Medical Report Generation Case Study

1. Business Problem

1.1 Description

- Generating radiology reports is time-consuming and requires extensive expertise in practice. successfully applied to image classification and image captioning tasks, radiology report ge understanding and linking complicated medical visual contents with accurate natural langua of open-access datasets that contain paired medical images and reports remain very limited
- Considering the demands of accurately interpreting medical images in large amounts within generation model can be helpful.The automatic generation of radiology reports given medic operationally and improve clinical patient care.Radiology reports contain information summa further diagnosis and follow-up recommendations.
- Common radiographic observations are enlarged cardiom, cardiomegaly, lung opacity, lung l atelectasis, pneumothorax, pleural effusion, pleural other, fracture, support devices, and no fi report generation is highly desired to alleviate the workload.

1.2 Problem Statement

For given Chest x-ray images we need to generate medical conclusive report so that doctor can procceed for fur

1.3 Source / useful links

Source: Indiana University Chest X-Ray Collection

- https://academictorrents.com/details/66450ba52ba3f83fbf82ef9c91f2bde0e845aba9
- https://openi.nlm.nih.gov/faq
- http://ling.snu.ac.kr/class/AI_Agent/deep_learning_for_nlp.pdf
- https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-trair
- **Research Papers:**
- https://arxiv.org/pdf/1907.09085.pdf
- https://papers.nips.cc/paper/7426-hybrid-retrieval-generation-reinforced-agent-for-medical-i
- http://openaccess.thecvf.com/content_cvpr_2017/papers/Wang_ChestX-ray8_Hospital-Scal
- http://www.cs.jhu.edu/~lelu/publication/TieNet_CVPR2018_spotlight.pdf
- https://www.aclweb.org/anthology/P19-1657.pdf
- https://zpascal.net/cvpr2016/Shin_Learning_to_Read_CVPR_2016_paper.pdf
- https://www.medrxiv.org/content/10.1101/19013342v1.full.pdf

1.4 Real World / Business Objectives and Constraints

- Here objective is to generate impressions(Conclusional Report) for given an X-Ray image.
- Incorrect reports could lead to incorrect treatment, delayed treatment, or no treatment at all, worse, and they may even die.
- No strict latency constraints.

2. Machine Learning problem

2.1 Data

2.1.1 Data Overview

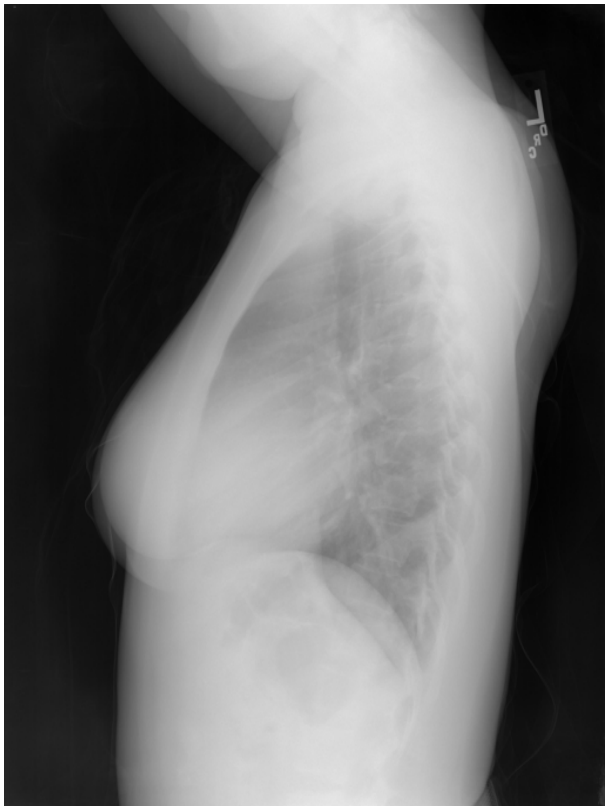
The data is in 2 folders :one contains image files and another contains report files

- All the report data is in XML files.
- Size of total report data: 30.1MB
- Number of xml files : 3955
- total size of image files: 1.28GB
- total number of images: 7470

Data Field Explaination

- COMPARISON:Information about the type of chest view.
- INDICATION:These are the changes observed in patients health
- FINDINGS:This provides overall conclusive information about chest x-ray
- IMPRESSION:This describes major substantial knowledge
- image id: X-ray image id associated with report

2.1.2 Example Data point



```
# 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
import xml.etree.ElementTree as ET
column_list=[]
file= 'C:/Users/Admin/Downloads/Medical_case_study/reports/1.xml'
tree = ET.parse(file)
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')=='COMPARISON':
                                comparison=name.text
                            elif name.get('Label')=='INDICATION':
                                indication=name.text
                            elif name.get('Label')=='FINDINGS':
                                findings=name.text
                            elif name.get('Label')=='IMPRESSION':
                                impression=name.text

        for p_image in root.findall('parentImage'):
            idd = p_image.get('id')
            print("\nCOMPARISON:",comparison)
            print("\nINDICATION:",indication)
            print("\nFINDINGS:",findings)
            print("\nIMPRESSION:",impression)
            print("\nid:",idd)
            break
```

COMPARISON: None.

INDICATION: Positive TB test

FINDINGS: The cardiac silhouette and mediastinum size are within normal limits. There is no pulmonary ed

IMPRESSION: Normal chest x-XXXX.

id: CXR1_1_IM-0001-3001

2.2 Mapping the real-world problem to a Machine Learning Problem

2.2.1 Type of Machine Learning Problem

- Its a Image Sequence Generation Problem.
- sequence generation problems include generation of a sequence given a single observation image so it will call as image Caption Generation.
- Given an image as input, generate a sequence of words that describes an image.

2.2.2 Performance metric

The evaluation metrics we use is BLEU scores.

- 1) BLEU : BLEU (Papineni et al., 2002) is one of the first metrics that have been in use for me has been initially proposed for machine translation, and defined as the geometric mean of n-penalty for short sentences.

```
!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
```

```
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
```

3. Exploratory Data Analysis

3.1 Data Loading

Lets see how many image files we have

```
directory = 'C:/Users/Admin/Downloads/Medical_case_study/images'
number_of_image_files= os.listdir(directory) # dir is your directory path
print("Total number of image files:",len(number_of_image_files))
```

Total number of image files: 7470

```
#print some file names
count=0
print("Some file names are:")
for name in listdir(directory):
    if count==5:
        break
    image_id = name.split('.')[0]
    print(image_id)
    count=count+1
```

Some file names are:
CXR1000_IM-0003-1001
CXR1000_IM-0003-2001
CXR1000_IM-0003-3001
CXR1001_IM-0004-1001
CXR1001_IM-0004-1002

Lets check the number of reports

```
directory = 'C:/Users/Admin/Downloads/Medical_case_study/reports'
number_of_report_files= os.listdir(directory)
print("Total number of report files:",len(number_of_report_files))
```

Total number of report files: 3955

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]
```

```
        # 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)
        #f='C:/Users/Admin/Downloads/Medical_case_study/reports/'+filename
        #print(filename)
        #count=0
        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

```
count=0
#finding none values in impression
```

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

```
    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')


# after cleaning impression data
count=0
for k,v in cleaned_id_impressions.items():
    if count!=1:
        print(k,"\t",v)
        count+=1
```

```
CXR1_1_IM-0001-3001	 startseq normal chest  endseq
```

```
# after cleaning findings data
count=0
for k,v in cleaned_id_findings.items():
    if count!=1:
        print(k,"\t",v,"\n")
        count+=1
```

```
CXR1_1_IM-0001-3001	 the cardiac silhouette and mediastinum size are within normal limits there is no p
```

Model 1: image_vector + partial impression = predicts next

3) Split image identifiers into train,cv,test

```
keys=list(cleaned_id_impressions.keys())
random.shuffle(keys)
train_id,cv_id,test_id=keys[0:21754],keys[21754:22004],keys[22004:]
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 :	21754
cv size     :	250
test size  :	250
```

Define Necessary Functions

```
def create_sequences(tokenizer, max_length, descriptions, image_features,vocab_size):
    """this method is to create input sequences"""
    X1, X2, y = 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])
            X2.append(in_seq)
            y.append(out_seq)
    return array(X1), array(X2), array(y)

# 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
```

4.1 Prepare train data

```
train_id_impressions=load_respective_set(cleaned_id_impressions, train_id)
print('Total train Descriptions          : ',len(train_id_impressions))

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

# prepare tokenizer
tokenizer = create_tokenizer(train_id_impressions)

vocab_size = len(tokenizer.word_index) + 1
print('\nVocabulary Size                   : ', vocab_size)

# pad to fixed length
max_length = max(len(s.split()) for s in list(train_id_impressions.values()))
print('\nDescription maximum Length  : ',max_length)
```

```
Total train Descriptions          :  18883

Total train images                :  18883

Vocabulary Size                   :  1208

Description maximum Length  :  114
```

```
#https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/
# Load Glove vectors
glove_words = pickle.load(open('/content/drive/My Drive/Xception/glove_vectors', 'rb'))#
embedding_matrix = np.zeros((vocab_size, 300))

for word, i in tokenizer.word_index.items():
    embedding_vector = glove_words.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector


# prepare sequences
X1train, X2train, ytrain = create_sequences(tokenizer, max_length,train_id_impressions,train_image_features,vocal
```

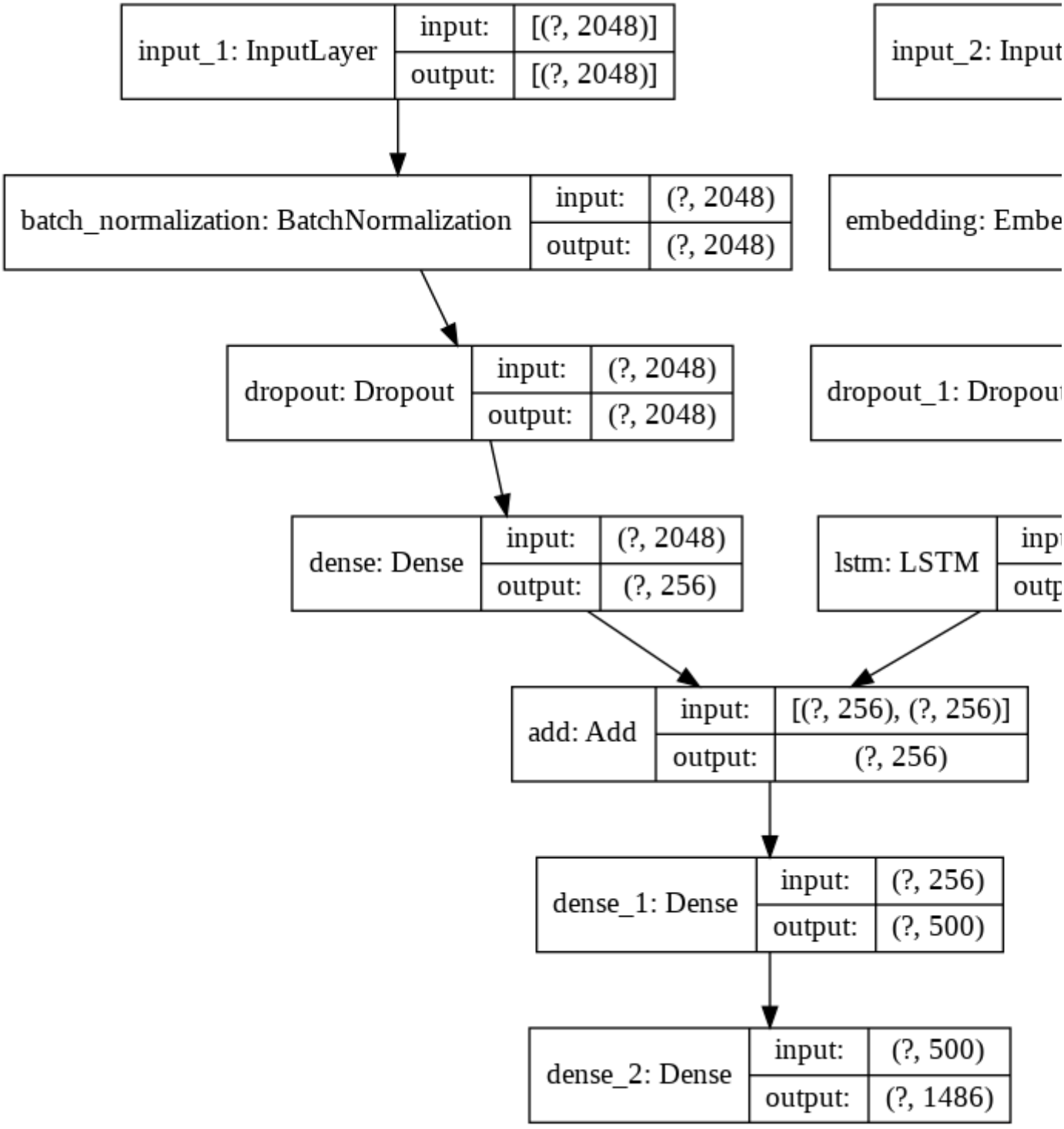
X1train, X2train, ytrain = create_sequences(tokenizer, max_length,train_id_impressions,train_image_features,vocal

4.2 Prepare cv data

```
cv_id_impressions=load_respective_set(cleaned_id_impressions, cv_id)
print('\nTotal cv Descriptions : ',len(cv_id_impressions))

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

# prepare sequences
X1cv, X2cv, ycv = create_sequences(tokenizer, max_length,cv_id_impressions,cv_image_features,vocab_size)
```

```
Total cv Descriptions :  250

Total cv images       :  250
```

Build Deep Learning Model 1.1 : with Batch Normalization + Dropout2(0.3)




```
def define_model(vocab_size, max_length,embedding_matrix):
    """this method is used to define model"""
    # feature extractor model
    inputs1 = tf.keras.layers.Input(shape=(2048,))
    fe = tf.keras.layers.BatchNormalization()(inputs1)
    fe1 = tf.keras.layers.Dropout(0.4)(fe)
    fe2 = tf.keras.layers.Dense(256, activation='relu')(fe1)

    # sequence model
    inputs2 = tf.keras.layers.Input(shape=(max_length,))
    se1 = tf.keras.layers.Embedding(vocab_size,300,weights=[embedding_matrix],trainable=False ,mask_zero=True
    se2 = tf.keras.layers.Dropout(0.3)(se1)
    se3 = tf.keras.layers.LSTM(256)(se2)
```

```
    # decoder model
    decoder1 = tf.keras.layers.add([fe2, se3])
    decoder2 = tf.keras.layers.Dense(500, activation='relu')(decoder1)
    outputs = tf.keras.layers.Dense(vocab_size, activation='softmax')(decoder2)
    # tie it together [image, seq] [word]
    model = tf.keras.models.Model(inputs=[inputs1, inputs2], 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/Model1/checkmergearchitecture.png', show_sha
    return model


model = define_model(vocab_size, max_length,embedding_matrix)
```

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 2048)]	0	
input_2 (InputLayer)	[(None, 125)]	0	
batch_normalization (BatchNorma	(None, 2048)	8192	input_1[0][0]
embedding (Embedding)	(None, 125, 300)	445800	input_2[0][0]
dropout (Dropout)	(None, 2048)	0	batch_normalization[0][0]
dropout_1 (Dropout)	(None, 125, 300)	0	embedding[0][0]
dense (Dense)	(None, 256)	524544	dropout[0][0]
lstm (LSTM)	(None, 256)	570368	dropout_1[0][0]
add (Add)	(None, 256)	0	dense[0][0] lstm[0][0]
dense_1 (Dense)	(None, 500)	128500	add[0][0]
dense_2 (Dense)	(None, 1486)	744486	dense_1[0][0]

Total params: 2,421,890
Trainable params: 1,971,994
Non-trainable params: 449,896

```
xtrain=[X1train, X2train]
xcv=[X1cv, X2cv]

# define checkpoint callback
checkpoint = tf.keras.callbacks.ModelCheckpoint('/content/drive/My Drive/Xception/Model1/checkmergecheckp

log_dir="/content/drive/My Drive/Xception/Model1/checkmergetensorboardlogs1/logs/fit/" + datetime.now().st
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=30, verbose=2,callbacks=[tensorboard_callback,checkpoint], va
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
Epoch 1/30

Epoch 00001: val_loss improved from inf to 2.80091, saving model to /content/drive/My Drive/Xception/M
463/463 - 214s - loss: 3.4785 - val_loss: 2.8009
Epoch 2/30

Epoch 00002: val_loss improved from 2.80091 to 2.01070, saving model to /content/drive/My Drive/Xcepti
463/463 - 211s - loss: 2.1981 - val_loss: 2.0107
Epoch 3/30

Epoch 00003: val_loss improved from 2.01070 to 1.51630, saving model to /content/drive/My Drive/Xcepti
463/463 - 211s - loss: 1.6290 - val_loss: 1.5163
Epoch 4/30

Epoch 00004: val_loss improved from 1.51630 to 1.14725, saving model to /content/drive/My Drive/Xcepti
463/463 - 211s - loss: 1.2549 - val_loss: 1.1472
Epoch 5/30

Epoch 00005: val_loss improved from 1.14725 to 0.90980, saving model to /content/drive/My Drive/Xcepti
463/463 - 211s - loss: 1.0093 - val_loss: 0.9098
Epoch 6/30

Epoch 00006: val_loss improved from 0.90980 to 0.76469, saving model to /content/drive/My Drive/Xcepti
463/463 - 209s - loss: 0.8516 - val_loss: 0.7647
Epoch 7/30

Epoch 00007: val_loss improved from 0.76469 to 0.68031, saving model to /content/drive/My Drive/Xcepti
463/463 - 212s - loss: 0.7603 - val_loss: 0.6803
Epoch 8/30

Epoch 00008: val_loss improved from 0.68031 to 0.63365, saving model to /content/drive/My Drive/Xcepti
463/463 - 212s - loss: 0.7039 - val_loss: 0.6337
Epoch 9/30

Epoch 00009: val_loss improved from 0.63365 to 0.60716, saving model to /content/drive/My Drive/Xcepti
463/463 - 212s - loss: 0.6674 - val_loss: 0.6072
Epoch 10/30

Epoch 00010: val_loss improved from 0.60716 to 0.58959, saving model to /content/drive/My Drive/Xcepti
463/463 - 214s - loss: 0.6415 - val_loss: 0.5896
Epoch 11/30

Epoch 00011: val_loss improved from 0.58959 to 0.56974, saving model to /content/drive/My Drive/Xcepti
463/463 - 212s - loss: 0.6197 - val_loss: 0.5697
Epoch 12/30

Epoch 00012: val_loss improved from 0.56974 to 0.56487, saving model to /content/drive/My Drive/Xcepti
463/463 - 213s - loss: 0.6068 - val_loss: 0.5649
Epoch 13/30

Epoch 00013: val_loss improved from 0.56487 to 0.54530, saving model to /content/drive/My Drive/Xcepti
463/463 - 213s - loss: 0.5937 - val_loss: 0.5453
Epoch 14/30

Epoch 00014: val_loss did not improve from 0.54530
463/463 - 212s - loss: 0.5855 - val_loss: 0.5467
Epoch 15/30
```

Epoch 00015: val_loss improved from 0.54530 to 0.54446, saving model to /content/drive/My Drive/Xcepti
463/463 - 209s - loss: 0.5758 - val_loss: 0.5445
Epoch 16/30

Epoch 00016: val_loss improved from 0.54446 to 0.54267, saving model to /content/drive/My Drive/Xcepti
463/463 - 211s - loss: 0.5683 - val_loss: 0.5427
Epoch 17/30

Epoch 00017: val_loss did not improve from 0.54267
463/463 - 211s - loss: 0.5626 - val_loss: 0.5436
Epoch 18/30

Epoch 00018: val_loss improved from 0.54267 to 0.53624, saving model to /content/drive/My Drive/Xcepti
463/463 - 210s - loss: 0.5569 - val_loss: 0.5362
Epoch 19/30

Epoch 00019: val_loss improved from 0.53624 to 0.52750, saving model to /content/drive/My Drive/Xcepti
463/463 - 212s - loss: 0.5503 - val_loss: 0.5275
Epoch 20/30

Epoch 00020: val_loss did not improve from 0.52750
463/463 - 210s - loss: 0.5445 - val_loss: 0.5298
Epoch 21/30

Epoch 00021: val_loss did not improve from 0.52750
463/463 - 211s - loss: 0.5410 - val_loss: 0.5429
Epoch 22/30

Epoch 00022: val_loss did not improve from 0.52750
463/463 - 212s - loss: 0.5344 - val_loss: 0.5277
Epoch 23/30

Epoch 00023: val_loss improved from 0.52750 to 0.52390, saving model to /content/drive/My Drive/Xcepti
463/463 - 212s - loss: 0.5314 - val_loss: 0.5239
Epoch 24/30

Epoch 00024: val_loss did not improve from 0.52390
463/463 - 211s - loss: 0.5288 - val_loss: 0.5256
Epoch 25/30

Epoch 00025: val_loss improved from 0.52390 to 0.51963, saving model to /content/drive/My Drive/Xcepti
463/463 - 215s - loss: 0.5266 - val_loss: 0.5196
Epoch 26/30

Epoch 00026: val_loss did not improve from 0.51963
463/463 - 212s - loss: 0.5213 - val_loss: 0.5255
Epoch 27/30

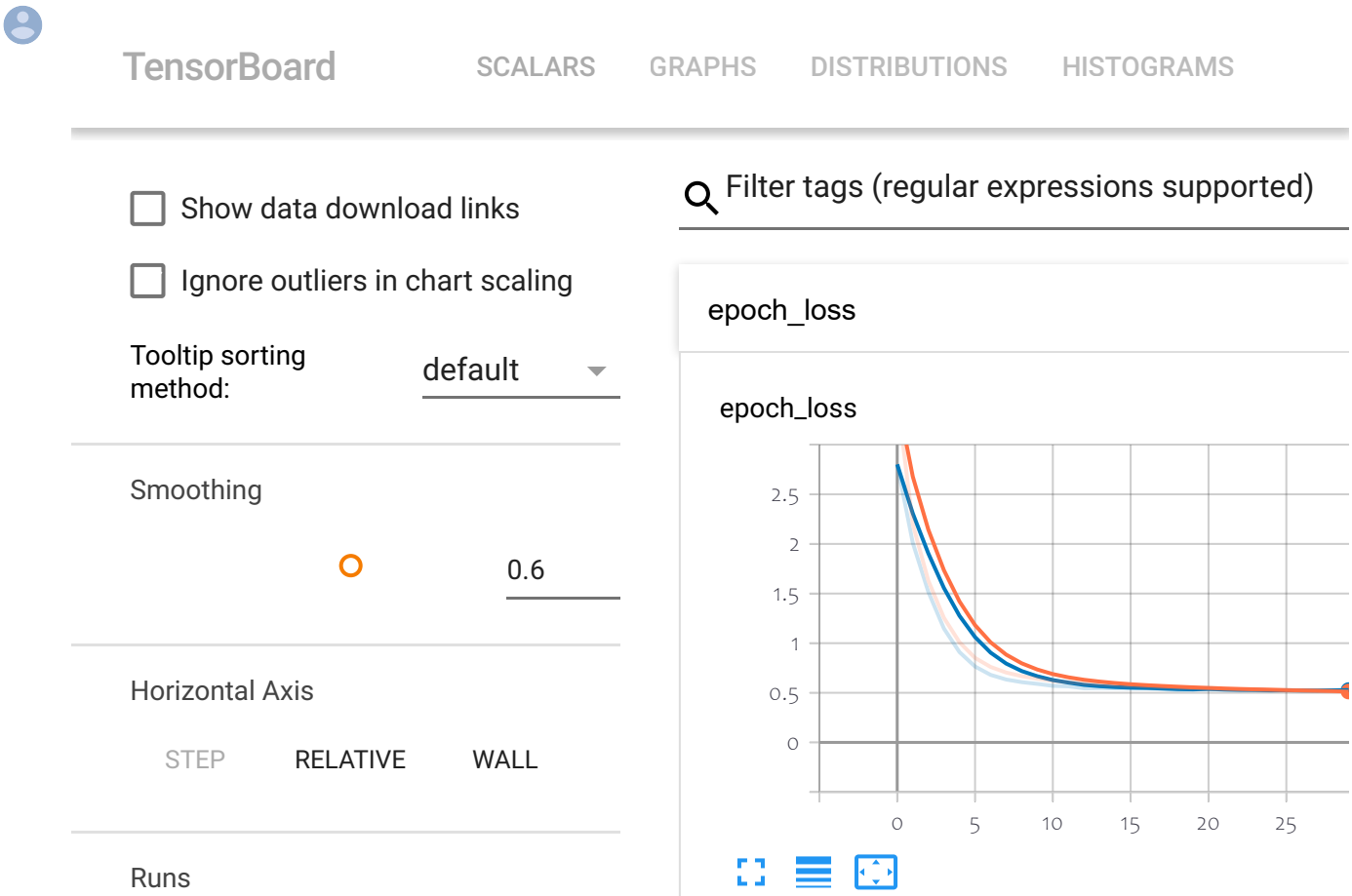
Epoch 00027: val_loss improved from 0.51963 to 0.51956, saving model to /content/drive/My Drive/Xcepti
463/463 - 212s - loss: 0.5186 - val_loss: 0.5196
Epoch 28/30

Epoch 00028: val_loss did not improve from 0.51956
463/463 - 212s - loss: 0.5162 - val_loss: 0.5216
Epoch 29/30

Epoch 00029: val_ loss did not improve from 0.51956

```
463/463 - 212s - loss: 0.5121 - val_loss: 0.5280
Epoch 30/30

Epoch 00030: val_loss did not improve from 0.51956
463/463 - 213s - loss: 0.5054 - val_loss: 0.5303
```

```
%tensorboard --logdir='/content/drive/My Drive/Xception/Model1/checkmergetensorboardlogs1/logs/fit'
```



4.4 Evaluate Model:1.1

```
# 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
def generate_desc(model, tokenizer, photo, max_length):
    # seed the generation process
    in_text = 'startseq'
    # 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)
        # predict next word
        #print(sequence.shape)
        #print(photo.shape)
        yhat = model.predict([photo,sequence], 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(model, descriptions, photos, tokenizer, max_length):
```

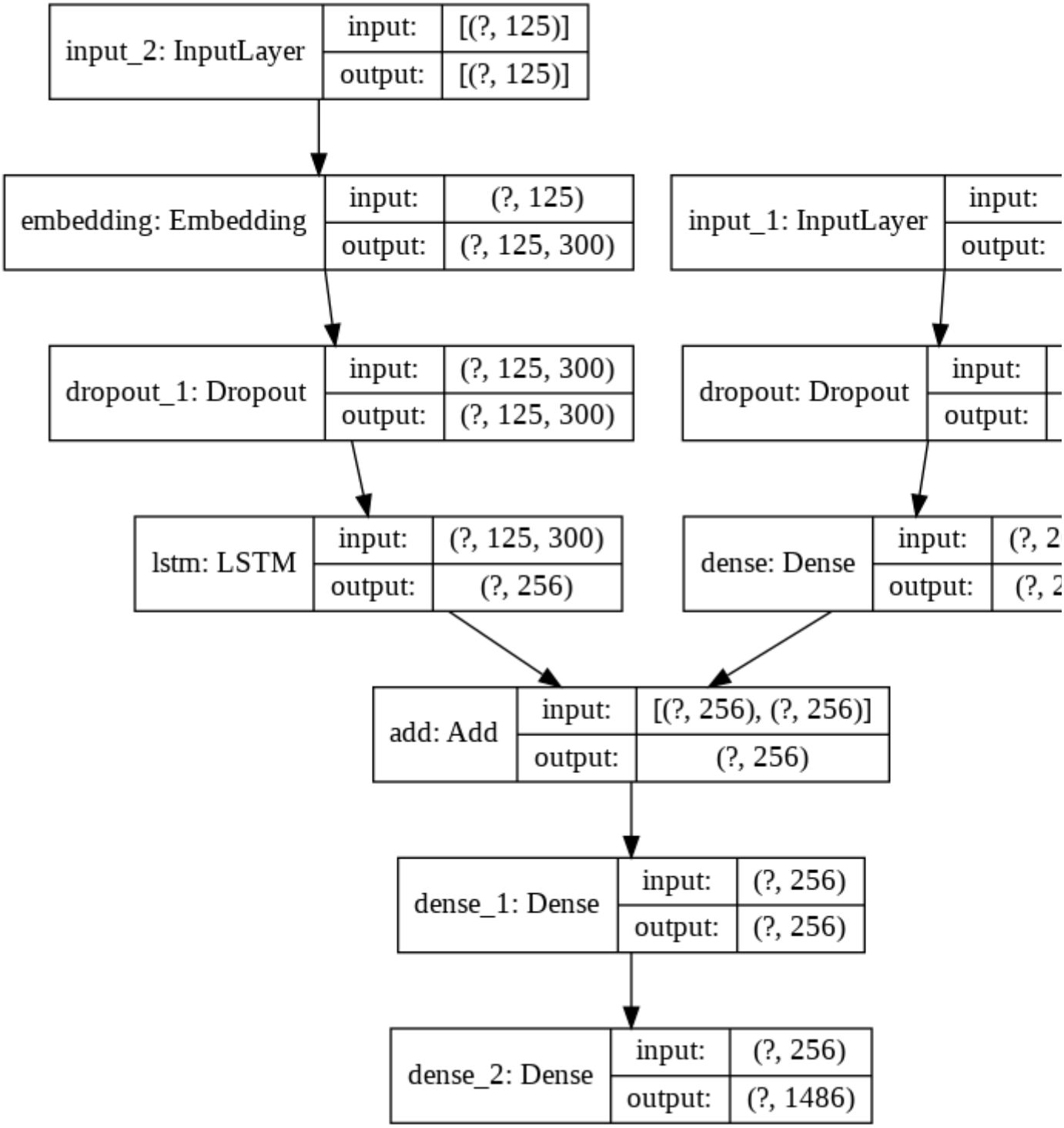
```
    actual, predicted = list(), list()
    # step over the whole set
    for key, desc in descriptions.items():
        # generate description
        yhat = generate_desc(model, tokenizer, photos[key], max_length)
        # clean up prediction
        yhat = cleanup_summary(yhat)
        # store 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-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)))
```

Prepare test data and Evaluate

```
test_id_impressions=load_respective_set(cleaned_id_impressions, test_id)
print('\nTotal test Descriptions : ',len(test_id_impressions))

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

# load the model
model2=tf.keras.models.load_model('/content/drive/My Drive/Xception/Model1/checkmergecheckpoint1.hdf5')
# evaluate model
evaluate_model(model2, test_id_impressions, test_image_features, tokenizer, max_length)
```

```
Total test Descriptions :  250

Total test images       :  250
BLEU-1: 0.110075
BLEU-2: 0.093749
BLEU-3: 0.087456
BLEU-4: 0.061429
```

Model 1.2 : without Batch Normalization + Dropout 1(0.5)+D




```
# merge  model
def define_model(vocab_size, max_length,embedding_matrix):
    """this method is used to define model"""
    # feature extractor model
    inputs1 = tf.keras.layers.Input(shape=(2048,))
    fe1 = tf.keras.layers.Dropout(0.5)(inputs1)
    fe2 = tf.keras.layers.Dense(256, activation='relu')(fe1)

    # sequence model
    inputs2 = tf.keras.layers.Input(shape=(max_length,))
    se1 = tf.keras.layers.Embedding(vocab_size,300,weights=[embedding_matrix],trainable=False ,mask_zero=True
    se2 = tf.keras.layers.Dropout(0.5)(se1)
    se3 = tf.keras.layers.LSTM(256)(se2)
```

```
    # decoder model
    decoder1 = tf.keras.layers.add([fe2, se3])
    decoder2 = tf.keras.layers.Dense(256, activation='relu')(decoder1)
    outputs = tf.keras.layers.Dense(vocab_size, activation='softmax')(decoder2)
    # tie it together [image, seq] [word]
    model = tf.keras.models.Model(inputs=[inputs1, inputs2], 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/Model1/mergearchitecture.png', show_shapes=
    return model


model = define_model(vocab_size, max_length,embedding_matrix)


xtrain=[X1train, X2train]
xcv=[X1cv, X2cv]

# define checkpoint callback
checkpoint = tf.keras.callbacks.ModelCheckpoint('/content/drive/My Drive/Xception/Model1/mergecheckpoint1.

log_dir="/content/drive/My Drive/Xception/Model1/mergetensorboardlogs1/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=30, verbose=2,callbacks=[tensorboard_callback,checkpoint], va
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
Epoch 1/30

Epoch 00001: val_loss improved from inf to 2.38937, saving model to /content/drive/My Drive/Xception/M
463/463 - 204s - loss: 2.6133 - val_loss: 2.3894
Epoch 2/30

Epoch 00002: val_loss improved from 2.38937 to 1.96590, saving model to /content/drive/My Drive/Xcepti
463/463 - 203s - loss: 2.1562 - val_loss: 1.9659
Epoch 3/30

Epoch 00003: val_loss improved from 1.96590 to 1.65229, saving model to /content/drive/My Drive/Xcepti
463/463 - 202s - loss: 1.8267 - val_loss: 1.6523
Epoch 4/30

Epoch 00004: val_loss improved from 1.65229 to 1.40402, saving model to /content/drive/My Drive/Xcepti
463/463 - 203s - loss: 1.5647 - val_loss: 1.4040
Epoch 5/30

Epoch 00005: val_loss improved from 1.40402 to 1.17387, saving model to /content/drive/My Drive/Xcepti
463/463 - 202s - loss: 1.3603 - val_loss: 1.1739
Epoch 6/30

Epoch 00006: val_loss improved from 1.17387 to 1.02322, saving model to /content/drive/My Drive/Xcepti
463/463 - 201s - loss: 1.1980 - val_loss: 1.0232
Epoch 7/30

Epoch 00007: val_loss improved from 1.02322 to 0.88878, saving model to /content/drive/My Drive/Xcepti
463/463 - 202s - loss: 1.0754 - val_loss: 0.8888
Epoch 8/30

Epoch 00008: val_loss improved from 0.88878 to 0.81076, saving model to /content/drive/My Drive/Xcepti
463/463 - 201s - loss: 0.9834 - val_loss: 0.8108
Epoch 9/30

Epoch 00009: val_loss improved from 0.81076 to 0.76043, saving model to /content/drive/My Drive/Xcepti
463/463 - 201s - loss: 0.9124 - val_loss: 0.7604
Epoch 10/30

Epoch 00010: val_loss improved from 0.76043 to 0.71084, saving model to /content/drive/My Drive/Xcepti
463/463 - 201s - loss: 0.8613 - val_loss: 0.7108
Epoch 11/30

Epoch 00011: val_loss improved from 0.71084 to 0.66950, saving model to /content/drive/My Drive/Xcepti
463/463 - 202s - loss: 0.8175 - val_loss: 0.6695
Epoch 12/30

Epoch 00012: val_loss improved from 0.66950 to 0.65561, saving model to /content/drive/My Drive/Xcepti
463/463 - 201s - loss: 0.7855 - val_loss: 0.6556
Epoch 13/30

Epoch 00013: val_loss improved from 0.65561 to 0.63177, saving model to /content/drive/My Drive/Xcepti
463/463 - 201s - loss: 0.7622 - val_loss: 0.6318
Epoch 14/30

Epoch 00014: val_loss improved from 0.63177 to 0.62864, saving model to /content/drive/My Drive/Xcepti
463/463 - 200s - loss: 0.7418 - val_loss: 0.6286
Epoch 15/30
```

Epoch 00015: val_loss improved from 0.62864 to 0.60357, saving model to /content/drive/My Drive/Xcepti
463/463 - 201s - loss: 0.7254 - val_loss: 0.6036
Epoch 16/30

Epoch 00016: val_loss did not improve from 0.60357
463/463 - 200s - loss: 0.7124 - val_loss: 0.6082
Epoch 17/30

Epoch 00017: val_loss improved from 0.60357 to 0.59079, saving model to /content/drive/My Drive/Xcepti
463/463 - 200s - loss: 0.7006 - val_loss: 0.5908
Epoch 18/30

Epoch 00018: val_loss did not improve from 0.59079
463/463 - 200s - loss: 0.6917 - val_loss: 0.5951
Epoch 19/30

Epoch 00019: val_loss improved from 0.59079 to 0.58644, saving model to /content/drive/My Drive/Xcepti
463/463 - 201s - loss: 0.6852 - val_loss: 0.5864
Epoch 20/30

Epoch 00020: val_loss improved from 0.58644 to 0.58042, saving model to /content/drive/My Drive/Xcepti
463/463 - 202s - loss: 0.6755 - val_loss: 0.5804
Epoch 21/30

Epoch 00021: val_loss improved from 0.58042 to 0.56641, saving model to /content/drive/My Drive/Xcepti
463/463 - 200s - loss: 0.6730 - val_loss: 0.5664
Epoch 22/30

Epoch 00022: val_loss did not improve from 0.56641
463/463 - 200s - loss: 0.6656 - val_loss: 0.5726
Epoch 23/30

Epoch 00023: val_loss did not improve from 0.56641
463/463 - 200s - loss: 0.6583 - val_loss: 0.5745
Epoch 24/30

Epoch 00024: val_loss did not improve from 0.56641
463/463 - 200s - loss: 0.6559 - val_loss: 0.5712
Epoch 25/30

Epoch 00025: val_loss improved from 0.56641 to 0.56433, saving model to /content/drive/My Drive/Xcepti
463/463 - 201s - loss: 0.6504 - val_loss: 0.5643
Epoch 26/30

Epoch 00026: val_loss did not improve from 0.56433
463/463 - 200s - loss: 0.6486 - val_loss: 0.5687
Epoch 27/30

Epoch 00027: val_loss did not improve from 0.56433
463/463 - 200s - loss: 0.6415 - val_loss: 0.5682
Epoch 28/30

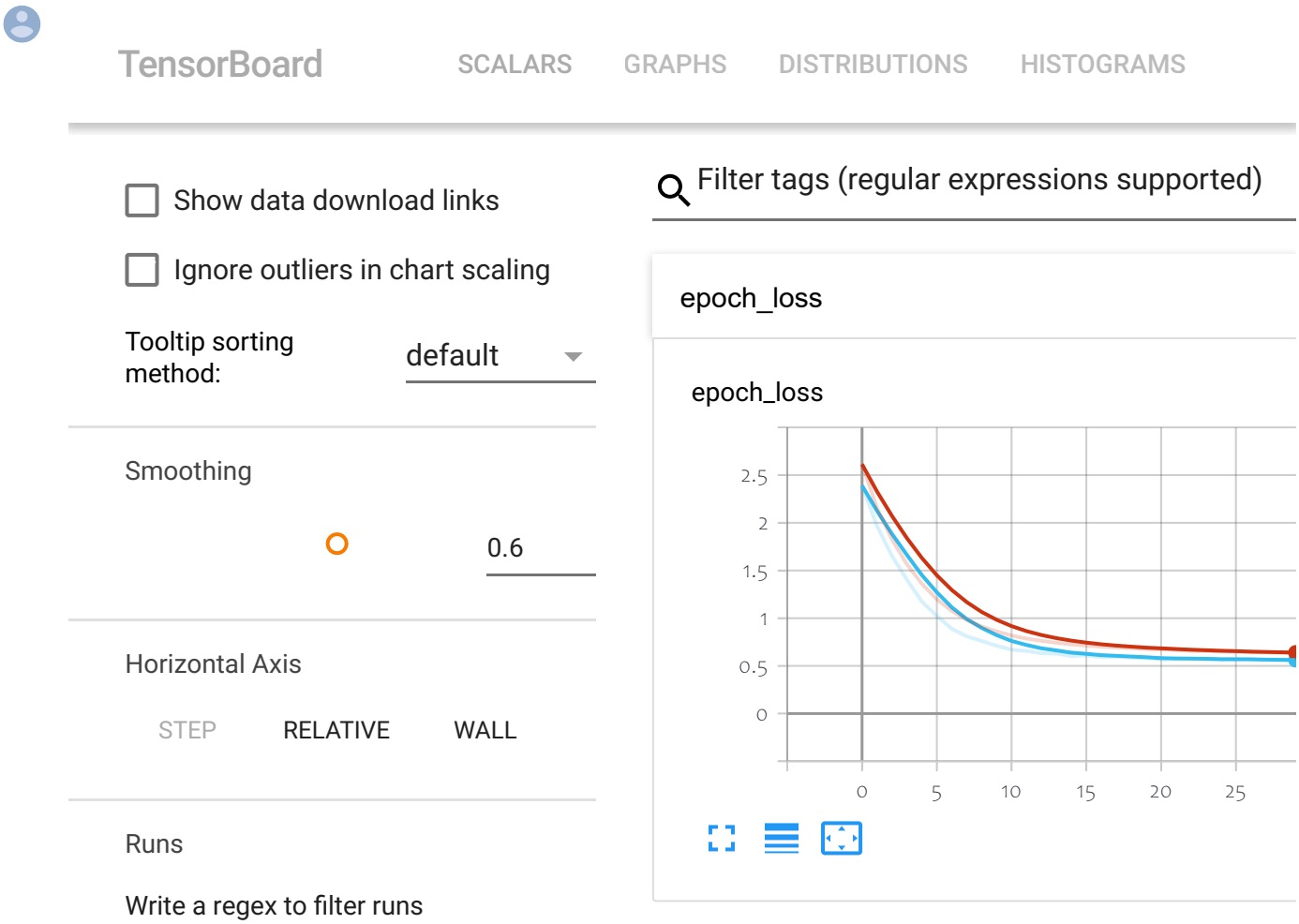
Epoch 00028: val_loss improved from 0.56433 to 0.56037, saving model to /content/drive/My Drive/Xcepti
463/463 - 200s - loss: 0.6406 - val_loss: 0.5604
Epoch 29/30

Epoch 00029: val_loss improved from 0.56037 to 0.56019, saving model to /content/drive/My Drive/Xcepti

463/463 - 200s - loss: 0.6375 - val_loss: 0.5602
Epoch 30/30

Epoch 00030: val_loss improved from 0.56019 to 0.55950, saving model to /content/drive/My Drive/Xcepti
463/463 - 201s - loss: 0.6340 - val_loss: 0.5595

```
%tensorboard --logdir='/content/drive/My Drive/Xception/Model1/mergetensorboardlogs1/logs/fit'
```



4.4 Evaluate Model 1.2

```
print('\nTotal test Descriptions : ',len(test_id_impressions))

print('\nTotal test images       : ' ,len(test_image_features))

# load the model
model2=tf.keras.models.load_model('/content/drive/My Drive/Xception/Model1/mergecheckpoint1.hdf5')
# evaluate model
evaluate_model(model2, test_id_impressions, test_image_features, tokenizer, max_length)
```

```
Total test Descriptions :  250

Total test images       :  250
BLEU-1: 0.130980
BLEU-2: 0.105110
BLEU-3: 0.092586
BLEU-4: 0.112439
```

Model 2 : (model 1 + We use findings also)

Train,cv,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
```

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)

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

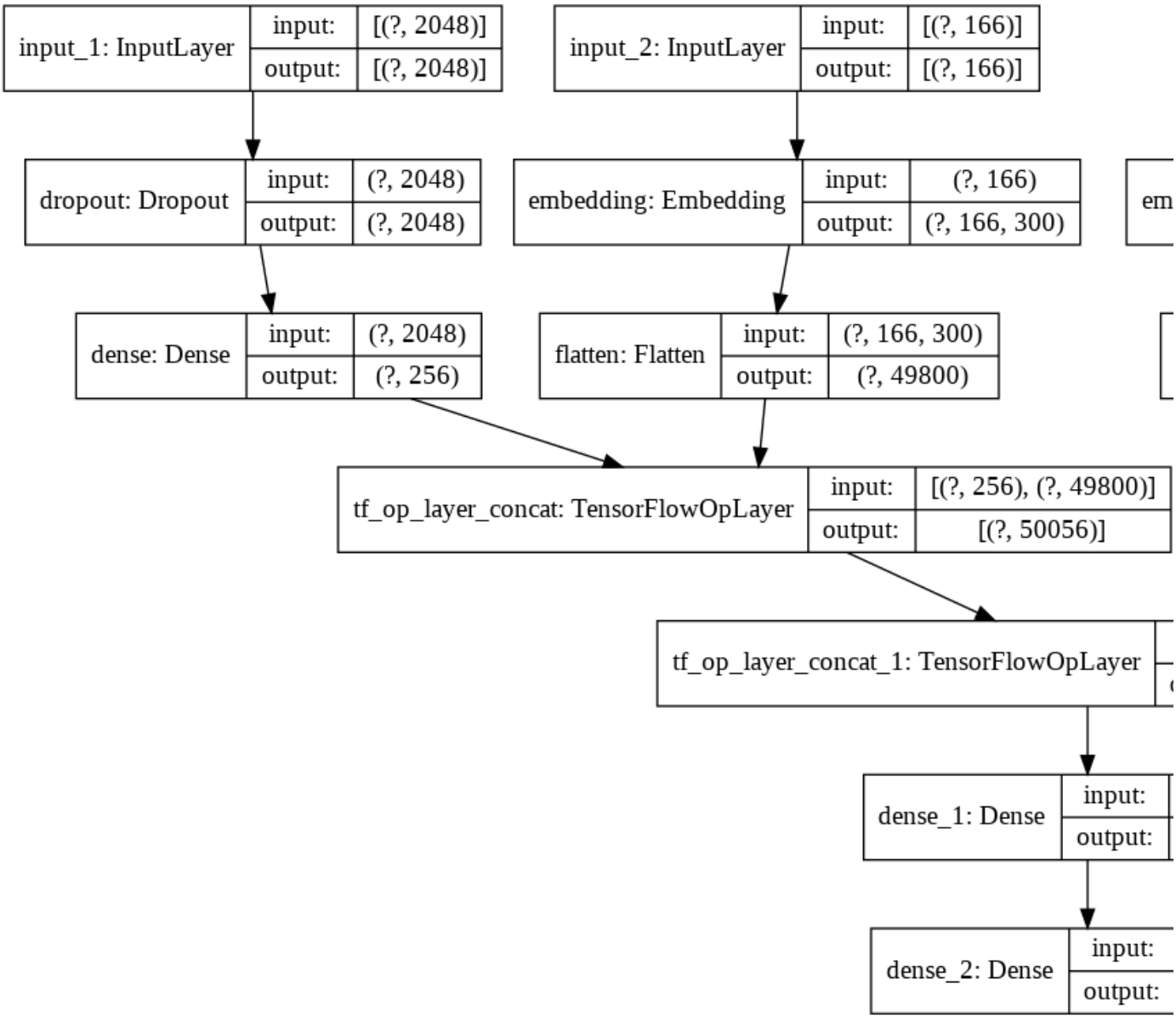
```
cv_image_features = load_image_features(image_extracted_features,cv_id)
print('\nTotal cv images      : ' ,len(cv_image_features))

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

Deep Learning Model 2.1: Without Batch Normalization+Dro




```
# 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
    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/mergearchitecture.png', show_shapes=
    return model


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

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 2048)]       0
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 166)]        0
__________________________________________________________________________________________________
input_3 (InputLayer)            [(None, 114)]        0
__________________________________________________________________________________________________
dropout (Dropout)               (None, 2048)         0           input_1[0][0]
__________________________________________________________________________________________________
embedding (Embedding)           (None, 166, 300)     468900      input_2[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 114, 300)     362400      input_3[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 256)          524544      dropout[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 49800)        0           embedding[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 114, 300)     0           embedding_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat (TensorFlowO [(None, 50056)]      0           dense[0][0]
                                                                 flatten[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     (None, 256)          570368      dropout_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_1 (TensorFlo [(None, 50312)]      0           tf_op_layer_concat[0][0]
                                                                 lstm[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 500)          25156500    tf_op_layer_concat_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 1208)         605208      dense_1[0][0]
==================================================================================================
Total params: 27,687,920
Trainable params: 26,856,620
Non-trainable params: 831,300
__________________________________________________________________________________________________
```

```
# define the model
model = define_model(findings_max_length,findings_vocab_size,impressions_max_length,impressions_vocab_size
# define the model
xtrain=[X1train, X2train , X3train]
xcv=[X1cv, X2cv, X3cv]
# define checkpoint callback
checkpoint = tf.keras.callbacks.ModelCheckpoint('/content/drive/My Drive/Xception/Model2/mergemodelcheck.

log_dir="/content/drive/My Drive/Xception/Model2/mergetensorboardlogs/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=30, verbose=2,callbacks=[tensorboard_callback,checkpoint], va
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
Epoch 1/30

Epoch 00001: val_loss improved from inf to 1.43328, saving model to /content/drive/My Drive/Xception/M
314/314 - 156s - loss: 2.6899 - val_loss: 1.4333
Epoch 2/30

Epoch 00002: val_loss improved from 1.43328 to 0.80556, saving model to /content/drive/My Drive/Xcepti
314/314 - 156s - loss: 1.2346 - val_loss: 0.8056
Epoch 3/30

Epoch 00003: val_loss improved from 0.80556 to 0.50078, saving model to /content/drive/My Drive/Xcepti
314/314 - 155s - loss: 0.7730 - val_loss: 0.5008
Epoch 4/30

Epoch 00004: val_loss improved from 0.50078 to 0.34965, saving model to /content/drive/My Drive/Xcepti
314/314 - 155s - loss: 0.5154 - val_loss: 0.3496
Epoch 5/30

Epoch 00005: val_loss improved from 0.34965 to 0.22912, saving model to /content/drive/My Drive/Xcepti
314/314 - 154s - loss: 0.3672 - val_loss: 0.2291
Epoch 6/30

Epoch 00006: val_loss improved from 0.22912 to 0.16170, saving model to /content/drive/My Drive/Xcepti
314/314 - 156s - loss: 0.2738 - val_loss: 0.1617
Epoch 7/30

Epoch 00007: val_loss improved from 0.16170 to 0.12171, saving model to /content/drive/My Drive/Xcepti
314/314 - 154s - loss: 0.2150 - val_loss: 0.1217
Epoch 8/30

Epoch 00008: val_loss improved from 0.12171 to 0.10196, saving model to /content/drive/My Drive/Xcepti
314/314 - 155s - loss: 0.1735 - val_loss: 0.1020
Epoch 9/30

Epoch 00009: val_loss improved from 0.10196 to 0.07042, saving model to /content/drive/My Drive/Xcepti
314/314 - 153s - loss: 0.1523 - val_loss: 0.0704
Epoch 10/30

Epoch 00010: val_loss did not improve from 0.07042
314/314 - 152s - loss: 0.1361 - val_loss: 0.0765
Epoch 11/30

Epoch 00011: val_loss did not improve from 0.07042
314/314 - 151s - loss: 0.1210 - val_loss: 0.1470
Epoch 12/30

Epoch 00012: val_loss did not improve from 0.07042
314/314 - 151s - loss: 0.1147 - val_loss: 0.1001
Epoch 13/30

Epoch 00013: val_loss improved from 0.07042 to 0.05188, saving model to /content/drive/My Drive/Xcepti
314/314 - 153s - loss: 0.1162 - val_loss: 0.0519
Epoch 14/30

Epoch 00014: val_loss did not improve from 0.05188
314/314 - 152s - loss: 0.1063 - val_loss: 0.0667
Epoch 15/30
```

Epoch 00015: val_loss improved from 0.05188 to 0.04682, saving model to /content/drive/My Drive/Xcepti
314/314 - 154s - loss: 0.1105 - val_loss: 0.0468
Epoch 16/30

Epoch 00016: val_loss did not improve from 0.04682
314/314 - 153s - loss: 0.1031 - val_loss: 0.0731
Epoch 17/30

Epoch 00017: val_loss improved from 0.04682 to 0.04316, saving model to /content/drive/My Drive/Xcepti
314/314 - 153s - loss: 0.0924 - val_loss: 0.0432
Epoch 18/30

Epoch 00018: val_loss did not improve from 0.04316
314/314 - 152s - loss: 0.0899 - val_loss: 0.0453
Epoch 19/30

Epoch 00019: val_loss improved from 0.04316 to 0.03250, saving model to /content/drive/My Drive/Xcepti
314/314 - 154s - loss: 0.0872 - val_loss: 0.0325
Epoch 20/30

Epoch 00020: val_loss improved from 0.03250 to 0.03013, saving model to /content/drive/My Drive/Xcepti
314/314 - 153s - loss: 0.0900 - val_loss: 0.0301
Epoch 21/30

Epoch 00021: val_loss did not improve from 0.03013
314/314 - 152s - loss: 0.0838 - val_loss: 0.0339
Epoch 22/30

Epoch 00022: val_loss did not improve from 0.03013
314/314 - 151s - loss: 0.0816 - val_loss: 0.0381
Epoch 23/30

Epoch 00023: val_loss improved from 0.03013 to 0.03001, saving model to /content/drive/My Drive/Xcepti
314/314 - 152s - loss: 0.0813 - val_loss: 0.0300
Epoch 24/30

Epoch 00024: val_loss improved from 0.03001 to 0.02231, saving model to /content/drive/My Drive/Xcepti
314/314 - 153s - loss: 0.0743 - val_loss: 0.0223
Epoch 25/30

Epoch 00025: val_loss did not improve from 0.02231
314/314 - 153s - loss: 0.0703 - val_loss: 0.0266
Epoch 26/30

Epoch 00026: val_loss did not improve from 0.02231
314/314 - 151s - loss: 0.0743 - val_loss: 0.0299
Epoch 27/30

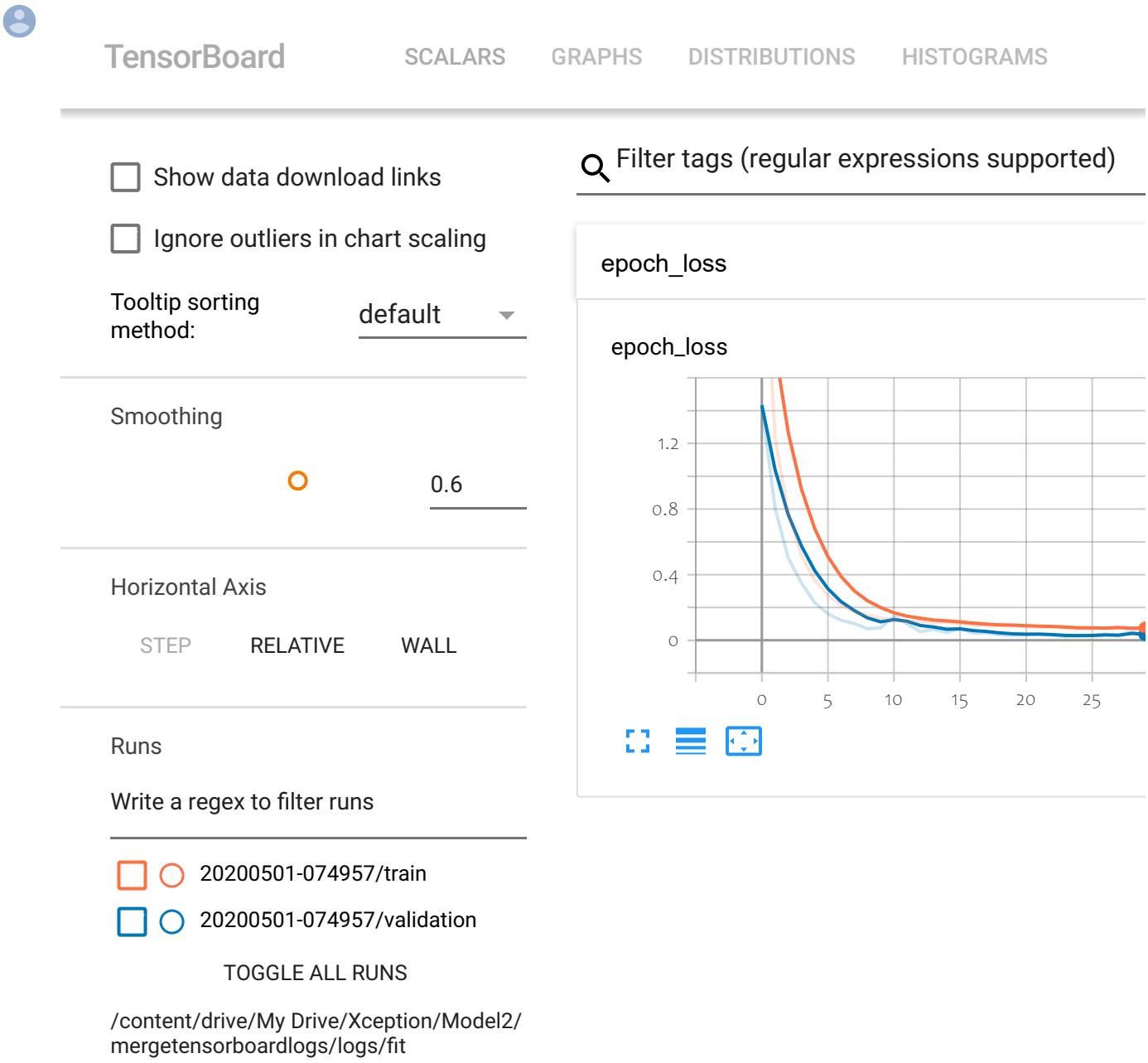
Epoch 00027: val_loss did not improve from 0.02231
314/314 - 151s - loss: 0.0729 - val_loss: 0.0380
Epoch 28/30

Epoch 00028: val_loss did not improve from 0.02231
314/314 - 150s - loss: 0.0807 - val_loss: 0.0284
Epoch 29/30

Epoch 00029: val_ loss did not improve from 0.02231

```
314/314 - 151s - loss: 0.0684 - val_loss: 0.0581
Epoch 30/30

Epoch 00030: val_loss did not improve from 0.02231
314/314 - 150s - loss: 0.0739 - val_loss: 0.0271
```

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



Evaluate Model 2.1

```
# 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
def generate_desc(model, 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]))
    # 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)
        # predict next word
        yhat = model.predict([photo,finding,sequence], 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(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(model, 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-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)))
```

Prepare test data and Evaluate

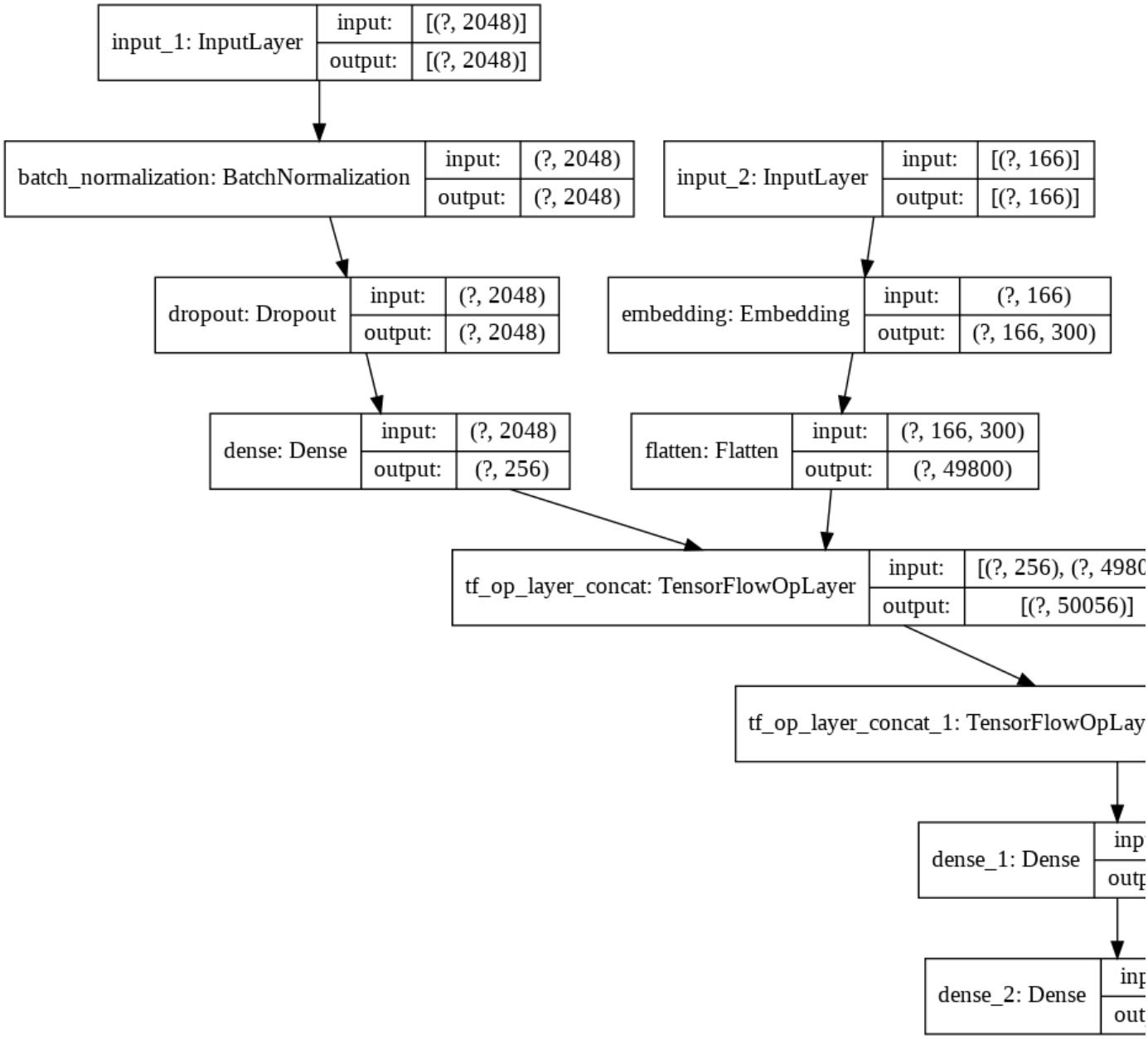
```
test_image_features = load_image_features(image_extracted_features,test_id)
print('\nTotal test images      : ' ,len(test_image_features))

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

#findings_max_length =166
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
```

```
# load the model
model1=tf.keras.models.load_model('/content/drive/My Drive/Xception/Model2/mergemodelcheck.hdf5')
# evaluate model
evaluate_model(model1, test_id_impressions, test_image_features, impressions_tokenizer, impressions_max_lengt
```

```
BLEU-1: 0.972408
BLEU-2: 0.967069
BLEU-3: 0.963182
BLEU-4: 0.948923
```

Model 2.2 : With Batch Normalization+Dropout1(0.3)+Dropo




```
# 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,))
    fe = tf.keras.layers.BatchNormalization()(inputs1)
    image1 = tf.keras.layers.Dropout(0.3)(fe)
    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)#tf.keras.layers.add
```

```
    # sequence model
    inputs3 = tf.keras.layers.Input(shape=(impressions_max_length,))
    impressions1 = tf.keras.layers.Embedding(impressions_vocab_size,300,weights=[impressions_embedding_matr
    impressions2 = tf.keras.layers.Dropout(0.3)(impressions1)
    impressions3 = tf.keras.layers.LSTM(256)(impressions2)

    # decoder model
    decoder1= tf.concat([im_f, impressions3],axis=1)
    #decoder1 = tf.keras.layers.add([im_f, impressions3])
    decoder2 = tf.keras.layers.Dense(700, 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/checkmergearchitecture.png', show_sha
    return model


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

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 2048)]       0
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 2048)         8192        input_1[0][0]
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 166)]        0
__________________________________________________________________________________________________
input_3 (InputLayer)            [(None, 114)]        0
__________________________________________________________________________________________________
dropout (Dropout)               (None, 2048)         0           batch_normalization[0][0]
__________________________________________________________________________________________________
embedding (Embedding)           (None, 166, 300)     468900      input_2[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 114, 300)     362400      input_3[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 256)          524544      dropout[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 49800)        0           embedding[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 114, 300)     0           embedding_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat (TensorFlowO [(None, 50056)]      0           dense[0][0]
                                                                 flatten[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     (None, 256)          570368      dropout_1[0][0]
__________________________________________________________________________________________________
tf_op_layer_concat_1 (TensorFlo [(None, 50312)]      0           tf_op_layer_concat[0][0]
                                                                 lstm[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 700)          35219100    tf_op_layer_concat_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 1208)         846808      dense_1[0][0]
==================================================================================================
Total params: 38,000,312
Trainable params: 37,164,916
Non-trainable params: 835,396
__________________________________________________________________________________________________
```

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

log_dir="/content/drive/My Drive/Xception/Model2/checkmergetensorboardlogs/logs/fit/" + datetime.now().strf
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=30, verbose=2,callbacks=[tensorboard_callback,checkpoint], va
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
Epoch 1/30

Epoch 00001: val_loss improved from inf to 1.25510, saving model to /content/drive/My Drive/Xception/M
314/314 - 170s - loss: 2.4755 - val_loss: 1.2551
Epoch 2/30

Epoch 00002: val_loss improved from 1.25510 to 0.67928, saving model to /content/drive/My Drive/Xcepti
314/314 - 171s - loss: 1.0146 - val_loss: 0.6793
Epoch 3/30

Epoch 00003: val_loss improved from 0.67928 to 0.39512, saving model to /content/drive/My Drive/Xcepti
314/314 - 168s - loss: 0.5753 - val_loss: 0.3951
Epoch 4/30

Epoch 00004: val_loss improved from 0.39512 to 0.25764, saving model to /content/drive/My Drive/Xcepti
314/314 - 169s - loss: 0.3583 - val_loss: 0.2576
Epoch 5/30

Epoch 00005: val_loss improved from 0.25764 to 0.20384, saving model to /content/drive/My Drive/Xcepti
314/314 - 169s - loss: 0.2417 - val_loss: 0.2038
Epoch 6/30

Epoch 00006: val_loss improved from 0.20384 to 0.14985, saving model to /content/drive/My Drive/Xcepti
314/314 - 170s - loss: 0.1756 - val_loss: 0.1498
Epoch 7/30

Epoch 00007: val_loss improved from 0.14985 to 0.12724, saving model to /content/drive/My Drive/Xcepti
314/314 - 170s - loss: 0.1377 - val_loss: 0.1272
Epoch 8/30

Epoch 00008: val_loss improved from 0.12724 to 0.12343, saving model to /content/drive/My Drive/Xcepti
314/314 - 169s - loss: 0.1253 - val_loss: 0.1234
Epoch 9/30

Epoch 00009: val_loss did not improve from 0.12343
314/314 - 168s - loss: 0.1156 - val_loss: 0.1313
Epoch 10/30

Epoch 00010: val_loss improved from 0.12343 to 0.12041, saving model to /content/drive/My Drive/Xcepti
314/314 - 169s - loss: 0.1061 - val_loss: 0.1204
Epoch 11/30

Epoch 00011: val_loss did not improve from 0.12041
314/314 - 168s - loss: 0.1019 - val_loss: 0.1262
Epoch 12/30

Epoch 00012: val_loss improved from 0.12041 to 0.07640, saving model to /content/drive/My Drive/Xcepti
314/314 - 169s - loss: 0.0958 - val_loss: 0.0764
Epoch 13/30

Epoch 00013: val_loss did not improve from 0.07640
314/314 - 167s - loss: 0.0909 - val_loss: 0.1221
Epoch 14/30

Epoch 00014: val_loss did not improve from 0.07640
314/314 - 167s - loss: 0.0919 - val_loss: 0.1195
Epoch 15/30
```

Epoch 00015: val_loss did not improve from 0.07640
314/314 - 167s - loss: 0.0850 - val_loss: 0.0867
Epoch 16/30

Epoch 00016: val_loss did not improve from 0.07640
314/314 - 167s - loss: 0.0925 - val_loss: 0.1007
Epoch 17/30

Epoch 00017: val_loss did not improve from 0.07640
314/314 - 167s - loss: 0.0813 - val_loss: 0.0827
Epoch 18/30

Epoch 00018: val_loss improved from 0.07640 to 0.06105, saving model to /content/drive/My Drive/Xcepti
314/314 - 170s - loss: 0.0692 - val_loss: 0.0610
Epoch 19/30

Epoch 00019: val_loss did not improve from 0.06105
314/314 - 167s - loss: 0.0625 - val_loss: 0.0753
Epoch 20/30

Epoch 00020: val_loss did not improve from 0.06105
314/314 - 166s - loss: 0.0688 - val_loss: 0.0798
Epoch 21/30

Epoch 00021: val_loss did not improve from 0.06105
314/314 - 165s - loss: 0.0696 - val_loss: 0.1083
Epoch 22/30

Epoch 00022: val_loss did not improve from 0.06105
314/314 - 167s - loss: 0.0729 - val_loss: 0.1452
Epoch 23/30

Epoch 00023: val_loss did not improve from 0.06105
314/314 - 166s - loss: 0.0703 - val_loss: 0.0733
Epoch 24/30

Epoch 00024: val_loss did not improve from 0.06105
314/314 - 166s - loss: 0.0602 - val_loss: 0.0936
Epoch 25/30

Epoch 00025: val_loss did not improve from 0.06105
314/314 - 167s - loss: 0.0566 - val_loss: 0.0664
Epoch 26/30

Epoch 00026: val_loss did not improve from 0.06105
314/314 - 167s - loss: 0.0590 - val_loss: 0.0661
Epoch 27/30

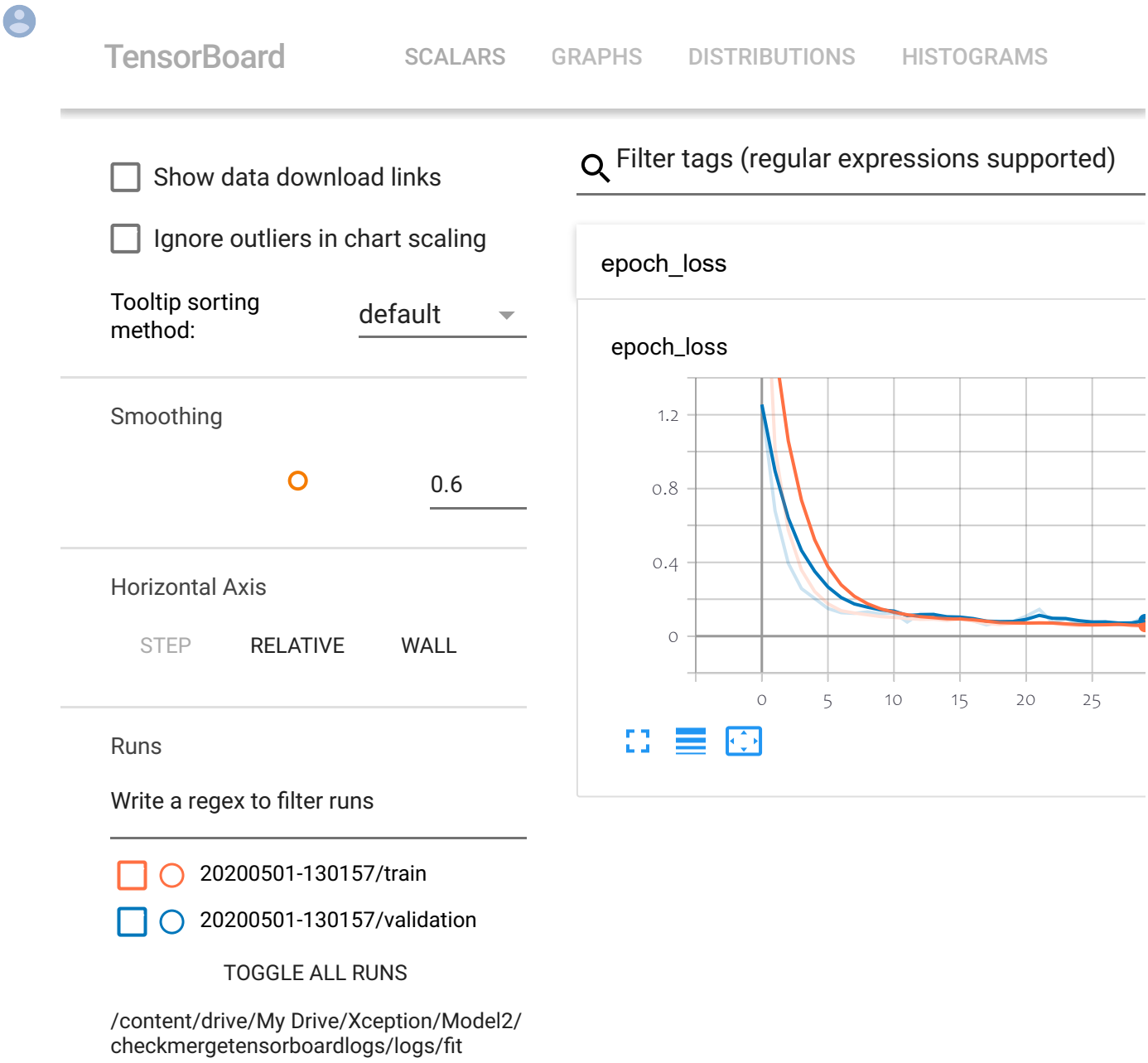
Epoch 00027: val_loss did not improve from 0.06105
314/314 - 167s - loss: 0.0622 - val_loss: 0.0779
Epoch 28/30

Epoch 00028: val_loss did not improve from 0.06105
314/314 - 166s - loss: 0.0663 - val_loss: 0.0620
Epoch 29/30

Epoch 00029: val_loss did not improve from 0.06105

```
314/314 - 167s - loss: 0.0550 - val_loss: 0.0726
Epoch 30/30

Epoch 00030: val_loss did not improve from 0.06105
314/314 - 166s - loss: 0.0526 - val_loss: 0.1030
```

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



Evaluate

```
# load the model
model1=tf.keras.models.load_model('/content/drive/My Drive/Xception/Model2/checkmergemodelcheck.hdf5')
# evaluate model
evaluate_model(model1, test_id_impressions, test_image_features, impressions_tokenizer, impressions_max_lengt
```

```
BLEU-1: 0.947188
BLEU-2: 0.939861
BLEU-3: 0.936384
BLEU-4: 0.914837
```

Results

Results of Previous Submission

```
x = PrettyTable()
x.field_names = ["Model","is_finding_used", "score"]
x.add_row(["Model 1", "no", 0.097])
x.add_row(["Model 2", "yes", 0.8440])
print(x)
```

```
+---------+-----------------+-------+
|  Model  | is_finding_used | score |
+---------+-----------------+-------+
| Model 1 |        no       | 0.097 |
| Model 2 |       yes       | 0.844 |
+---------+-----------------+-------+
```

```
x = PrettyTable()
x.field_names = ["Model","is_finding_used","is_BN_used","Dropout 1","Dropout 2", "BLUE score"]
x.add_row(["Model 1.1", "\tno","\tyes",0.4,0.3, 0.1100])
x.add_row(["Model 1.2", "\tno","\tno",0.5,0.5, 0.1309])
x.add_row(["Model 2.1", "\tyes","\tno",0.5,0.5, 0.9724])
x.add_row(["Model 2.2", "\tyes","\tyes",0.3,0.3, 0.9471])
print(x)
```

```
+-----------+-----------------+------------+-----------+-----------+------------+
|   Model   | is_finding_used | is_BN_used | Dropout 1 | Dropout 2 | BLUE score |
+-----------+-----------------+------------+-----------+-----------+------------+
| Model 1.1 |        no       |        yes |    0.4    |    0.3    |    0.11    |
| Model 1.2 |        no       |        no  |    0.5    |    0.5    |   0.1309   |
| Model 2.1 |        yes      |        no  |    0.5    |    0.5    |   0.9724   |
| Model 2.2 |        yes      |        yes |    0.3    |    0.3    |   0.9471   |
+-----------+-----------------+------------+-----------+-----------+------------+
```

Conclusion

- As we can see that After Applying Data Augmentation technique Model 2 has Performed ver
  tells us that our model2 is sensible which can predict on unseen data points.

- Now we can conclude that the use of findings plays a very important role in generating impre
- Another important thing is in this case study use of Batch Normalization affected performan

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) **Model 1**:

- we build our model 1 on image data+ impressions data only
- First we split data into train,cv and test sets.
- we define necessary functions after this we create input sequences where we convert text da 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

4) **Model 2**:

- we build this on [image data+findings] + impressions data
- we follow same procedure but the only change is we used findings also by adding this as inp

5) Summarize results