augmentation generator
datagen = ImageDataGenerator(
        rotation_range = 15,  # randomly rotate images in the range (degrees, 0 to 180)
        zoom_range = 0.2, # Randomly zoom image
        width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
        height_shift_range=0.1,  # randomly shift images vertically (fraction of total height)
        horizontal_flip = True )


#load Xception model
model = Xception(weights='imagenet')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_
91889664/91884032 [==============================] - 1s 0us/step
```

```
#to get extracted image features
def extract_features(directory,model):
    # re-structure the model
    model =  Model(inputs=model.inputs, outputs=model.layers[-2].output)
    model.summary()
    # extract features from each photo
    features = dict()
    for name in tqdm(listdir(directory)):
        # get image id
        image_id = name.split('.')[0]
```

```
        image_id = name.split('.')[0]
        # load an image from file
        filename = path.join(directory, name)
        image = load_img(filename, target_size=(299, 299))
        # convert the image pixels to a numpy array
        image = img_to_array(image)
        # reshape data for the model
        image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
        image1 = preprocess_input(image)
        feature = model.predict(image1, verbose=0)
        features[image_id] = feature
        it = datagen.flow(image, batch_size=1)
        # generate samples
        for i in range(1,3):
            # generate batch of images
            batch = it.next()
            # prepare the image for the xception model
            image = preprocess_input(batch)
            # get features
            feature = model.predict(image, verbose=0)
            # store feature
            id=image_id+str(i)
            features[id] = feature
    return features


# extract features from all images
directory = '/content/drive/My Drive/images'
image_extracted_features = extract_features(directory,model)
```

```
  0%|          | 0/7470 [00:00<?, ?it/s]Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================
input_1 (InputLayer)         [(None, 299, 299, 3)      0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 149, 149, 32)      864       input_1[0][0]
_________________________________________________________________
block1_conv1_bn (BatchNormaliza (None, 149, 149, 32)   128       block1_conv1[0][0]
_________________________________________________________________
block1_conv1_act (Activation)   (None, 149, 149, 32)   0         block1_conv1_bn[0][0]
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 147, 147, 64)      18432     block1_conv1_act[0][0]
_________________________________________________________________
block1_conv2_bn (BatchNormaliza (None, 147, 147, 64)   256       block1_conv2[0][0]
_________________________________________________________________
block1_conv2_act (Activation)   (None, 147, 147, 64)   0         block1_conv2_bn[0][0]
_________________________________________________________________
block2_sepconv1 (SeparableConv2 (None, 147, 147, 128 8768      block1_conv2_act[0][0]
_________________________________________________________________
block2_sepconv1_bn (BatchNormal (None, 147, 147, 128 512       block2_sepconv1[0][0]
_________________________________________________________________
block2_sepconv2_act (Activation (None, 147, 147, 128 0         block2_sepconv1_bn[0][0]
_________________________________________________________________
block2_sepconv2 (SeparableConv2 (None, 147, 147, 128 17536     block2_sepconv2_act[0][0]
_________________________________________________________________
block2_sepconv2_bn (BatchNormal (None, 147, 147, 128 512       block2_sepconv2[0][0]
_________________________________________________________________
conv2d (Conv2D)              (None, 74, 74, 128)  8192        block1_conv2_act[0][0]
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 74, 74, 128)  0         block2_sepconv2_bn[0][0]
_________________________________________________________________
batch_normalization (BatchNorma (None, 74, 74, 128)  512       conv2d[0][0]
_________________________________________________________________
add (Add)                    (None, 74, 74, 128)  0         block2_pool[0][0]
                                                           batch_normalization[0][0]
_________________________________________________________________
block3_sepconv1_act (Activation (None, 74, 74, 128)  0         add[0][0]
_________________________________________________________________
block3_sepconv1 (SeparableConv2 (None, 74, 74, 256)  33920     block3_sepconv1_act[0][0]
_________________________________________________________________
block3_sepconv1_bn (BatchNormal (None, 74, 74, 256)  1024      block3_sepconv1[0][0]
_________________________________________________________________
block3_sepconv2_act (Activation (None, 74, 74, 256)  0         block3_sepconv1_bn[0][0]
_________________________________________________________________
block3_sepconv2 (SeparableConv2 (None, 74, 74, 256)  67840     block3_sepconv2_act[0][0]
_________________________________________________________________
block3_sepconv2_bn (BatchNormal (None, 74, 74, 256)  1024      block3_sepconv2[0][0]
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 37, 37, 256)  32768     add[0][0]
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 37, 37, 256)  0         block3_sepconv2_bn[0][0]
_________________________________________________________________
batch_normalization_1 (BatchNor (None, 37, 37, 256)  1024      conv2d_1[0][0]
_________________________________________________________________
add_1 (Add)                  (None, 37, 37, 256)  0         block3_pool[0][0]
                                                           batch_normalization_1[0][0]
```

```
block4_sepconv1_act (Activation (None, 37, 37, 256)  0           add_1[0][0]
__________________________________________________________________________________________________
block4_sepconv1 (SeparableConv2 (None, 37, 37, 728)  188672      block4_sepconv1_act[0][0]
__________________________________________________________________________________________________
block4_sepconv1_bn (BatchNormal (None, 37, 37, 728)  2912        block4_sepconv1[0][0]
__________________________________________________________________________________________________
block4_sepconv2_act (Activation (None, 37, 37, 728)  0           block4_sepconv1_bn[0][0]
__________________________________________________________________________________________________
block4_sepconv2 (SeparableConv2 (None, 37, 37, 728)  536536      block4_sepconv2_act[0][0]
__________________________________________________________________________________________________
block4_sepconv2_bn (BatchNormal (None, 37, 37, 728)  2912        block4_sepconv2[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 19, 19, 728)  186368      add_1[0][0]
__________________________________________________________________________________________________
block4_pool (MaxPooling2D)      (None, 19, 19, 728)  0           block4_sepconv2_bn[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 19, 19, 728)  2912        conv2d_2[0][0]
__________________________________________________________________________________________________
add_2 (Add)                     (None, 19, 19, 728)  0           block4_pool[0][0]
                                                                 batch_normalization_2[0][0]
__________________________________________________________________________________________________
block5_sepconv1_act (Activation (None, 19, 19, 728)  0           add_2[0][0]
__________________________________________________________________________________________________
block5_sepconv1 (SeparableConv2 (None, 19, 19, 728)  536536      block5_sepconv1_act[0][0]
__________________________________________________________________________________________________
block5_sepconv1_bn (BatchNormal (None, 19, 19, 728)  2912        block5_sepconv1[0][0]
__________________________________________________________________________________________________
block5_sepconv2_act (Activation (None, 19, 19, 728)  0           block5_sepconv1_bn[0][0]
__________________________________________________________________________________________________
block5_sepconv2 (SeparableConv2 (None, 19, 19, 728)  536536      block5_sepconv2_act[0][0]
__________________________________________________________________________________________________
block5_sepconv2_bn (BatchNormal (None, 19, 19, 728)  2912        block5_sepconv2[0][0]
__________________________________________________________________________________________________
block5_sepconv3_act (Activation (None, 19, 19, 728)  0           block5_sepconv2_bn[0][0]
__________________________________________________________________________________________________
block5_sepconv3 (SeparableConv2 (None, 19, 19, 728)  536536      block5_sepconv3_act[0][0]
__________________________________________________________________________________________________
block5_sepconv3_bn (BatchNormal (None, 19, 19, 728)  2912        block5_sepconv3[0][0]
__________________________________________________________________________________________________
add_3 (Add)                     (None, 19, 19, 728)  0           block5_sepconv3_bn[0][0]
                                                                 add_2[0][0]
__________________________________________________________________________________________________
block6_sepconv1_act (Activation (None, 19, 19, 728)  0           add_3[0][0]
__________________________________________________________________________________________________
block6_sepconv1 (SeparableConv2 (None, 19, 19, 728)  536536      block6_sepconv1_act[0][0]
__________________________________________________________________________________________________
block6_sepconv1_bn (BatchNormal (None, 19, 19, 728)  2912        block6_sepconv1[0][0]
__________________________________________________________________________________________________
block6_sepconv2_act (Activation (None, 19, 19, 728)  0           block6_sepconv1_bn[0][0]
__________________________________________________________________________________________________
block6_sepconv2 (SeparableConv2 (None, 19, 19, 728)  536536      block6_sepconv2_act[0][0]
__________________________________________________________________________________________________
block6_sepconv2_bn (BatchNormal (None, 19, 19, 728)  2912        block6_sepconv2[0][0]
__________________________________________________________________________________________________
block6_sepconv3_act (Activation (None, 19, 19, 728)  0           block6_sepconv2_bn[0][0]
__________________________________________________________________________________________________
block6_sepconv3 (SeparableConv2 (None, 19, 19, 728)  536536      block6_sepconv3_act[0][0]
```

block6_sepconv3_bn (BatchNormal (None, 19, 19, 728) 2912 block6_sepconv3[0][0]

add_4 (Add) (None, 19, 19, 728) 0 block6_sepconv3_bn[0][0]
add_3[0][0]

block7_sepconv1_act (Activation (None, 19, 19, 728) 0 add_4[0][0]

block7_sepconv1 (SeparableConv2 (None, 19, 19, 728) 536536 block7_sepconv1_act[0][0]

block7_sepconv1_bn (BatchNormal (None, 19, 19, 728) 2912 block7_sepconv1[0][0]

block7_sepconv2_act (Activation (None, 19, 19, 728) 0 block7_sepconv1_bn[0][0]

block7_sepconv2 (SeparableConv2 (None, 19, 19, 728) 536536 block7_sepconv2_act[0][0]

block7_sepconv2_bn (BatchNormal (None, 19, 19, 728) 2912 block7_sepconv2[0][0]

block7_sepconv3_act (Activation (None, 19, 19, 728) 0 block7_sepconv2_bn[0][0]

block7_sepconv3 (SeparableConv2 (None, 19, 19, 728) 536536 block7_sepconv3_act[0][0]

block7_sepconv3_bn (BatchNormal (None, 19, 19, 728) 2912 block7_sepconv3[0][0]

add_5 (Add) (None, 19, 19, 728) 0 block7_sepconv3_bn[0][0]
add_4[0][0]

block8_sepconv1_act (Activation (None, 19, 19, 728) 0 add_5[0][0]

block8_sepconv1 (SeparableConv2 (None, 19, 19, 728) 536536 block8_sepconv1_act[0][0]

block8_sepconv1_bn (BatchNormal (None, 19, 19, 728) 2912 block8_sepconv1[0][0]

block8_sepconv2_act (Activation (None, 19, 19, 728) 0 block8_sepconv1_bn[0][0]

block8_sepconv2 (SeparableConv2 (None, 19, 19, 728) 536536 block8_sepconv2_act[0][0]

block8_sepconv2_bn (BatchNormal (None, 19, 19, 728) 2912 block8_sepconv2[0][0]

block8_sepconv3_act (Activation (None, 19, 19, 728) 0 block8_sepconv2_bn[0][0]

block8_sepconv3 (SeparableConv2 (None, 19, 19, 728) 536536 block8_sepconv3_act[0][0]

block8_sepconv3_bn (BatchNormal (None, 19, 19, 728) 2912 block8_sepconv3[0][0]

add_6 (Add) (None, 19, 19, 728) 0 block8_sepconv3_bn[0][0]
add_5[0][0]

block9_sepconv1_act (Activation (None, 19, 19, 728) 0 add_6[0][0]

block9_sepconv1 (SeparableConv2 (None, 19, 19, 728) 536536 block9_sepconv1_act[0][0]

block9_sepconv1_bn (BatchNormal (None, 19, 19, 728) 2912 block9_sepconv1[0][0]

block9_sepconv2_act (Activation (None, 19, 19, 728) 0 block9_sepconv1_bn[0][0]

block9_sepconv2 (SeparableConv2 (None, 19, 19, 728) 536536 block9_sepconv2_act[0][0]

block9_sepconv2_bn (BatchNormal (None, 19, 19, 728) 2912 block9_sepconv2[0][0]

```
block9_sepconv3_act (Activation (None, 19, 19, 728)  0           block9_sepconv2_bn[0][0]
__________________________________________________________________________________________________
block9_sepconv3 (SeparableConv2 (None, 19, 19, 728)  536536      block9_sepconv3_act[0][0]
__________________________________________________________________________________________________
block9_sepconv3_bn (BatchNormal (None, 19, 19, 728)  2912        block9_sepconv3[0][0]
__________________________________________________________________________________________________
add_7 (Add)                     (None, 19, 19, 728)  0           block9_sepconv3_bn[0][0]
                                                                 add_6[0][0]
__________________________________________________________________________________________________
block10_sepconv1_act (Activatio (None, 19, 19, 728)  0           add_7[0][0]
__________________________________________________________________________________________________
block10_sepconv1 (SeparableConv (None, 19, 19, 728)  536536      block10_sepconv1_act[0][0]
__________________________________________________________________________________________________
block10_sepconv1_bn (BatchNorma (None, 19, 19, 728)  2912        block10_sepconv1[0][0]
__________________________________________________________________________________________________
block10_sepconv2_act (Activatio (None, 19, 19, 728)  0           block10_sepconv1_bn[0][0]
__________________________________________________________________________________________________
block10_sepconv2 (SeparableConv (None, 19, 19, 728)  536536      block10_sepconv2_act[0][0]
__________________________________________________________________________________________________
block10_sepconv2_bn (BatchNorma (None, 19, 19, 728)  2912        block10_sepconv2[0][0]
__________________________________________________________________________________________________
block10_sepconv3_act (Activatio (None, 19, 19, 728)  0           block10_sepconv2_bn[0][0]
__________________________________________________________________________________________________
block10_sepconv3 (SeparableConv (None, 19, 19, 728)  536536      block10_sepconv3_act[0][0]
__________________________________________________________________________________________________
block10_sepconv3_bn (BatchNorma (None, 19, 19, 728)  2912        block10_sepconv3[0][0]
__________________________________________________________________________________________________
add_8 (Add)                     (None, 19, 19, 728)  0           block10_sepconv3_bn[0][0]
                                                                 add_7[0][0]
__________________________________________________________________________________________________
block11_sepconv1_act (Activatio (None, 19, 19, 728)  0           add_8[0][0]
__________________________________________________________________________________________________
block11_sepconv1 (SeparableConv (None, 19, 19, 728)  536536      block11_sepconv1_act[0][0]
__________________________________________________________________________________________________
block11_sepconv1_bn (BatchNorma (None, 19, 19, 728)  2912        block11_sepconv1[0][0]
__________________________________________________________________________________________________
block11_sepconv2_act (Activatio (None, 19, 19, 728)  0           block11_sepconv1_bn[0][0]
__________________________________________________________________________________________________
block11_sepconv2 (SeparableConv (None, 19, 19, 728)  536536      block11_sepconv2_act[0][0]
__________________________________________________________________________________________________
block11_sepconv2_bn (BatchNorma (None, 19, 19, 728)  2912        block11_sepconv2[0][0]
__________________________________________________________________________________________________
block11_sepconv3_act (Activatio (None, 19, 19, 728)  0           block11_sepconv2_bn[0][0]
__________________________________________________________________________________________________
block11_sepconv3 (SeparableConv (None, 19, 19, 728)  536536      block11_sepconv3_act[0][0]
__________________________________________________________________________________________________
block11_sepconv3_bn (BatchNorma (None, 19, 19, 728)  2912        block11_sepconv3[0][0]
__________________________________________________________________________________________________
add_9 (Add)                     (None, 19, 19, 728)  0           block11_sepconv3_bn[0][0]
                                                                 add_8[0][0]
__________________________________________________________________________________________________
block12_sepconv1_act (Activatio (None, 19, 19, 728)  0           add_9[0][0]
__________________________________________________________________________________________________
block12_sepconv1 (SeparableConv (None, 19, 19, 728)  536536      block12_sepconv1_act[0][0]
__________________________________________________________________________________________________
block12_sepconv1_bn (BatchNorma (None, 19, 19, 728)  2912        block12_sepconv1[0][0]
__________________________________________________________________________________________________
```

```
block12_sepconv2_act (Activatio (None, 19, 19, 728)  0          block12_sepconv1_bn[0][0]
__________________________________________________________________________________________________
block12_sepconv2 (SeparableConv (None, 19, 19, 728)  536536     block12_sepconv2_act[0][0]
__________________________________________________________________________________________________
block12_sepconv2_bn (BatchNorma (None, 19, 19, 728)  2912       block12_sepconv2[0][0]
__________________________________________________________________________________________________
block12_sepconv3_act (Activatio (None, 19, 19, 728)  0          block12_sepconv2_bn[0][0]
__________________________________________________________________________________________________
block12_sepconv3 (SeparableConv (None, 19, 19, 728)  536536     block12_sepconv3_act[0][0]
__________________________________________________________________________________________________
block12_sepconv3_bn (BatchNorma (None, 19, 19, 728)  2912       block12_sepconv3[0][0]
__________________________________________________________________________________________________
add_10 (Add)                    (None, 19, 19, 728)  0          block12_sepconv3_bn[0][0]
                                                                add_9[0][0]
__________________________________________________________________________________________________
block13_sepconv1_act (Activatio (None, 19, 19, 728)  0          add_10[0][0]
__________________________________________________________________________________________________
block13_sepconv1 (SeparableConv (None, 19, 19, 728)  536536     block13_sepconv1_act[0][0]
__________________________________________________________________________________________________
block13_sepconv1_bn (BatchNorma (None, 19, 19, 728)  2912       block13_sepconv1[0][0]
__________________________________________________________________________________________________
block13_sepconv2_act (Activatio (None, 19, 19, 728)  0          block13_sepconv1_bn[0][0]
__________________________________________________________________________________________________
block13_sepconv2 (SeparableConv (None, 19, 19, 1024) 752024     block13_sepconv2_act[0][0]
__________________________________________________________________________________________________
block13_sepconv2_bn (BatchNorma (None, 19, 19, 1024) 4096       block13_sepconv2[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 10, 10, 1024) 745472     add_10[0][0]
__________________________________________________________________________________________________
block13_pool (MaxPooling2D)     (None, 10, 10, 1024) 0          block13_sepconv2_bn[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 10, 10, 1024) 4096       conv2d_3[0][0]
__________________________________________________________________________________________________
add_11 (Add)                    (None, 10, 10, 1024) 0          block13_pool[0][0]
                                                                batch_normalization_3[0][0]
__________________________________________________________________________________________________
block14_sepconv1 (SeparableConv (None, 10, 10, 1536) 1582080    add_11[0][0]
__________________________________________________________________________________________________
block14_sepconv1_bn (BatchNorma (None, 10, 10, 1536) 6144       block14_sepconv1[0][0]
__________________________________________________________________________________________________
block14_sepconv1_act (Activatio (None, 10, 10, 1536) 0          block14_sepconv1_bn[0][0]
__________________________________________________________________________________________________
block14_sepconv2 (SeparableConv (None, 10, 10, 2048) 3159552    block14_sepconv1_act[0][0]
__________________________________________________________________________________________________
block14_sepconv2_bn (BatchNorma (None, 10, 10, 2048) 8192       block14_sepconv2[0][0]
__________________________________________________________________________________________________
block14_sepconv2_act (Activatio (None, 10, 10, 2048) 0          block14_sepconv2_bn[0][0]
__________________________________________________________________________________________________
avg_pool (GlobalAveragePooling2 (None, 2048)         0          block14_sepconv2_act[0][0]
==================================================================================================
Total params: 20,861,480
Trainable params: 20,806,952
Non-trainable params: 54,528
__________________________________________________________________________________________________
100%|██████████| 7470/7470 [1:47:55<00:00,  1.15it/s]
```

```
print('Number of total images: ', len(image_extracted_features))
```

```
print('Extracted Feature vector size: ' ,len(image_extracted_features['CXR1_1_IM-0001-3001'][0]))
```

```
Number of total images:  22410
Extracted Feature vector size:  2048
```

2) Load text data

```
# this code is for extracting data from xml files
# xml's sample code learned from https: //www.youtube.com/watch?v=PNNg4xKbCtA
#https://docs.python.org/3.4/library/xml.etree.elementtree.html
id_impression=dict()
id_finding=dict()
directory = 'reports'

for filename in tqdm(listdir(directory)):
    if filename.endswith(".xml"):
        f=path.join(directory,filename)
        tree = ET.parse(f)
        root = tree.getroot()
        for child in root:
            if child.tag=='MedlineCitation':
                for attr in child:
                    if attr.tag=='Article':
                        for i in attr:
                            if i.tag=='Abstract':
                                for name in i:
                                    if name.get('Label')=='FINDINGS':
                                        finding=name.text
                                    elif name.get('Label')=='IMPRESSION':
                                        impression=name.text

        for p_image in root.findall('parentImage'):
            idd = p_image.get('id')
            id_impression[idd]=impression
            id_finding[idd]=finding
            for i in range(1,3):
                id=idd+str(i)
                id_impression[id]=impression
                id_finding[id]=finding
```

```
100%|
```

Data Cleaning

- 2.1 Check for None values

```
count1=0
#finding none values in impression
for k,v in id_impression.items():
    if id_impression[k] is None:
        count1=count1+1
```

```
print("Impression data contains",count1,"None Values")
```

Impression data contains 156 None Values

```
count=0
#finding none values in finding
for k,v in id_finding.items():
    if id_finding[k] is None:
        count=count+1
print("Finding data contains",count,"None Values")
```

Finding data contains 2991 None Values

```
count=0
#finding none values in findings and impressions
for k,v in id_finding.items():
    if (id_finding[k] is None) and (id_impression[k] is None) :
        count=count+1
print("There are",count,"datapoints whose Finding and impressions data are None")
```

There are 120 datapoints whose Finding and impressions data are None

```
count=0
#finding none values in finding or impression
for k,v in id_finding.items():
    if (id_finding[k] is None) or (id_impression[k] is None) :
        count=count+1
print("There are",count,"datapoints whose Finding or impressions data are None")
```

There are3027datapoints whose Finding or impressions data are None

2.2 Clean missing data

- We delete entries whose impressions are None

```
# removing none impressions
id_impression_none_removed = { k : v for k,v in id_impression.items() if v is not None}
print("After removing None contained impressions,number of final impressions: "+str(len(id_impression)) +" - "+s
```

After removing None contained impressions,number of final impressions: 22410 - 156 = 22254

- We delete entries whose impressions or findings are None

```
#finding none values in finding
count=0
finding_for_2nd_model=dict()
#impression_for_2nd_model=dict()
for k,v in id_finding.items():
    if (id_finding[k] is None) or (id_impression[k] is None) :
        count=count+1
        continue
```

```
    else:
        finding_for_2nd_model[k]=v
        #impression_for_2nd_model[k]=id_impression[k]

print("There are",count,"datapoints whose Finding or impressions data are None , so we have removed them.")
print("Number of final Finding datapoints: "+str(len(id_finding))+" - ",count,"=",len(finding_for_2nd_model))
```

```
There are 3027 datapoints whose Finding or impressions data are None , so we have removed them.
Number of final Finding datapoints: 22410-3027= 19383
```

2.3 Text Preprocessing

```
def clean_descriptions(descriptions,add_token):
    """this function cleans decription text"""
    descriptionss=dict()
    for key, desc in descriptions.items():
        sent = desc.replace('x-XXXX',' ')
        sent = sent.lower()
        sent = sent.replace('xxxx',' ')
        sent = sent.replace('x-xxxx',' ')
        sent = re.sub('[^A-Za-z]+', ' ', sent)
        if add_token=='yes':
            sent = 'startseq ' +sent+ ' endseq'
        descriptionss[key]=sent.strip()
    return descriptionss

cleaned_id_impressions = clean_descriptions(id_impression_none_removed,'yes')
cleaned_id_findings = clean_descriptions(finding_for_2nd_model,'no')
```

Train ,cv and test split

```
keys=list(cleaned_id_findings.keys())
random.shuffle(keys)
train_id,cv_id,test_id=keys[0:18883],keys[18883:19133],keys[19133:]

train_id = dict.fromkeys(train_id ,1)
cv_id = dict.fromkeys(cv_id ,1)
test_id = dict.fromkeys(test_id ,1)
print("train size :\t",len(train_id))
print("cv size     :\t",len(cv_id))
print("test size  :\t",len(test_id))
```

```
train size :    18883
cv size     :    250
test size  :    250
```

Define Necessary Functions

```
def create_sequences(tokenizer, max_length, descriptions, image_features,finding_features,vocab_size):
    X1, X2,X3, y = list(), list(),list(), list()
    # walk through each image identifier
    for key, desc in descriptions.items():
        # encode the sequence
        seq = tokenizer.texts_to_sequences([desc])[0]
        # split one sequence into multiple X,y pairs
        for i in range(1, len(seq)):
            # split into input and output pair
            in_seq, out_seq = seq[:i], seq[i]
            # pad input sequence
            in_seq = pad_sequences([in_seq], maxlen=max_length)[0]
            # encode output sequence
            out_seq = to_categorical([out_seq], num_classes=vocab_size)[0]
            # store
            X1.append(image_features[key][0])
            #print(image_features[key][0])
            X2.append(finding_features[key])
            #print(finding_features[key])
            X3.append(in_seq)
            y.append(out_seq)
    return array(X1),array(X2), array(X3), array(y)


def finding_sequnces(tokenizer,max_length,id_findings):
    finding_sequences = dict()
    for k,v in id_findings.items():
        seq = tokenizer.texts_to_sequences([v])[0]
        seq = pad_sequences([seq], maxlen=max_length)[0]
        finding_sequences[k]=seq
    return finding_sequences

# load clean respective set into memory
def load_respective_set(dictt,dataset):
    """ to load description of given dataset"""
    descriptions = dict()
    for k,v in dataset.items():
        descriptions[k]=dictt[k]
    return descriptions


def load_image_features(dictt, dataset):
    """ to load image features of given dataset"""
    features = {k: dictt[k] for k in dataset}
    return features


def create_tokenizer(descriptions):
    '''fit a tokenizer for given descriptions '''
    lines = list(descriptions.values())
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(lines)
    return tokenizer
```

Prepare Train Data

```
train_image_features = load_image_features(image_extracted_features,train_id)
print('\nTotal train images      : ' ,len(train_image_features))

#-------------------------------------------------------------------------------------------
train_id_findings=load_respective_set(cleaned_id_findings, train_id)
print('\nTotal train findings    : ',len(train_id_findings))

findings_max_length = max(len(s.split()) for s in list(train_id_findings.values()))
print('\nMaximum Length of Findings: ',findings_max_length)

findings_tokenizer = create_tokenizer(train_id_findings)
train_id_finding_sequnces = finding_sequnces(findings_tokenizer,findings_max_length,train_id_findings)

findings_vocab_size = len(findings_tokenizer.word_index) + 1
print('\nVocab size of Findings: ',findings_vocab_size)

glove_words = pickle.load(open('/content/drive/My Drive/Xception/glove_vectors', 'rb'))
findings_embedding_matrix = np.zeros((findings_vocab_size, 300))
for word, i in findings_tokenizer.word_index.items():
    embedding_vector = glove_words.get(word)
    if embedding_vector is not None:
        findings_embedding_matrix[i] = embedding_vector

print("\n-------------------------------------------------------------------------------\n")

train_id_impressions=load_respective_set(cleaned_id_impressions, train_id)
print('\nTotal train impressions : ',len(train_id_impressions))

impressions_max_length = max(len(s.split()) for s in list(train_id_impressions.values()))
print('\nDescription maximum Length: ',impressions_max_length)

# prepare tokenizer
impressions_tokenizer = create_tokenizer(train_id_impressions)

impressions_vocab_size = len(impressions_tokenizer.word_index) + 1
print('\n Impressions Vocabulary Size: ', impressions_vocab_size)

impressions_embedding_matrix = np.zeros((impressions_vocab_size, 300))
for word, i in impressions_tokenizer.word_index.items():
    embedding_vector = glove_words.get(word)
    if embedding_vector is not None:
        impressions_embedding_matrix[i] = embedding_vector

# pad to fixed length
X1train, X2train, X3train, ytrain = create_sequences(impressions_tokenizer,impressions_max_length ,train_id_impr
```

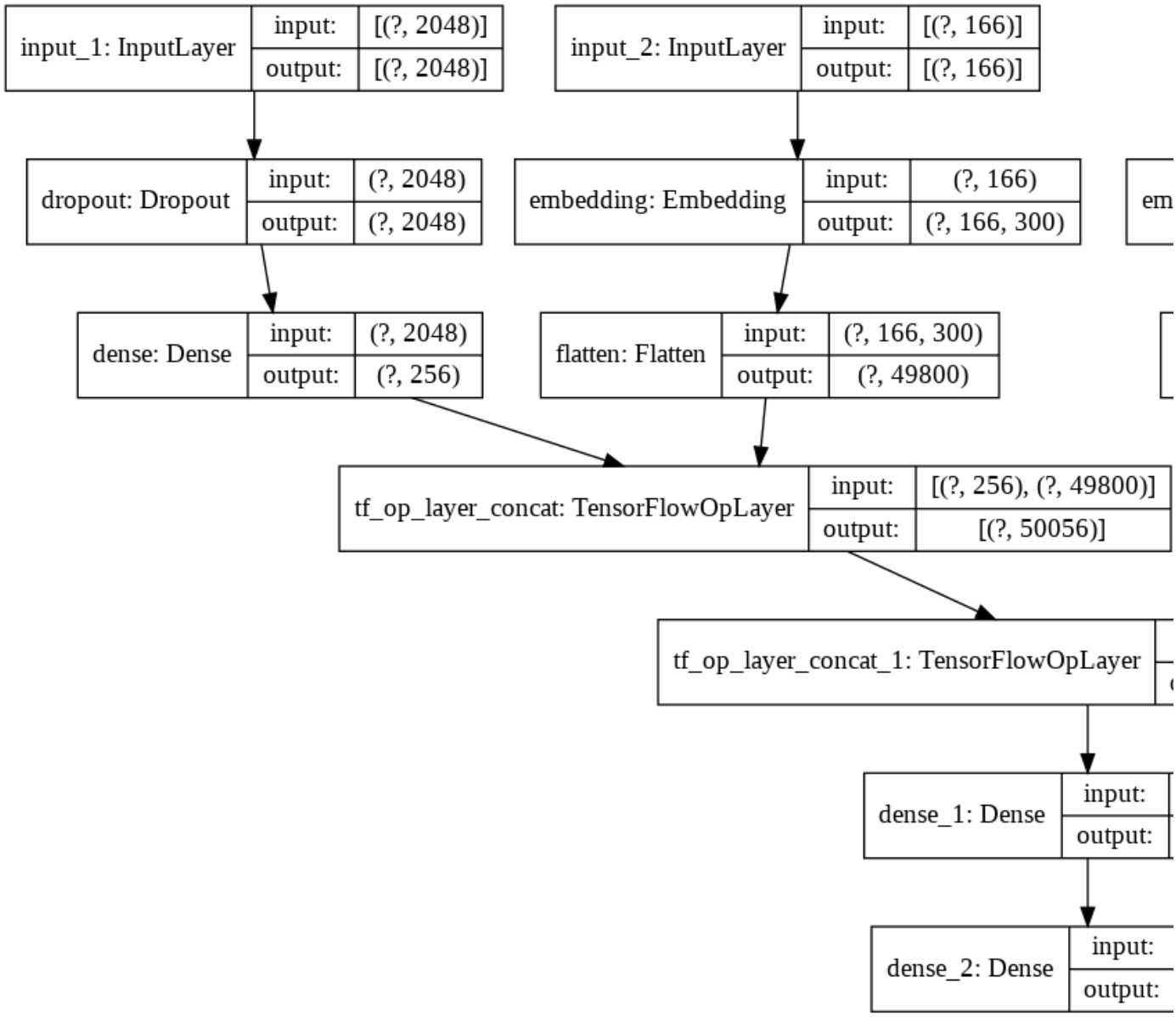
```
Total train images      :  18883

Total train findings    :  18883

Maximum Length of Findings:  166

Vocab size of Findings:  1563

--------------------------------------------------------------------------------

Total train impressions :  18883

Description maximum Length:  114

 Impressions Vocabulary Size:  1208
```

Prepare CV Data

```
cv_image_features = load_image_features(image_extracted_features,cv_id)
print('\nTotal cv images      : ' ,len(cv_image_features))

cv_id_findings=load_respective_set(cleaned_id_findings, cv_id)
print('\nTotal cv findings    : ',len(cv_id_findings))

cv_id_finding_sequnces = finding_sequnces(findings_tokenizer,findings_max_length,cv_id_findings)

print("\n---------------------------------\n")

cv_id_impressions=load_respective_set(cleaned_id_impressions, cv_id)
print('Total cv impressions : ',len(cv_id_impressions))

X1cv, X2cv, X3cv,ycv = create_sequences(impressions_tokenizer, impressions_max_length,cv_id_impressions,cv_im
```

```
Total cv images      :  250

Total cv findings    :  250

---------------------------------

Total cv impressions :  250
```

Build Deep Learning Model : ([image, findings],impression)



Define model

```
# define the captioning model
def define_model(findings_max_length,findings_vocab_size,impressions_max_length,impressions_vocab_size,findi
    # feature extractor model
    inputs1 = tf.keras.layers.Input(shape=(2048,))
    image1 = tf.keras.layers.Dropout(0.5)(inputs1)
    image2 = tf.keras.layers.Dense(256, activation='relu')(image1)

    # finding model
    inputs2 = tf.keras.layers.Input(shape=(findings_max_length,))
    findings1 = tf.keras.layers.Embedding(findings_vocab_size,300,weights=[findings_embedding_matrix],trainable
    findings2 = tf.keras.layers.Flatten()(findings1)

    im_f = tf.concat([image2, findings2],axis=1)

    # sequence model
    inputs3 = tf.keras.layers.Input(shape=(impressions_max_length,))
    impressions1 = tf.keras.layers.Embedding(impressions_vocab_size,300,weights=[impressions_embedding_matr
```

```
    impressions1 = tf.keras.layers.Embedding(impressions_vocab_size,500,weights=[impressions_embedding_matr
    impressions2 = tf.keras.layers.Dropout(0.5)(impressions1)
    impressions3 = tf.keras.layers.LSTM(256)(impressions2)

    # decoder model
    decoder1= tf.concat([im_f, impressions3],axis=1)
    decoder2 = tf.keras.layers.Dense(500, activation='relu')(decoder1)
    outputs = tf.keras.layers.Dense(impressions_vocab_size, activation='softmax')(decoder2)
    # tie it together [image, seq] [word]
    model = tf.keras.models.Model(inputs=[inputs1, inputs2, inputs3], outputs=outputs)
    # compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam')
    # summarize model
    model.summary()
    plot_model(model, to_file='/content/drive/My Drive/Xception/Model2/finalmodel.png', show_shapes=True)
    return model

# define the model
model = define_model(findings_max_length,findings_vocab_size,impressions_max_length,impressions_vocab_size
```

Model: "model"

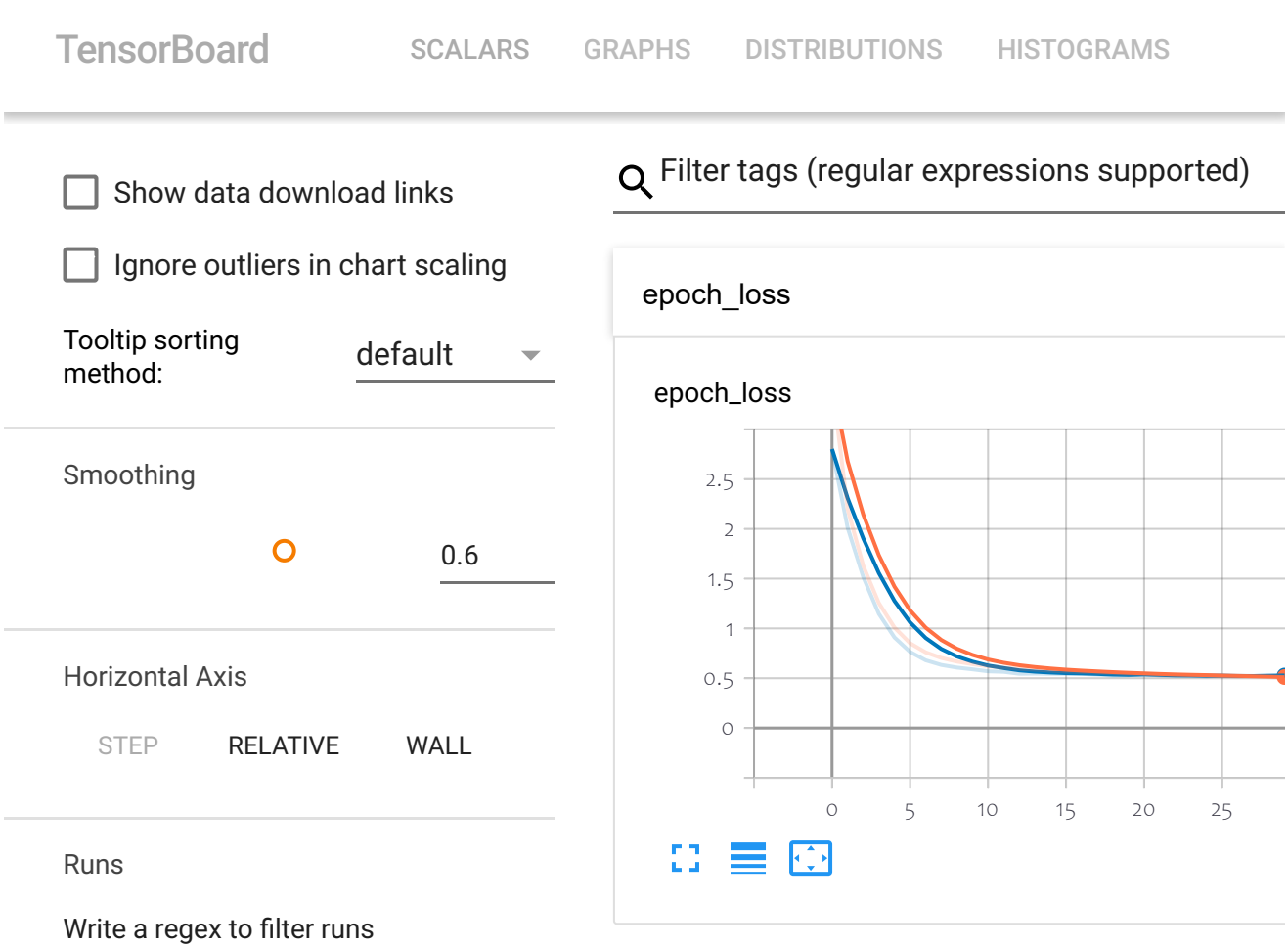
Run Model

```
xtrain=[X1train, X2train , X3train]
xcv=[X1cv, X2cv, X3cv]
# define checkpoint callback
checkpoint = tf.keras.callbacks.ModelCheckpoint('/content/drive/My Drive/Xception/Model2/finalcheck.hdf5', mo

log_dir="/content/drive/My Drive/Xception/Model2/finaltensorboardlogs/logs/fit/" + datetime.now().strftime("%
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1, write_graph=True,write

# fit model
h=model.fit(xtrain, ytrain,batch_size=512, epochs=25, verbose=2,callbacks=[tensorboard_callback,checkpoint], va
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
Epoch 1/25

Epoch 00001: val_loss improved from inf to 1.69536, saving model to /content/drive/My Drive/Xception/M
314/314 - 161s - loss: 2.6970 - val_loss: 1.6954
Epoch 2/25

Epoch 00002: val_loss improved from 1.69536 to 0.99211, saving model to /content/drive/My Drive/Xcepti
314/314 - 160s - loss: 1.1903 - val_loss: 0.9921
Epoch 3/25

Epoch 00003: val_loss improved from 0.99211 to 0.67842, saving model to /content/drive/My Drive/Xcepti
314/314 - 160s - loss: 0.7286 - val_loss: 0.6784
Epoch 4/25

Epoch 00004: val_loss improved from 0.67842 to 0.51504, saving model to /content/drive/My Drive/Xcepti
314/314 - 159s - loss: 0.4776 - val_loss: 0.5150
Epoch 5/25

Epoch 00005: val_loss improved from 0.51504 to 0.36081, saving model to /content/drive/My Drive/Xcepti
314/314 - 158s - loss: 0.3292 - val_loss: 0.3608
Epoch 6/25

Epoch 00006: val_loss improved from 0.36081 to 0.25499, saving model to /content/drive/My Drive/Xcepti
314/314 - 158s - loss: 0.2474 - val_loss: 0.2550
Epoch 7/25

Epoch 00007: val_loss improved from 0.25499 to 0.22702, saving model to /content/drive/My Drive/Xcepti
314/314 - 158s - loss: 0.1933 - val_loss: 0.2270
Epoch 8/25

Epoch 00008: val_loss improved from 0.22702 to 0.15800, saving model to /content/drive/My Drive/Xcepti
314/314 - 157s - loss: 0.1587 - val_loss: 0.1580
Epoch 9/25

Epoch 00009: val_loss improved from 0.15800 to 0.13293, saving model to /content/drive/My Drive/Xcepti
314/314 - 158s - loss: 0.1382 - val_loss: 0.1329
Epoch 10/25

Epoch 00010: val_loss improved from 0.13293 to 0.12608, saving model to /content/drive/My Drive/Xcepti
314/314 - 157s - loss: 0.1309 - val_loss: 0.1261
Epoch 11/25

Epoch 00011: val_loss improved from 0.12608 to 0.10120, saving model to /content/drive/My Drive/Xcepti
314/314 - 155s - loss: 0.1142 - val_loss: 0.1012
Epoch 12/25

Epoch 00012: val_loss improved from 0.10120 to 0.09612, saving model to /content/drive/My Drive/Xcepti
314/314 - 156s - loss: 0.1052 - val_loss: 0.0961
Epoch 13/25

Epoch 00013: val_loss did not improve from 0.09612
314/314 - 155s - loss: 0.1107 - val_loss: 0.1080
Epoch 14/25

Epoch 00014: val_loss improved from 0.09612 to 0.09071, saving model to /content/drive/My Drive/Xcepti
314/314 - 155s - loss: 0.1010 - val_loss: 0.0907
Epoch 15/25
```

```
Epoch 00015: val_loss did not improve from 0.09071
314/314 - 155s - loss: 0.1040 - val_loss: 0.1046
Epoch 16/25

Epoch 00016: val_loss improved from 0.09071 to 0.05937, saving model to /content/drive/My Drive/Xcepti
314/314 - 154s - loss: 0.0959 - val_loss: 0.0594
Epoch 17/25

Epoch 00017: val_loss improved from 0.05937 to 0.05246, saving model to /content/drive/My Drive/Xcepti
314/314 - 155s - loss: 0.0865 - val_loss: 0.0525
Epoch 18/25

Epoch 00018: val_loss did not improve from 0.05246
314/314 - 154s - loss: 0.0862 - val_loss: 0.0748
Epoch 19/25

Epoch 00019: val_loss improved from 0.05246 to 0.04458, saving model to /content/drive/My Drive/Xcepti
314/314 - 154s - loss: 0.0872 - val_loss: 0.0446
Epoch 20/25

Epoch 00020: val_loss did not improve from 0.04458
314/314 - 153s - loss: 0.0801 - val_loss: 0.0640
Epoch 21/25

Epoch 00021: val_loss did not improve from 0.04458
314/314 - 153s - loss: 0.0790 - val_loss: 0.0515
Epoch 22/25

Epoch 00022: val_loss did not improve from 0.04458
314/314 - 154s - loss: 0.0731 - val_loss: 0.0502
Epoch 23/25

Epoch 00023: val_loss did not improve from 0.04458
314/314 - 153s - loss: 0.0759 - val_loss: 0.0673
Epoch 24/25

Epoch 00024: val_loss improved from 0.04458 to 0.04004, saving model to /content/drive/My Drive/Xcepti
314/314 - 154s - loss: 0.0770 - val_loss: 0.0400
Epoch 25/25

Epoch 00025: val_loss did not improve from 0.04004
314/314 - 153s - loss: 0.0852 - val_loss: 0.0431
```

```
%tensorboard --logdir='/content/drive/My Drive/Xception/Model2/finaltensorboardlogs/logs/fit'
```



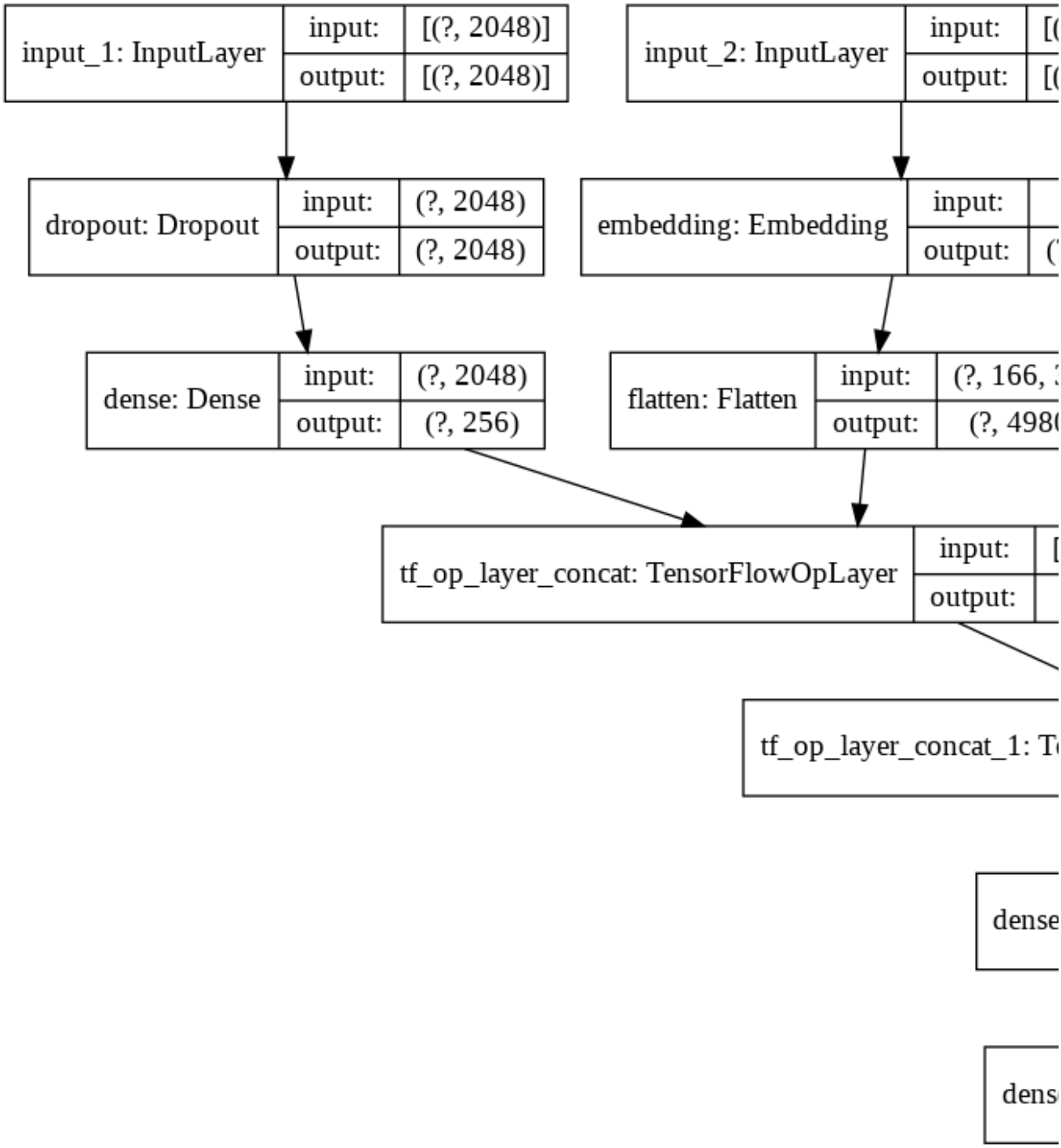
Prepare test data

```
test_image_features = load_image_features(image_extracted_features,test_id)
print('\nTotal test images      : ' ,len(test_image_features))

test_id_findings=load_respective_set(cleaned_id_findings, test_id)
print('\nTotal test findings    : ',len(test_id_findings))
```

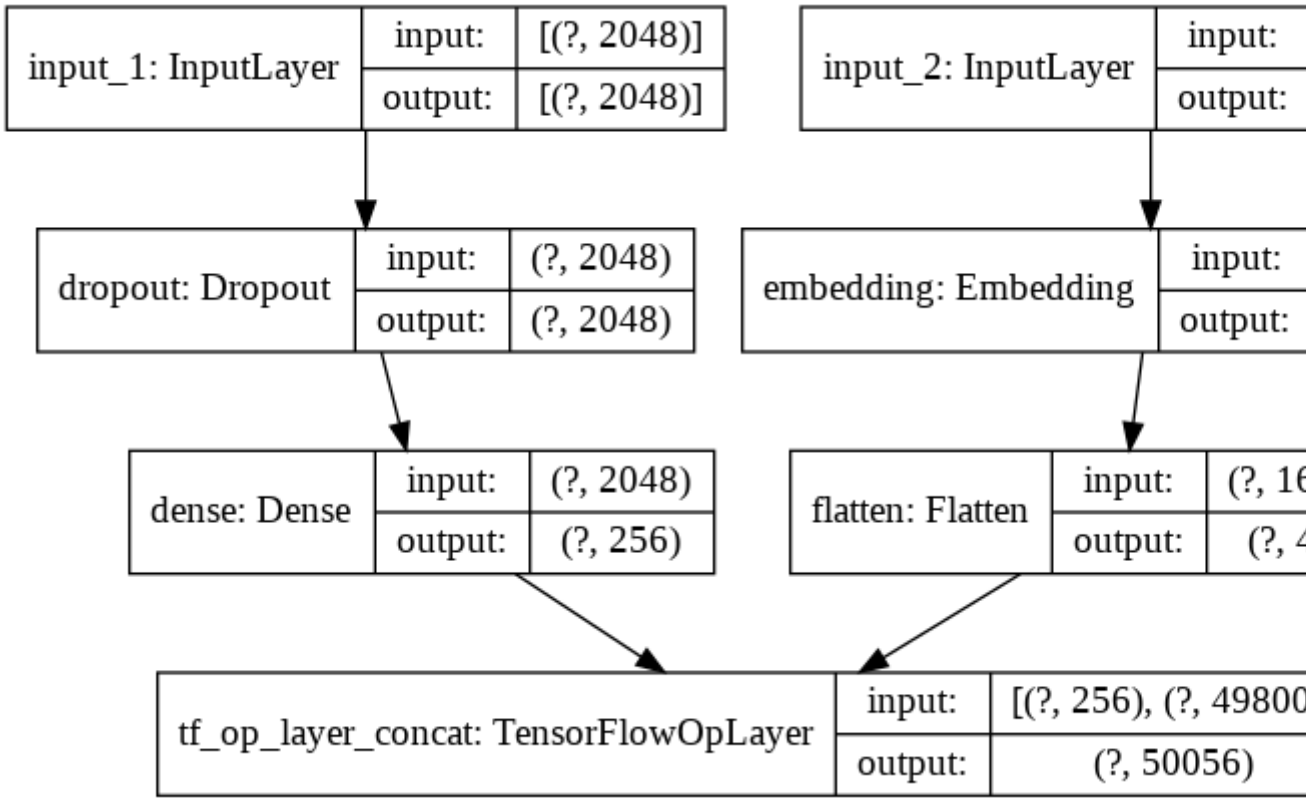
```
test_id_finding_sequnces = finding_sequnces(findings_tokenizer,findings_max_length,test_id_findings)

print("\n-------------------------------------\n")

test_id_impressions=load_respective_set(cleaned_id_impressions, test_id)
print('Total test impressions : ',len(test_id_impressions))
```

```
Total test images      :  250

Total test findings    :  250

---------------------------------------

Total test impressions :  250
```

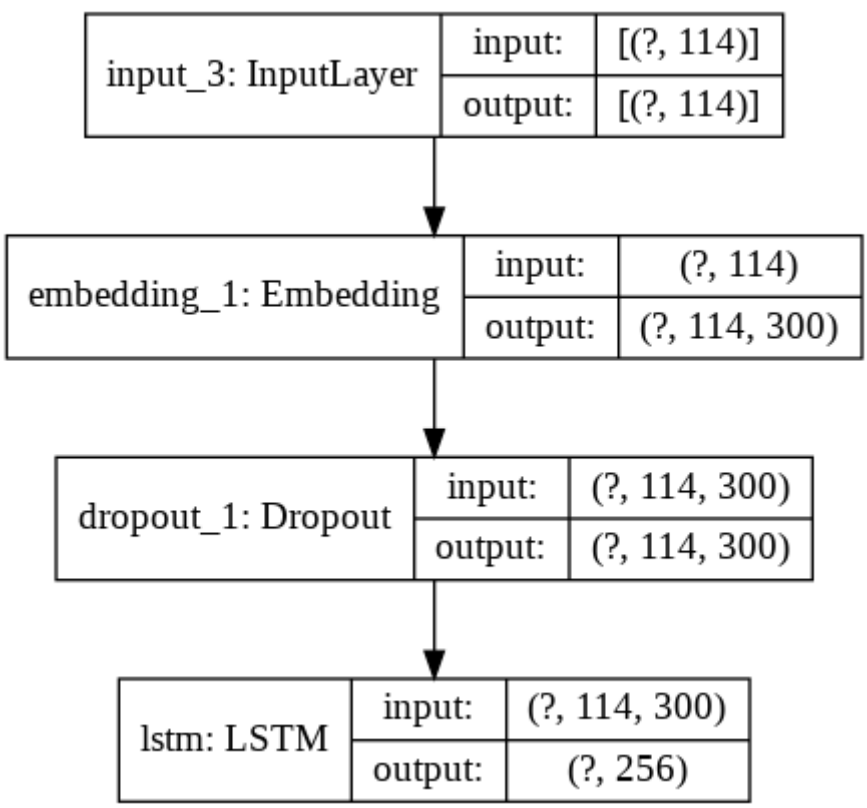
```
#here we display model architecture again just to understand further flow easily
# load the model
model1=tf.keras.models.load_model('/content/drive/My Drive/Xception/Model2/finalcheck.hdf5')
tf.keras.utils.plot_model(model1, sh1ow_shapes=True)
```



#we calling below architecture recurrently while predicting next word in the sequence so we trying to compute o
#https://stackoverflow.com/questions/41711190/keras-how-to-get-the-output-of-each-layer
#https://stackoverflow.com/questions/43452353/obtaining-output-of-an-intermediate-layer-in-tensorflow-keras
model1a = Model(inputs=model1.inputs[:2], outputs=model1.layers[-5].output)
tf.keras.utils.plot_model(model1a, show_shapes=True)

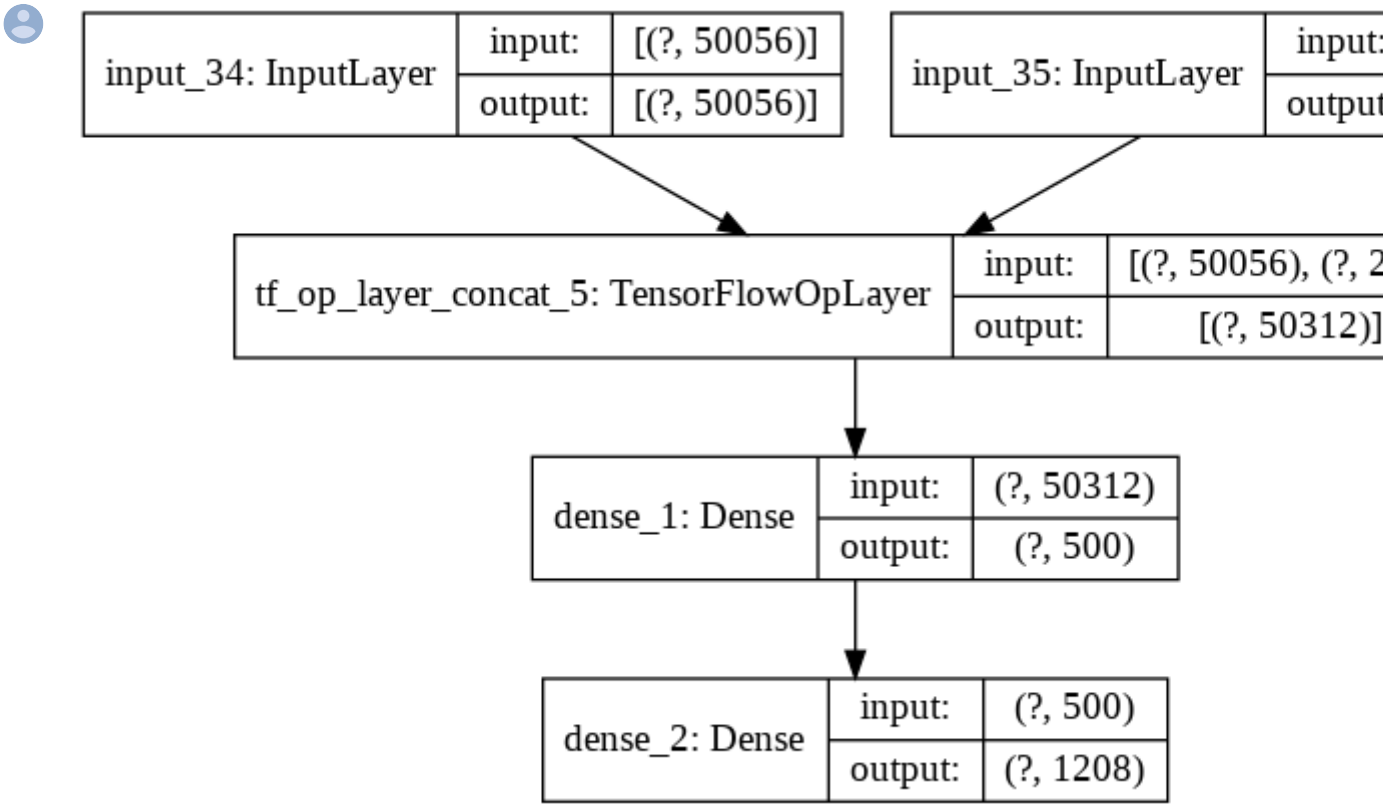


#we call below architecture for generating each word in the sequence
model1b = Model(inputs=model1.inputs[2], outputs=model1.layers[-4].output)
tf.keras.utils.plot_model(model1b, show_shapes=True)



# index of desired layer
idx = 11
# get the input shape of desired layer
input_shape = model1.layers[idx].get_input_shape_at(0)

```
a,b=input_shape
layer_input1 = Input(shape=a[1])
layer_input2 = Input(shape=b[1])

merged_input= tf.concat([layer_input1, layer_input2],axis=1)
x = merged_input
for layer1 in model1.layers[12:]:
    x = layer1(x)

#https://keras.io/getting-started/functional-api-guide/
# create the model
new_model1 = Model(inputs=[layer_input1,layer_input2],outputs= x)
new_model1.load_weights("/content/drive/My Drive/Xception/Model2/finalcheck.hdf5", by_name=True)
tf.keras.utils.plot_model(new_model1, show_shapes=True)
```



Evaluate Model

```
# map an integer to a word
def word_for_id(integer, tokenizer):
    for word, index in tokenizer.word_index.items():
        if index == integer:
            return word
    return None


# generate a description for an image(changed approach)
def generate_desc(tokenizer, photo, max_length,finding):
    # seed the generation process
    in_text = 'startseq'
    photo=photo.reshape((1,photo.shape[0]))
    finding=finding.reshape((1,finding.shape[0]))
```

```
    finding=finding.reshape((1,finding.shape[0]))
    output1=model1a.predict([photo,finding], verbose=0)
    # iterate over the whole length of the sequence

    for i in range(max_length):
        # integer encode input sequence
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        # pad input
        sequence = pad_sequences([sequence], maxlen=max_length)
        output2 = model1b.predict(sequence, verbose=0)

        yhat = new_model1.predict([output1, output2], verbose=0)
        # convert probability to integer
        yhat = argmax(yhat)
        # map integer to word
        word = word_for_id(yhat, tokenizer)
        # stop if we cannot map the word
        if word is None:
            break
        # append as input for generating the next word
        in_text += ' ' + word
        # stop if we predict the end of the sequence
        if word == 'endseq':
            break
    return in_text


# remove start/end sequence tokens from a summary
def cleanup_summary(summary):
    # remove start of sequence token
    index = summary.find('startseq ')
    if index > -1:
        summary = summary[len('startseq '):]
    # remove end of sequence token
    index = summary.find(' endseq')
    if index > -1:
        summary = summary[:index]
    return summary


# evaluate the skill of the model
def evaluate_model(descriptions, photos, tokenizer, max_length,findings):
    actual, predicted = list(), list()
    # step over the whole set
    for key, desc in descriptions.items():
        # generate description
        yhat = generate_desc(tokenizer, photos[key][0], max_length,findings[key])
        # clean up prediction
        yhat = cleanup_summary(yhat)
        # store actual and predicted
        references = [cleanup_summary(desc).split()]
        actual.append(references)
        predicted.append(yhat.split())
    # calculate BLEU score
    print('BLEU-1: %f' % corpus_bleu(actual, predicted, weights=(1.0, 0, 0, 0)))
    print('BLEU-2: %f' % corpus_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0)))
```

```
    print('BLEU-2: %f' % corpus_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0)))
    print('BLEU-3: %f' % corpus_bleu(actual, predicted, weights=(0.3, 0.3, 0.3, 0)))
    print('BLEU-4: %f' % corpus_bleu(actual, predicted, weights=(0.25, 0.25, 0.25, 0.25)))


# evaluate model
evaluate_model(test_id_impressions, test_image_features, impressions_tokenizer, impressions_max_length,test_id
```

```
BLEU-1: 0.958754
BLEU-2: 0.950945
BLEU-3: 0.946297
BLEU-4: 0.927488
```

Conclusion: This model is sensible and ready to predict on unseen data points.

Step by Step Procedure to solve this case study

1) Apply data augmentation technique on images to increase image data at the same time preproc
extracted feature vectors.
2) Now we done with image features ,lets move to the text data.As we know the text data we got is
these files ,remove none values,preprocess it and store in dictionary format where key is a image i
because it will be easier to get required data field in next steps.
3) we build our model on [image data+findings] + impressions data

- First we split data into train,cv and test sets.
- we define necessary functions after this we create input sequences i.e. we convert text data
  maximum length of impression's text data for further processing.
- Training :Define Deep learning model for training and we save log files and give checkpoint p
- Model Evaluation: we define necessary functions after this we load our best model to predict
  5) Summarize results