```
!pip install tensorflow==2.3
```

```
Requirement already satisfied: tensorflow==2.3 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: absl-py>=0.7.0 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: gast==0.3.3 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: tensorboard<3,>=2.3.0 in /usr/local/lib/python3.6/dist
Requirement already satisfied: h5py<2.11.0,>=2.10.0 in /usr/local/lib/python3.6/dist-
Requirement already satisfied: keras-preprocessing<1.2,>=1.1.1 in /usr/local/lib/pyth
Requirement already satisfied: astunparse==1.6.3 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: wrapt>=1.11.1 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: numpy<1.19.0,>=1.16.0 in /usr/local/lib/python3.6/dist
Requirement already satisfied: scipy==1.4.1 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: grpcio>=1.8.6 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: tensorflow-estimator<2.4.0,>=2.3.0 in /usr/local/lib/p
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.6/dist-pack
Requirement already satisfied: google-pasta>=0.1.8 in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: google-auth<2,>=1.6.3 in /usr/local/lib/python3.6/dist
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/pyt
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: rsa<5,>=3.1.4; python_version >= "3" in /usr/local/lib
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.6/dist
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.6/dis
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.6/d
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: importlib-metadata; python_version < "3.8" in /usr/loc
Requirement already satisfied: pyasn1>=0.1.3 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.6/dist-packages (f
```

```
!pip install imgaug==0.4
```

```
Collecting imgaug==0.4
  Downloading https://files.pythonhosted.org/packages/66/b1/af3142c4a85cba6da9f4ebb5f
     |████████████████████████████████| 952kB 9.0MB/s
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: scikit-image>=0.14.2 in /usr/local/lib/python3.6/dist-
Requirement already satisfied: Shapely in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from im
Requirement already satisfied: opencv-python in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: imageio in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: Pillow in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-
```

```
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.6/dist-pack
ERROR: albumentations 0.1.12 has requirement imgaug<0.2.7,>=0.2.5, but you'll have im
Installing collected packages: imgaug
  Found existing installation: imgaug 0.2.9
    Uninstalling imgaug-0.2.9:
      Successfully uninstalled imgaug-0.2.9
Successfully installed imgaug-0.4.0
```

```
!pip install SimpleITK
```

```
Collecting SimpleITK
  Downloading https://files.pythonhosted.org/packages/f3/cb/a15f4612af8e37f3627fc7fb2
     |████████████████████████████████| 44.9MB 63kB/s
Installing collected packages: SimpleITK
Successfully installed SimpleITK-2.0.1
```

```
#import libraries
import os
import numpy as np
import skimage.io as io
import SimpleITK as sitk
import matplotlib.pyplot as plt
import pandas as pd
import glob
import glob2
##mod unet

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#%matplotlib inline
import tensorflow as tf
import keras
import keras.backend as K
from keras.utils import to_categorical
from keras import metrics
from keras.models import Model, load_model
from keras.layers import Input, BatchNormalization, Activation, Dense, Dropout,Maximum

from keras.layers.convolutional import Conv2D, Conv2DTranspose,Conv3D,Conv3DTranspose
from keras.layers.pooling import MaxPooling2D, GlobalMaxPool2D,MaxPooling3D
from keras.layers.merge import concatenate, add
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
from keras.optimizers import Adam


from skimage.io import imread, imshow, concatenate_images
from skimage.transform import resize
from sklearn.utils import class_weight
```

```
from keras.callbacks import ModelCheckpoint
from keras.callbacks import CSVLogger
from keras.callbacks import EarlyStopping

import os
from skimage.io import imread, imshow, concatenate_images
from skimage.transform import resize
# from medpy.io import load
import numpy as np

#import cv2
import nibabel as nib
from PIL import Image
import random


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

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

EXPLORATORY DATA ANALYSIS

```
#TRAINING FILES
#LGG FILES
all_seg_files_lgg = glob2.glob('/content/training/LGG/**/*seg.nii.gz')
all_lgg_files=glob2.glob('/content/training/LGG/**/*.nii.gz')
training_lgg_files=[]
for i in all_lgg_files:
  if "_seg" not in i.split("/")[-1]:
    training_lgg_files.append(i)

#HGG FILES
all_seg_files_hgg = glob2.glob('/content/training/HGG/**/*seg.nii.gz')
all_hgg_files=glob2.glob('/content/training/HGG/**/*.nii.gz')
training_hgg_files=[]
for i in all_hgg_files:
  if "_seg" not in i.split("/")[-1]:
    training_hgg_files.append(i)

total_no_lgg_imgvolumes=len(training_lgg_files)
print("LGG VOLUMES",total_no_lgg_imgvolumes)
output_masks_lgg=len(all_seg_files_lgg)
print("LGG MASKS",output_masks_lgg)

total_no_hgg_imgvolumes=len(training_hgg_files)
print("HGG VOLUMES",total_no_hgg_imgvolumes)
output_masks_hgg=len(all_seg_files_hgg)
print("HGG MASKS",output_masks_hgg)
```

```
     LGG VOLUMES 300
     LGG MASKS 75
```

```
     HGG VOLUMES 840
     HGG MASKS 210
```

```
#observations
#Here there are two types of image volumes -LGG and HGG
#There are 4 types of modalities in each type of Lgg and Hgg volumes of each patient
#There are 300 LGG volumes and 75 LGG masks
#There are 840 HGG volumes and 210 HGG masks
#for each patient,there are 4 modalities and segmentation mask (ground truth label)
```

```
#VALIDATION FILES
all_val_files = glob2.glob("/content/validation/**/*nii.gz")
print("VALIDATION FILES ",len(all_val_files))
```

```
     VALIDATION FILES  264
```

```
#There are 264 validation files provided by the dataset(there is no specific lgg and hgg f
```

```
os.listdir("/content/training/LGG/Brats18_TCIA13_654_1/")
os.listdir("/content/training/HGG/Brats18_CBICA_AMH_1/")
```

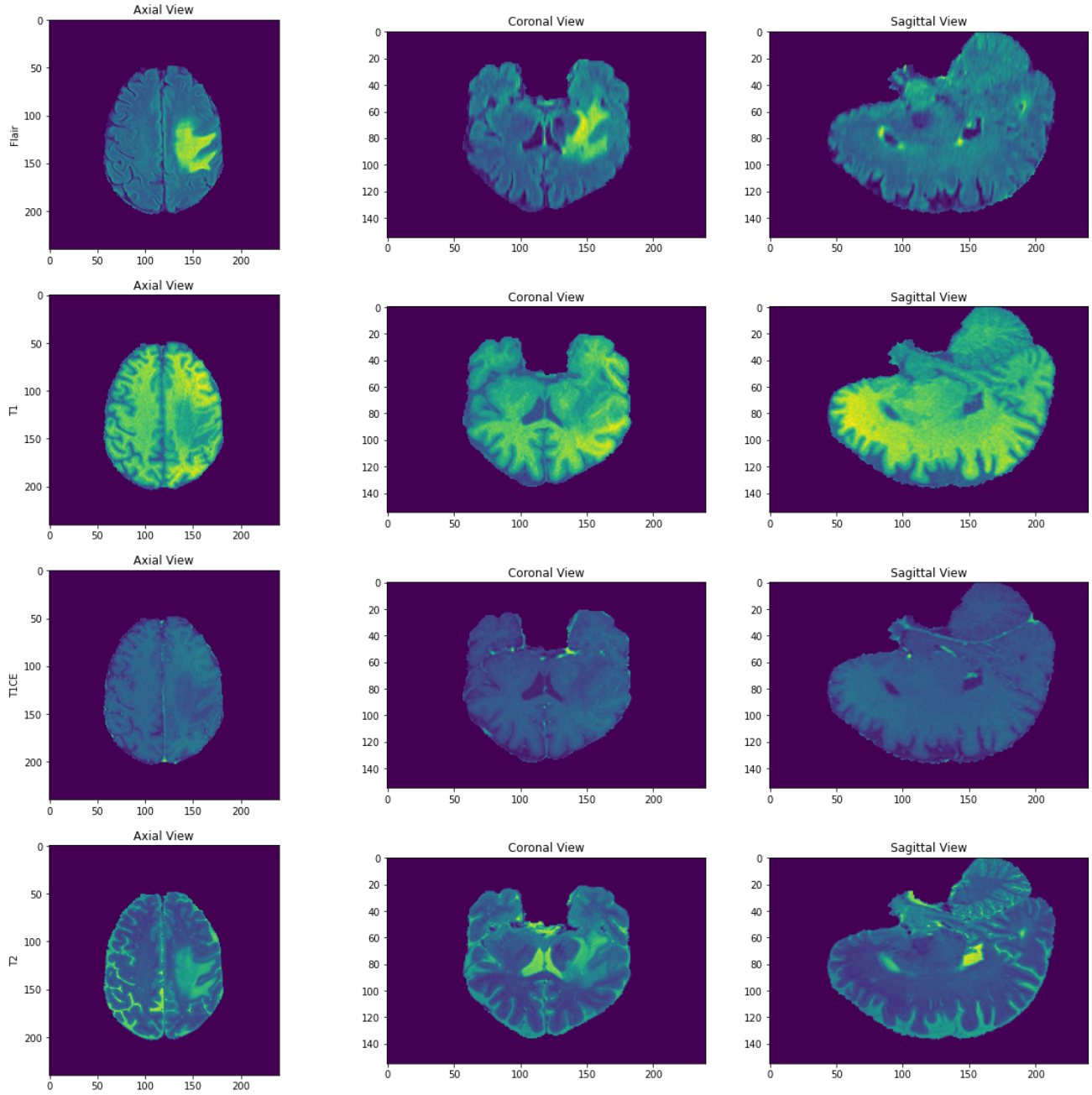
```
     ['Brats18_CBICA_AMH_1_seg.nii.gz',
      'Brats18_CBICA_AMH_1_t1ce.nii.gz',
      'Brats18_CBICA_AMH_1_t2.nii.gz',
      'Brats18_CBICA_AMH_1_t1.nii.gz',
      'Brats18_CBICA_AMH_1_flair.nii.gz']
```

```
#for each patient,there are 4 modalities and one segmentation mask
```

```
#to read each image volume as array
##lgg modalities
imglgg_flair = io.imread("/content/training/LGG/Brats18_TCIA13_654_1/Brats18_TCIA13_654_1_
imglgg_t1=io.imread("/content/training/LGG/Brats18_TCIA13_654_1/Brats18_TCIA13_654_1_t1.ni
imglgg_t1ce=io.imread("/content/training/LGG/Brats18_TCIA13_654_1/Brats18_TCIA13_654_1_t1c
imglgg_t2=io.imread("/content/training/LGG/Brats18_TCIA13_654_1/Brats18_TCIA13_654_1_t2.ni

#hgg modalities
imghgg_flair = io.imread("/content/training/HGG/Brats18_CBICA_AMH_1/Brats18_CBICA_AMH_1_fl
imghgg_t1=io.imread("/content/training/HGG/Brats18_CBICA_AMH_1/Brats18_CBICA_AMH_1_t1.nii.
imghgg_t1ce=io.imread("/content/training/HGG/Brats18_CBICA_AMH_1/Brats18_CBICA_AMH_1_t1ce.
imghgg_t2=io.imread("/content/training/HGG/Brats18_CBICA_AMH_1/Brats18_CBICA_AMH_1_t2.nii.
```

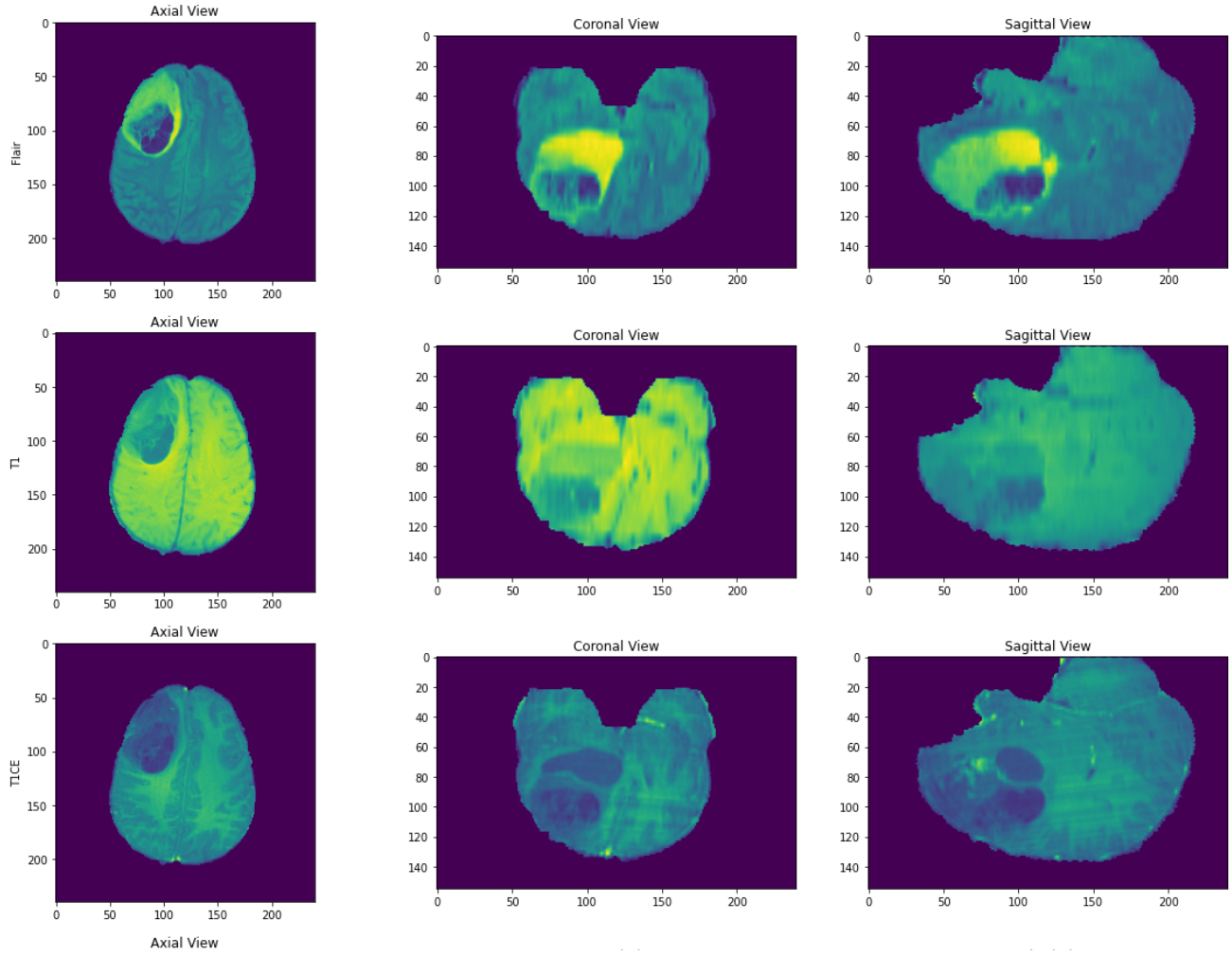
```
#lgg
print("shape of each modality of lgg",imglgg_flair.shape)

#hgg
print("shape of each modality of hgg",imghgg_flair.shape)
```

```
     shape of each modality of lgg (155, 240, 240)
     shape of each modality of hgg (155, 240, 240)


#Dimensions of the image volume=(depth,height,width)
#here depth/no.of slices that can be done=155
#height=240,width=240


#different views of image volumes of each modality of lgg and hgg
image_hgg_modalities=[imghgg_flair,imghgg_t1,imghgg_t1ce,imghgg_t2]
image_lgg_modalities=[imglgg_flair,imglgg_t1,imglgg_t1ce,imglgg_t2]


#visualization of a slice of each modality in different views of lgg and hgg
#there are 3 different views of a slice there are Axial,coronal,saggital view

#hgg modalities
k=1
plt.figure(figsize=(20,20))
for l,i in enumerate(image_hgg_modalities):
  for j in range(3):
    if (j==0):
      plt.subplot(4,3,k)
      plt.imshow(i[100,:,:])
      plt.title("Axial View")
      if(l==0):
        plt.ylabel("Flair")
      if(l==1):
        plt.ylabel("T1")
      if(l==2):
        plt.ylabel("T1CE")
      if(l==3):
        plt.ylabel("T2")

      k=k+1
    elif (j==1):
      plt.subplot(4,3,k)
      plt.imshow(i[:,100,:])
      plt.title("Coronal View")
      k+=1
    else:
      plt.subplot(4,3,k)
      plt.imshow(i[:,:,100])
      plt.title("Sagittal View")
      k+=1
 .
```




```
#lgg modalities
k=1
plt.figure(figsize=(20,20))
for l,i in enumerate(image_lgg_modalities):
  for j in range(3):
    if (j==0):
      plt.subplot(4,3,k)
      plt.imshow(i[100,:,:])
      plt.title("Axial View")
```

```
    if(l==0):
      plt.ylabel("Flair")
    if(l==1):
      plt.ylabel("T1")
    if(l==2):
      plt.ylabel("T1CE")
    if(l==3):
      plt.ylabel("T2")

    k=k+1
  elif (j==1):
    plt.subplot(4,3,k)
    plt.imshow(i[:,100,:])
    plt.title("Coronal View")
    k+=1
  else:
    plt.subplot(4,3,k)
    plt.imshow(i[:,:,100])
    plt.title("Sagittal View")
    k+=1
```




```
os.listdir("/content/HGG/Brats18_CBICA_AMH_1")
```

```
['Brats18_CBICA_AMH_1_flair.nii.gz',
 'Brats18_CBICA_AMH_1_t1.nii.gz',
 'Brats18_CBICA_AMH_1_seg.nii.gz',
 'Brats18_CBICA_AMH_1_t1ce.nii.gz',
 'Brats18_CBICA_AMH_1_t2.nii.gz']
```

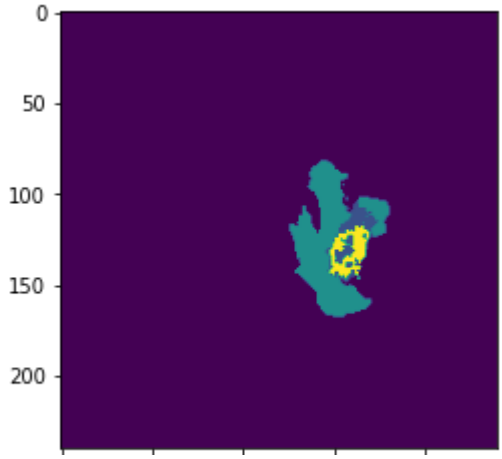


```
#to read segmentation mask/ground truth label of a patient of hgg and lgg

imglgg_seg=io.imread("/content/LGG/Brats18_TCIA10_387_1/Brats18_TCIA10_387_1_seg.nii.gz",
imghgg_seg=io.imread("/content/HGG/Brats18_CBICA_AMH_1/Brats18_CBICA_AMH_1_seg.nii.gz", pl


print("GROUND TRUTH LABEL/MASK SHAPE OF LGG",imglgg_seg.shape)
print("GROUND TRUTH LABEL/MASK SHAPE OF HGG",imghgg_seg.shape)
```

```
GROUND TRUTH LABEL/MASK SHAPE OF LGG (155, 240, 240)
GROUND TRUTH LABEL/MASK SHAPE OF HGG (155, 240, 240)
```

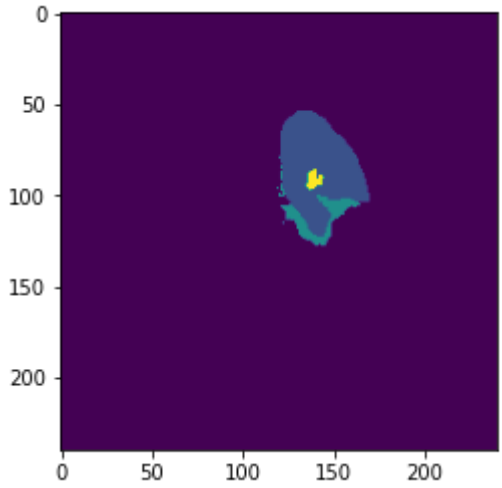
```
#Here the shape of segmentation mask of lgg and hgg is (155,240,240)
#depth=155


plt.imshow(imghgg_seg[75,:,:])
```

```
<matplotlib.image.AxesImage at 0x7f6fffe712e8>
```




```
plt.imshow(imglgg_seg[100,:,:])
```

```
<matplotlib.image.AxesImage at 0x7f6fff9b2ac8>
```




```
unique_labels_hgg=np.unique(imghgg_seg[75,:,:])
unique_labels_lgg=np.unique(imglgg_seg[100,:,:])

print("UNIQUE LABELS OF HGG",unique_labels_hgg)
print("UNIQUE LABELS OF LGG",unique_labels_lgg)
```

```
UNIQUE LABELS OF HGG [0 1 2 4]
UNIQUE LABELS OF LGG [0 1 2 4]
```

```
#There are 4 labels for segmentation mask/ground truth
# only tumor part is visble and segmented by 3 labels(green,yellow,blue) and another label


## to check for class imbalances in data for either lgg/hgg

#converted to one dimension
imghgg_seg.reshape(-1,1).shape

#converted to series data to plot value counts of each class
a=pd.Series(list(imghgg_seg.reshape(-1,1)))
b=a.value_counts()
print(b)
```

```
[0]    8802713
[2]     107610
[4]       9002
[1]       8675
dtype: int64
```

```
#observation:
#here we can observe that there are more no.of class0 labels which are non-tumorous
#so there is a high class imbalance in the data
```

```
b.index=[0,2,4,1]
```

```
b
```

```
0    8802713
2     107610
4       9002
1       8675
dtype: int64
```

```
#calculated the percentage of class labels and stored them in the dictionary
x={}
for i in b.index:
  x[i]=b[i]/sum(b.values)
```

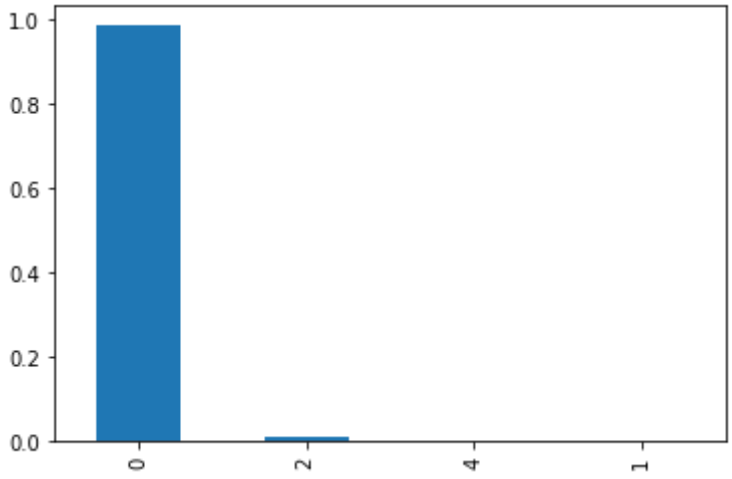
```
x
```

```
{0: 0.9859669578853046,
 1: 0.0009716621863799283,
 2: 0.012053091397849462,
 4: 0.0010082885304659498}
```

```
#visualization of class labels using a bar plot
```

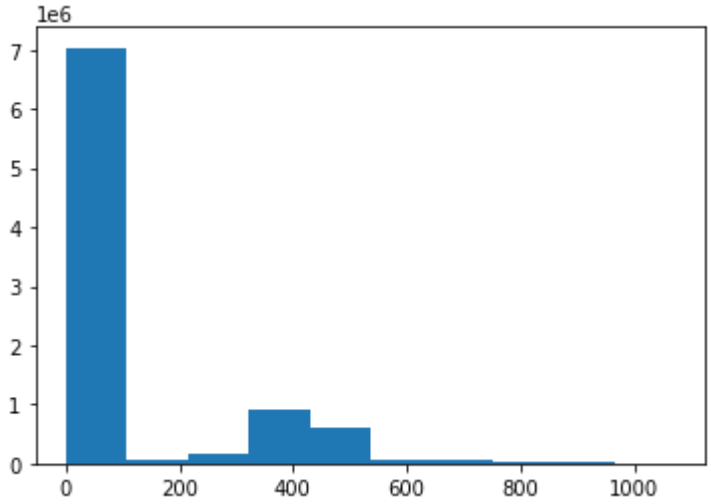
```
pd.Series(x).plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6fd3fc42e8>
```




```
#to plot histogram of pixel intensities of a modality hgg and lgg volume
```

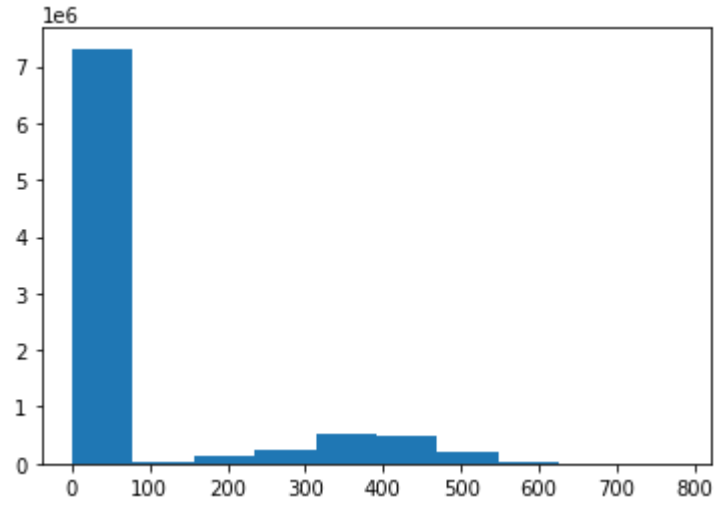
```
#to plot histogram of pixel intensities of a modality hgg and lgg volume
#to check whether they follow similar pixel distribution or not and also to check if any o

def plot_hist(img):
  img2=img.reshape(-1,1)
  plt.hist(img2)
  img3=pd.Series(list(img2))
  img4=img3.value_counts()

  return img4


#histogram for lgg
imglgg_flair_counts=plot_hist(imglgg_flair)
```



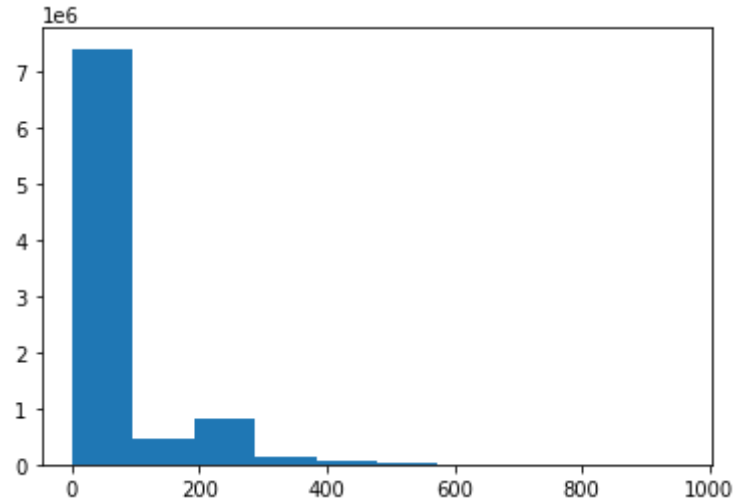

```
imglgg_flair_counts
```

```
[0]       7006363
[395]       10181
[403]       10127
[394]       10079
[401]       10072
             ...
[1007]           1
[1072]           1
[1052]           1
[1006]           1
[997]            1
Length: 1004, dtype: int64
```

```
#observation
# most of the pixel values are zero
#and pixel values are right skewed
# wen can also observe that pixel values arearound 0 and some of the pixels distrubuted ar
#and there are outliers as there are very few valuesof pixels with intensity after 500
#there is lot of variation in single volume
#we need to normalize so as to deal with skewness


#similary done for lgg of t1 modality
imghgg_t1_counts=plot_hist(imghgg_t1)
```




```
#histogram for hgg
imghgg_flair_counts=plot_hist(imghgg_flair)
```




```
# most of the pixel values are zero
#and pixel values are right skwed
# wen can also observe that pixel values arearound 0 and some of the pixels distrubuted ar
#and there are outliers as there are very few valuesof pixels with intensity after 300
#there is lot of variation in single volume
#we need to normalize so as to deal with skewness



#correlation matrix
#to check the correlation between slices along the depth of the image of hgg and lgg volum

#for hgg volume
#iterated with the depth of the image and taken slices so as to calculate correlation betw
#converted each slice pixel values into list of values(1 dimensional)
p=[]
for i in range(imghgg_flair.shape[0]):
    slice_=list(imghgg_flair[i,:,:].reshape(-1,1))
    p.append(slice_)
```

```
# converted slices pixel values into data frame so that each column represent a slice pixe
p_=np.array(p)
m=p_.reshape(155,57600).T

print(m.shape)
n=pd.DataFrame(m)
```

```
(57600, 155)
```

```
n.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(

5 rows × 155 columns

```
#no.of missing values
n.isna().sum().value_counts()
```

```
0    155
dtype: int64
```

```
#to check correlation of each column/slice
n.corr()
```

DATA PRE PROCESSING

```
#creating new directories for storing hgg and lgg files
#hgg
os.makedirs("new_data")
os.makedirs("new_data/FLAIR_HGG")
os.makedirs("new_data/MASK_HGG")
os.makedirs("new_data/T1_HGG")
os.makedirs("new_data/T1CE_HGG")
os.makedirs("new_data/T2_HGG")

#lgg

os.makedirs("new_data/FLAIR_LGG")
os.makedirs("new_data/MASK_LGG")
os.makedirs("new_data/T1_LGG")
os.makedirs("new_data/T1CE_LGG")
os.makedirs("new_data/T2_LGG")


def bias_field_correction(image_path,output_path):
  """
  Before giving to the pre-processing stage the raw data and is bias corrected as this
  bias signal is a very low frequency signal and smooth which will corrupt the mri image w
  image segmentation algorithms to process the images

  Function Parameters:
  --------------------
  image_path: path of the image
  output_path : corrected output path of image

  """
    img = sitk.ReadImage(image_path)
    img_mask = sitk.OtsuThreshold(img)
    img = sitk.Cast(img, sitk.sitkFloat32)
    corrector = sitk.N4BiasFieldCorrectionImageFilter()
    img_c = corrector.Execute(img, img_mask)
    sitk.WriteImage(img_c,output_path)


def noise_removal_data(source,patients,grade):

    """
  This functions performs the removal of bias signal from image volumes of lgg and hgg
  Input:
  source:Directory of patients image volumes of lgg and hgg
  patients:list of paths of patients directories
  grade:lgg type or hgg type volume

  """
  for patient in patients:
    #looping through modality of patients image volumes
    modalities=os.listdir(source+"/"+patient)
    v=[]
```

```
    for seq in modalities:
      #looping for each sequence of modalities of a patient
        #for FLAIR sequence
        if ("flair" in seq):
            #print(source+'/'+patient+'/'+seq)
            #print('FLAIR_LGG/'+patient+'/'+seq)
            bias_field_correction(source+'/'+patient+'/'+seq,'new_data/FLAIR_'+grade+'/'+s


        #t1 sequence
        if (("t1" in seq) and ("t1ce" not in seq)):

            bias_field_correction(source+'/'+patient+'/'+seq,'new_data/T1_'+grade+'/'+seq)



        #for T1CE sequence
        if ("t1ce" in seq):

            bias_field_correction(source+'/'+patient+'/'+seq,'new_data/T1CE_'+grade+'/'+se




        #for T2 sequence
        if ("t2" in seq):

            bias_field_correction(source+'/'+patient+'/'+seq,'new_data/T2_'+grade+'/'+seq)



source_lgg="training/LGG"
source_hgg="training/HGG"
patients_lgg=os.listdir(source_lgg)
patients_hgg=os.listdir(source_hgg)
noise_removal_data(source_lgg,patients_lgg,'LGG')
noise_removal_data(source_hgg,patients_hgg,'HGG')


def preprocessing(image_volume):
  """
  This function is used to preprocess the given corrected image volume of lgg ang hgg
  1.perform standardization for non zero pixels in array
  2.clipping image to range [-5,5]
  3.Normalizing non brain region pixels

  Function Parameters:
  Image_volume: Input image volume of lgg or hgg

  Returns:
  scaled_image:pre-processed image volume
  """

    #compute std dev and mean for non zero elements in array
    #standardization
```

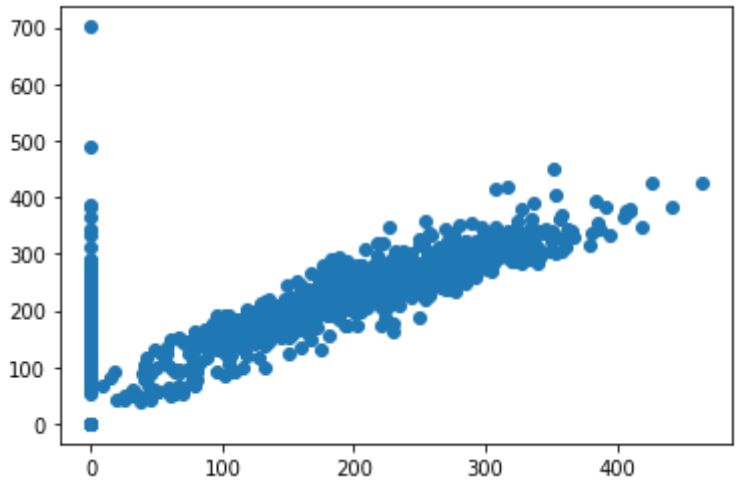
```
    std_dev=np.std(image_volume[np.nonzero(image_volume)])
    mean=np.mean(image_volume[np.nonzero(image_volume)])
    stdzn=(image_volume-mean)/std_dev

    #clipping the image to range [-5,5]
    clip_=np.clip(stdzn,-5,5)
    #to set non brain region to 0 before passing it to normalization
    mask_=(image_volume!=0)

    #after rescaling,multiply the rescaled image with mask to get image which has non brai

    rescaled_image=(clip_ - clip_.min()) / (clip_.max() - clip_.min())

    rescaled_image=mask_*rescaled_image

    return rescaled_image


#to take a threshold for min.no of non zero pixels

a=io.imread("training/HGG/Brats18_2013_2_1/Brats18_2013_2_1_flair.nii.gz", plugin='simplei
min_nonzeros=np.count_nonzero(a[18])


#creating new directory to collect slices of lgg and hgg
os.makedirs("new_data_slices")
#hgg
os.makedirs("new_data_slices/FLAIR_LGG")
os.makedirs("new_data_slices/T1_LGG")
os.makedirs("new_data_slices/T1CE_LGG")
os.makedirs("new_data_slices/T2_LGG")
os.makedirs("new_data_slices/MASK_LGG")
#lgg
os.makedirs("new_data_slices/FLAIR_LGG")
os.makedirs("new_data_slices/T1_LGG")
os.makedirs("new_data_slices/T1CE_LGG")
os.makedirs("new_data_slices/T2_LGG")
os.makedirs("new_data_slices/MASK_LGG")


def create_slices(grade_type,modality,grade,collected=False,b=[]):

    """
    This function create slices of an image volume
    Function parameters:
    ---------------------
     grade_type: list of image volumes of type of grade
     modality:type of image sequences -FLAIR,T1,T1CE,T2,MASK
     grade:HGG or LGG
     collected: False:collects the slice indexes of image volumes which are useful to take
     eg:if index 5  slice of a image flair volume is collected first then its same index i
     likewise done for all slices and all image volumes of patients
     b=[]:It is empty when no slice index is taken

     Returns:
     --------
```

	0	1	2	3	4	5	6	7	
0	1.000000	0.918879	0.834216	0.756760	0.688122	0.643113	0.609584	0.584395	0.5

```
#observation
#If the values of the ith or jth feature/slice do not vary,
#then the respective standard deviation will be zero and so will the denominator of the fr
#here from above table141th slice there is no variation in the pixel value,so these slices
```

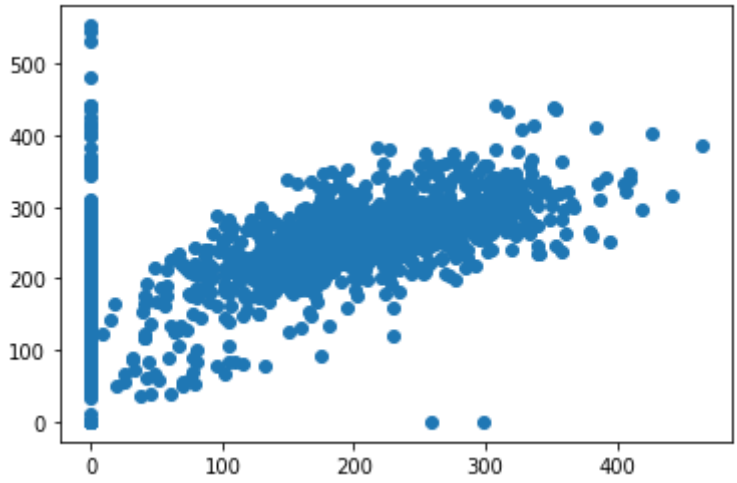
```
#visualization of correlation between successive slices using scatter plot
#slices of 0 &1
plt.scatter(n[0],n[1])
```

```
<matplotlib.collections.PathCollection at 0x7f6fd379fc18>
```




```
#slices of 0&2
plt.scatter(n[0],n[2])
```

```
<matplotlib.collections.PathCollection at 0x7f6fd33170f0>
```




```
#observation
#for every slice there is strong coreleation between 2 or 3 slice around that slice.
#for example 0th slice is having strong correlation to,1 ,2,3.and 4th slice having correla
```

```
     b:slice indexes

    """

    count=0
    for image in grade_type:
        image_volume=io.imread("new_data/"+modality+"_"+grade+"/"+image, plugin='simpleitk
        x=image.split('.')[0]

        patient=x.replace(x.split("_")[-1],"")
        #image_volume=preprocessing(image_volume)

        if (collected == False):
            v=[]
            for slice_ in range(image_volume.shape[0]):
                if (slice_%2==1) and (slice_<141):
                    if(np.count_nonzero(image_volume[slice_])>=min_nonzeros):
                        np.save("new_data_slices/"+modality+"_"+grade+"/"+patient+modality
                        v.append(slice_)

            b.append(v)

        else:

            while(count<len(b)):

           # image_volume=io.imread("new_data/"+modality+"_"+grade+"/"+image, plugin='simp


                for slice_other_modality in b[count]:
                    #print(slice_other_modality)
                    #break
                    val=image_volume[slice_other_modality]

                    np.save("new_data_slices/"+modality+"_"+grade+"/"+patient+modality+"_"
                count+=1
                break

    return b



#unzipping the files where all slices are created for all image volumes
#here lgg files are taken


!unzip "/content/drive/My Drive/new_data_slices_lgg.zip"
```

```
     Streaming output truncated to the last 5000 lines.
       inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_65.npy
       inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_67.npy
       inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_69.npy
       inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_71.npy
       inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_73.npy
       inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_75.npy
       inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_77.npy
```

```
  inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_79.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_81.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_83.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_85.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_87.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_89.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_91.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_93.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_95.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_97.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_639_1_FLAIR_99.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_101.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_103.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_105.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_107.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_109.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_11.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_111.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_113.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_115.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_117.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_119.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_13.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_15.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_17.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_19.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_21.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_23.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_25.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_27.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_29.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_31.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_33.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_35.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_37.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_39.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_41.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_43.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_45.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_47.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_49.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_51.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_53.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_55.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_57.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_59.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_61.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_63.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_65.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_67.npy
  inflating: FLAIR_LGG/Brats18_TCIA10_640_1_FLAIR_69.npy
```

```
#store these slices
flair_lgg=os.listdir("/content/FLAIR_LGG")
mask_lgg=os.listdir("/content/MASK_LGG")
t1ce_lgg=os.listdir("/content/T1CE_LGG")
t2_lgg=os.listdir("/content/T2_LGG")
t1_lgg=os.listdir("/content/T1_LGG")

flair_lgg.sort()
t1_lgg.sort()
```

```
t1ce_lgg.sort()
t2_lgg.sort()
mask_lgg.sort()


flair_hgg=os.listdir("/content/FLAIR_HGG")
t1_hgg=os.listdir("/content/T1_HGG")
t2_hgg=os.listdir("/content/T2_HGG")
t1ce_hgg=os.listdir("/content/T1CE_HGG")
mask_hgg=os.listdir("/content/MASK_HGG")


flair_hgg.sort()
t1_hgg.sort()
t1ce_hgg.sort()
t2_hgg.sort()
mask_hgg.sort()


#stores these files in array format -lgg
flair_l=[]
t1_l=[]
t2_l=[]
t1ce_l=[]
mask_l=[]
for a,b,c,d,e in zip(flair_lgg,t1_lgg,t2_lgg,t1ce_lgg,mask_lgg):
  l="/content/FLAIR_LGG/"+a
  m="/content/T1_LGG/"+b
  n="/content/T2_LGG/"+c
  o="/content/T1CE_LGG/"+d
  p="/content/MASK_LGG/"+e

  flair_l.append(l)
  t1_l.append(m)
  t2_l.append(n)
  t1ce_l.append(o)
  mask_l.append(p)




#hgg
flair_h=[]
t1_h=[]
t2_h=[]
t1ce_h=[]
mask_h=[]
for a,b,c,d,e in zip(flair_hgg,t1_hgg,t2_hgg,t1ce_hgg,mask_hgg):
  l="/content/FLAIR_HGG/"+a
  m="/content/T1_HGG/"+b
  n="/content/T2_HGG/"+c
  o="/content/T1CE_HGG/"+d
  p="/content/MASK_HGG/"+e

  flair_h.append(l)
  t1_h.append(m)
```

```
  t2_h.append(n)
  t1ce_h.append(o)
  mask_h.append(p)



#dataframe
import pandas as pd
df=pd.DataFrame()
df['flair']=flair_l
df['t1']=t1_l
df['t1ce']=t1ce_l
df['t2']=t2_l
df['mask']=mask_l


df.head()
```

	flair	
0	/content/FLAIR_LGG/Brats18_2013_0_1_FLAIR_101.npy	/content/T1_LGG/Brats18_2013_0_1_T
1	/content/FLAIR_LGG/Brats18_2013_0_1_FLAIR_103.npy	/content/T1_LGG/Brats18_2013_0_1_T
2	/content/FLAIR_LGG/Brats18_2013_0_1_FLAIR_105.npy	/content/T1_LGG/Brats18_2013_0_1_T
3	/content/FLAIR_LGG/Brats18_2013_0_1_FLAIR_107.npy	/content/T1_LGG/Brats18_2013_0_1_T
4	/content/FLAIR_LGG/Brats18_2013_0_1_FLAIR_109.npy	/content/T1_LGG/Brats18_2013_0_1_T

```
# classes for data loading and preprocessing
#to create masks for each slice
classes=[0,1,2,4]
class Dataset:
    """Read images, apply augmentation and preprocessing transformations.

    Args:
        images_dir : path to images folder (directories of all sequences)
        masks_dir : path to segmentation masks folder
        classes : values of classes to extract from segmentation mask


    """

    def __init__(
            self,
            flair_paths,t1_paths,t2_paths,t1ce_paths,mask_paths,

            augmentation=None,
            classes=classes,
    ):
        self.images_flair =flair_paths
        self.images_t1 =t1_paths
        self.images_t2 =t2_paths
        self.images_t1ce =t1ce_paths
```

```
        self.masks_fps =mask_paths
        self.classes=classes
        self.augmentation = augmentation

    def __getitem__(self, i):

        # read data
        #print(self.images_flair[i])
        #print(i)
        image_flair = np.load(str (self.images_flair[i]))
        #image=image.reshape(240,240,1)
        image_t1 = np.load(str (self.images_t1[i]))
        image_t1ce = np.load(str (self.images_t1ce[i]))
        image_t2 = np.load(str (self.images_t2[i]))
        mask = np.load(str (self.masks_fps[i]))


        # extract certain classes from mask (e.g. cars)
        masks = [(mask == v) for v in self.classes]
        mask = np.stack(masks, axis=-1).astype('float')
        image=np.stack((image_flair,image_t1,image_t1ce,image_t2), axis=-1).astype('float'


        return image, mask


    def __len__(self):
      return len(self.masks_fps)



from sklearn.model_selection import train_test_split
X=df.drop(["mask"],axis=1)
y=df["mask"]
```

SPLIT DATA

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=10)


X_train.head()
```

```
train_data_unet= Dataset(
    flair_paths=list(X_train["flair"]),t1_paths=list(X_train["t1"]),t1ce_paths=list(X_trai
    classes=classes

)

test_data_unet= Dataset(
    flair_paths=list(X_test["flair"]),t1_paths=list(X_test["t1"]),t1ce_paths=list(X_test["
    classes=classes

)
```

```
train_data_unet[0][0].shape
```

```
(240, 240, 4)
```

```
#shape of each patch
import numpy as np
np.max(train_data_unet[55][0])
```

```
0.9384082555770874
```

```
import tensorflow as tf

class Dataloder(tf.keras.utils.Sequence):
    """Load data from dataset and form batches

    Args:
        dataset: instance of Dataset class for image loading and preprocessing.
        batch_size: Integet number of images in batch.
        shuffle: Boolean, if `True` shuffle image indexes each epoch.
    """

    def __init__(self, dataset, batch_size=1, shuffle=False):
        self.dataset = dataset
        self.batch_size = batch_size
        self.shuffle = shuffle
        self.indexes = np.arange(len(dataset))

        self.on_epoch_end()

    def __getitem__(self, i):

        # collect batch data
        start = i * self.batch_size
        stop = (i + 1) * self.batch_size
        data = []
        for j in range(start, stop):
            data.append(self.dataset[j])
```

```
        # transpose list of lists
        batch = [np.stack(samples, axis=0) for samples in zip(*data)]

        return tuple(batch)

    def __len__(self):
        """Denotes the number of batches per epoch"""
        return len(self.indexes) // self.batch_size

    def on_epoch_end(self):
        """Callback function to shuffle indexes each epoch"""
        if self.shuffle:
            self.indexes = np.random.permutation(self.indexes)
```

MODELLING OF DATA

BASELINE UNET MODEL

```
#ref:https://github.com/shalabh147/Brain-Tumor-Segmentation-and-Survival-Prediction-using-
def conv_block(input_mat,num_filters,kernel_size,batch_norm):
  X = Conv2D(num_filters,kernel_size=(kernel_size,kernel_size),strides=(1,1),padding='same
  if batch_norm:
    X = BatchNormalization()(X)

  X = Activation('relu')(X)

  X = Conv2D(num_filters,kernel_size=(kernel_size,kernel_size),strides=(1,1),padding='same
  if batch_norm:
    X = BatchNormalization()(X)

  X = Activation('relu')(X)

  return X

def Unet(input_img, n_filters = 128, dropout = 0.2, batch_norm = True):

  c1 = conv_block(input_img,n_filters,3,batch_norm)
  p1 = MaxPooling2D(pool_size=(2, 2), strides=2)(c1)
  p1 = Dropout(dropout)(p1)

  c2 = conv_block(p1,n_filters*2,3,batch_norm);
  p2 = MaxPooling2D(pool_size=(2,2) ,strides=2)(c2)
  p2 = Dropout(dropout)(p2)

  c3 = conv_block(p2,n_filters*4,3,batch_norm);
  p3 = MaxPooling2D(pool_size=(2,2) ,strides=2)(c3)
  p3 = Dropout(dropout)(p3)

  c4 = conv_block(p3,n_filters*8,3,batch_norm);
  p4 = MaxPooling2D(pool_size=(2,2) ,strides=2)(c4)
```

```
  p4 = Dropout(dropout)(p4)

  c5 = conv_block(p4,n_filters*16,3,batch_norm);

  u6 = Conv2DTranspose(n_filters*8, (3,3), strides=(2, 2), padding='same')(c5);
  u6 = concatenate([u6,c4]);
  c6 = conv_block(u6,n_filters*8,3,batch_norm)
  c6 = Dropout(dropout)(c6)
  u7 = Conv2DTranspose(n_filters*4,(3,3),strides = (2,2) , padding= 'same')(c6);

  u7 = concatenate([u7,c3]);
  c7 = conv_block(u7,n_filters*4,3,batch_norm)
  c7 = Dropout(dropout)(c7)
  u8 = Conv2DTranspose(n_filters*2,(3,3),strides = (2,2) , padding='same')(c7);
  u8 = concatenate([u8,c2]);

  c8 = conv_block(u8,n_filters*2,3,batch_norm)
  c8 = Dropout(dropout)(c8)
  u9 = Conv2DTranspose(n_filters,(3,3),strides = (2,2) , padding='same')(c8);

  u9 = concatenate([u9,c1]);

  c9 = conv_block(u9,n_filters,3,batch_norm)
  outputs = Conv2D(4, (1, 1), activation='softmax')(c9)

  model = Model(inputs=input_img, outputs=outputs)

  return model


def dice_coef(y_true, y_pred, epsilon=0.00001):
    """
    Dice = (2*|X & Y|)/ (|X|+ |Y|)
         =  2*sum(|A*B|)/(sum(A^2)+sum(B^2))
    ref: https://arxiv.org/pdf/1606.04797v1.pdf

    """
    axis = (0,1,2)
    dice_numerator = 2. * K.sum(y_true * y_pred, axis=axis) + epsilon
    dice_denominator = K.sum(y_true*y_true, axis=axis) + K.sum(y_pred*y_pred, axis=axis) +
    return K.mean((dice_numerator)/(dice_denominator))

def dice_coef_loss(y_true, y_pred):
    return 1-dice_coef(y_true, y_pred)



input_img = Input((240,240,4))
model_unet = Unet(input_img,128,0.14,True)
learning_rate =0.00095
model_unet.compile(optimizer=Adam(lr=learning_rate), loss=dice_coef_loss, metrics=[dice_co

model_unet.summary()
```

```
    Model: "functional_3"
    _________________________________________________________________
```

Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	[(None, 240, 240, 4)	0	
conv2d_19 (Conv2D)	(None, 240, 240, 128	4736	input_2[0][0]
batch_normalization_18 (BatchNo	(None, 240, 240, 128	512	conv2d_19[0][0]
activation_18 (Activation)	(None, 240, 240, 128	0	batch_normalizati
conv2d_20 (Conv2D)	(None, 240, 240, 128	147584	activation_18[0][
batch_normalization_19 (BatchNo	(None, 240, 240, 128	512	conv2d_20[0][0]
activation_19 (Activation)	(None, 240, 240, 128	0	batch_normalizati
max_pooling2d_4 (MaxPooling2D)	(None, 120, 120, 128	0	activation_19[0][
dropout_7 (Dropout)	(None, 120, 120, 128	0	max_pooling2d_4[0
conv2d_21 (Conv2D)	(None, 120, 120, 256	295168	dropout_7[0][0]
batch_normalization_20 (BatchNo	(None, 120, 120, 256	1024	conv2d_21[0][0]
activation_20 (Activation)	(None, 120, 120, 256	0	batch_normalizati
conv2d_22 (Conv2D)	(None, 120, 120, 256	590080	activation_20[0][
batch_normalization_21 (BatchNo	(None, 120, 120, 256	1024	conv2d_22[0][0]
activation_21 (Activation)	(None, 120, 120, 256	0	batch_normalizati
max_pooling2d_5 (MaxPooling2D)	(None, 60, 60, 256)	0	activation_21[0][
dropout_8 (Dropout)	(None, 60, 60, 256)	0	max_pooling2d_5[0
conv2d_23 (Conv2D)	(None, 60, 60, 512)	1180160	dropout_8[0][0]
batch_normalization_22 (BatchNo	(None, 60, 60, 512)	2048	conv2d_23[0][0]
activation_22 (Activation)	(None, 60, 60, 512)	0	batch_normalizati
conv2d_24 (Conv2D)	(None, 60, 60, 512)	2359808	activation_22[0][
batch_normalization_23 (BatchNo	(None, 60, 60, 512)	2048	conv2d_24[0][0]
activation_23 (Activation)	(None, 60, 60, 512)	0	batch_normalizati
max_pooling2d_6 (MaxPooling2D)	(None, 30, 30, 512)	0	activation_23[0][
dropout_9 (Dropout)	(None, 30, 30, 512)	0	max_pooling2d_6[0
conv2d_25 (Conv2D)	(None, 30, 30, 1024)	4719616	dropout_9[0][0]
batch_normalization_24 (BatchNo	(None, 30, 30, 1024)	4096	conv2d_25[0][0]

```
train_dataloader_unet = Dataloder(train_data_unet, batch_size=4, shuffle=True)
valid_dataloader_unet = Dataloder(test_data_unet, batch_size=1, shuffle=False)
```

```
# check shapes for errors
assert train_dataloader_unet[0][0].shape == (4, 240, 240,4)
assert train_dataloader_unet[0][1].shape == (4, 240, 240, 4)


tf.compat.v1.enable_eager_execution()


%load_ext tensorboard
# Clear any logs from previous runs
!rm -rf ./logs/
import tensorflow as tf
import datetime


log_dir="logs/fit/model-unet/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")


model_unet.load_weights("/content/drive/My Drive/128-cs2/weights-18-0.7679.hdf5")


# define callbacks for learning rate scheduling and best checkpoints saving
checkpoint_reducelr = [
    tf.keras.callbacks.ModelCheckpoint('/content/drive/My Drive/128-cs2/weights-{epoch:02d
    tf.keras.callbacks.ReduceLROnPlateau(

    monitor='val_dice_coef', factor=0.00002, patience=3, verbose=0, mode='max',
    min_delta=0.0001, cooldown=0, min_lr=0
),tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,write_
]
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoa
```

```
model_unet.optimizer.lr
```

```
<tf.Variable 'learning_rate:0' shape=() dtype=float32, numpy=0.00095>
```

```
history = model_unet.fit_generator(train_dataloader, steps_per_epoch=len(train_dataloader)
```

```
WARNING:tensorflow:From <ipython-input-34-9790b5ea1e94>:1: Model.fit_generator (from
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/350
  1/769 [..............................] - ETA: 0s - loss: 0.0675 - dice_coef: 0.9325
Instructions for updating:
use `tf.profiler.experimental.stop` instead.
  2/769 [..............................] - ETA: 6:37 - loss: 0.1739 - dice_coef: 0.82
769/769 [==============================] - ETA: 0s - loss: 0.2505 - dice_coef: 0.7495

Epoch 00001: val_dice_coef improved from -inf to 0.75507, saving model to /content/dr
769/769 [==============================] - 912s 1s/step - loss: 0.2505 - dice_coef: 0
Epoch 2/350
769/769 [==============================] - ETA: 0s - loss: 0.2576 - dice_coef: 0.7424
Epoch 00002: val_dice_coef improved from 0.75507 to 0.75889, saving model to /content
769/769 [==============================] - 912s 1s/step - loss: 0.2576 - dice_coef: 0
```

```
 Epoch 3/350
 698/769 [==========================>...] - ETA: 1:13 - loss: 0.2385 - dice_coef: 0.76
```

```
#unet best val dice coeff=0.7679


#prediction of unet model
def predict_(slice_):

  #image
  img_data=valid_dataloader_unet[slice_]
  image=img_data[0]# original image


  #ground truth image
  ground_truth=img_data[1][0]
  ground_truth=np.argmax(ground_truth,axis=-1)
  print("unique_classes in ground truth :",np.unique(ground_truth))


  #predicted image

  pred_img=model_unet.predict(image)
  pred_img=np.argmax(pred_img[0],axis=-1)
  print("unique_classes in predicted image :",np.unique(pred_img))

  return image,ground_truth,pred_img




image,ground_truth,pred_img=predict_(160)
```

```
 unique_classes in ground truth : [0 1 2]
 unique_classes in predicted image : [0 1 2]
```

```
#plot predictions
k=1
plt.figure(figsize=(30,30))
slice_no=[57,63,85,160,210]
for i in slice_no:
  image,groundtruth,pred_image=predict_(i)
  l=[image[0][:,:,2],groundtruth,pred_image] #taken for a single modality
  for i in l:
    plt.subplot(5,3,k)
    plt.imshow(i)
    if(k==1):
      plt.title("original image",fontdict={"fontsize":15})
    if(k==2):
      plt.title("ground Truth",fontdict={"fontsize":15})
    if(k==3):
      plt.title("predicted image",fontdict={"fontsize":15})

    k=k+1
```

EXPERIMENTATION TO IMPROVE THE RESULTS

```
unique_classes in ground truth : [0 1 2]
```

CANNET BASIC MODEL

```
unique classes in predicted image : [0 1 2]
```

```
import tensorflow as tf
# tf.compat.v1.enable_eager_execution()
from tensorflow import keras
from tensorflow.keras.layers import *
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import UpSampling2D
from tensorflow.keras.layers import MaxPooling2D, GlobalAveragePooling2D
from tensorflow.keras.layers import concatenate
from tensorflow.keras.layers import Multiply
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Input, Add, Dense, Activation, ZeroPadding2D, BatchNor
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.utils import plot_model
from tensorflow.keras.initializers import glorot_uniform
K.set_image_data_format('channels_last')
K.set_learning_phase(1)
```

```
WARNING:tensorflow:From <ipython-input-31-ab61530d47ee>:19: set_learning_phase (from
Instructions for updating:
Simply pass a True/False value to the `training` argument of the `__call__` method of
```


```
class stage1(tf.keras.layers.Layer):
    def __init__(self):
        super().__init__()


        self.conv=tf.keras.layers.Conv2D(64, kernel_size=(3,3), name='conv', padding="same
        self.batchn=tf.keras.layers.BatchNormalization( name='bn_conv')
        self.actv=tf.keras.layers.Activation('relu')
        self.maxp=tf.keras.layers.MaxPooling2D(pool_size=(2,2), strides=2)

    def call(self,X):

        #stage1
        op_1=self.conv(X)
        #print(op_1.shape)
        op_2=self.batchn(op_1)
        op_3=self.actv(op_2)
        op_4=self.maxp(op_3)

        return op_4
```

```
import tensorflow as tf

class convolutional_block(tf.keras.layers.Layer):
    def __init__(self, kernel=3,  filters=[4,4,8], stride=2, name="conv block"):
        super().__init__(name=name)
        self.F1, self.F2, self.F3 = filters
        self.kernel = kernel
        self.stride = stride

        self.conv1 = tf.keras.layers.Conv2D(self.F1,kernel_size=(1,1),strides=(self.stride
        self.bn1     = tf.keras.layers.BatchNormalization()
        self.bn2     = tf.keras.layers.BatchNormalization()
        self.bn3     = tf.keras.layers.BatchNormalization()
        self.bn4     = tf.keras.layers.BatchNormalization()
        self.act1    = tf.keras.layers.Activation('relu')
        self.act2    = tf.keras.layers.Activation('relu')
        self.act3    =tf.keras.layers.Activation('relu')
        self.act4    =tf.keras.layers.Activation('relu')
        self.conv2 = tf.keras.layers.Conv2D(self.F2,kernel_size=(3,3),padding='same')
        self.conv3 = tf.keras.layers.Conv2D(self.F3,kernel_size=(1,1))
        self.conv_parallel = tf.keras.layers.Conv2D(self.F3,kernel_size=(3,3),strides=(sel
        self.add = tf.keras.layers.Add()
    def call(self, X):
        # write the architecutre that was mentioned above

        conv_1 = self.conv1(X)
        bn_1   = self.bn1(conv_1)
        act_1   = self.act1(bn_1)
        conv_2 = self.conv2(act_1)
        bn_2   = self.bn2(conv_2)
        act_2   = self.act2(bn_2)
        conv_3 = self.conv3(act_2)
        bn_3   = self.bn3(conv_3)

        # parallel
        conv_p = self.conv_parallel(X)
        bn_4   = self.bn4(conv_p)
        act_3   = self.act3(bn_4)
        # element wise sum
        ele_sum = self.add([act_3,bn_3])

        X = self.act4(ele_sum)


        return X




class identity_block(tf.keras.layers.Layer):
    def __init__(self, kernel=3,  filters=[4,4,8], name="identity block"):
        super().__init__(name=name)
        self.F1, self.F2, self.F3 = filters
        self.kernel = kernel
```

```
        self.conv1 = tf.keras.layers.Conv2D(self.F1,kernel_size=(1,1))
        self.bn1     = tf.keras.layers.BatchNormalization()
        self.bn2     = tf.keras.layers.BatchNormalization()
        self.bn3     = tf.keras.layers.BatchNormalization()
        self.bn4     = tf.keras.layers.BatchNormalization()
        self.act1    = tf.keras.layers.Activation('relu')
        self.act2    = tf.keras.layers.Activation('relu')
        self.act3    =tf.keras.layers.Activation('relu')
        self.act4    =tf.keras.layers.Activation('relu')
        self.conv2 = tf.keras.layers.Conv2D(self.F2,kernel_size=(3,3),padding='same')
        self.conv3 = tf.keras.layers.Conv2D(self.F3,kernel_size=(1,1))
        self.add = tf.keras.layers.Add()
    def call(self, X):
        # write the architecutre that was mentioned above
        conv_1 = self.conv1(X)
        bn_1   = self.bn1(conv_1)
        act_1   = self.act1(bn_1)
        conv_2 = self.conv2(act_1)
        bn_2   = self.bn2(conv_2)
        act_2   = self.act2(bn_2)
        conv_3 = self.conv3(act_2)
        bn_3   = self.bn3(conv_3)
        out=self.add([X,bn_3])
        X=self.act4 (out)


        return X


class global_flow(tf.keras.layers.Layer):
    def __init__(self, name="global_flow"):
        super().__init__(name=name)
        self.glob=tf.keras.layers.GlobalAveragePooling2D()
        self.conv=tf.keras.layers.Conv2D(32,kernel_size=(1,1))
        self.bn=tf.keras.layers.BatchNormalization()
        self.act=tf.keras.layers.Activation('relu')
        self.up=tf.keras.layers.UpSampling2D(size=(28,28),interpolation='bilinear')

    def call(self, X):
        # implement the global flow operatiom
        out=self.glob(X)
        out=tf.expand_dims(out,1)
        out=tf.expand_dims(out,1)
        out1=self.bn(out)
        out2=self.act(out1)
        out3=self.conv(out2)
        X=self.up(out3)


        return X


class context_flow(tf.keras.layers.Layer):
    def __init__(self, name="context_flow"):
        super().__init__(name=name)
```

```
        self.avg=tf.keras.layers.AveragePooling2D(pool_size=(2, 2), strides=2, padding='va
        self.conv1 = tf.keras.layers.Conv2D(32,kernel_size=(3,3),padding='same')
        self.conv2 = tf.keras.layers.Conv2D(32,kernel_size=(3,3),padding='same')
        self.conv3 = tf.keras.layers.Conv2D(32,kernel_size=(1,1))
        self.conv4 = tf.keras.layers.Conv2D(32,kernel_size=(1,1))
        self.act=tf.keras.layers.Activation('relu')
        self.add = tf.keras.layers.Add()
        self.mul=tf.keras.layers.Multiply()
        self.sigmoid=tf.keras.layers.Activation('relu')
        self.up=tf.keras.layers.UpSampling2D(size=(2,2),interpolation='bilinear')
    def call(self, X):
        # here X will a list of two elements
        #INP, FLOW = X[0], X[1]
        concat=tf.concat([X[0], X[1]],-1)
        out1=self.avg(concat)
        out2=self.conv1(out1)
        X1=self.conv2(out2)

        output1=self.conv3(X1)
        output2=self.act(output1)
        output3=self.conv4(output2)
        output4=self.sigmoid(output3)

        out_mul=self.mul([X1,output4])
        Y=self.add([out_mul,X1])
        f=self.up(Y)


        # implement the context flow as mentioned in the above cell
        return f


class fsm(tf.keras.layers.Layer):
    def __init__(self, name="feature_selection"):
        super().__init__(name=name)
        self.conv1 = tf.keras.layers.Conv2D(32,kernel_size=(3,3),padding='same')
        self.glob=tf.keras.layers.GlobalAveragePooling2D()
        self.conv2 = tf.keras.layers.Conv2D(32,kernel_size=(1,1))
        self.bn=tf.keras.layers.BatchNormalization()
        self.sigmoid=tf.keras.layers.Activation('sigmoid')
        self.mul=tf.keras.layers.Multiply()
        self.up=tf.keras.layers.UpSampling2D(size=(2,2),interpolation='bilinear')
    def call(self, X):
        # implement the FSM modules based on image in the above cells

        out_X=self.conv1(X)
        out2=self.glob(out_X)
        out2=tf.expand_dims(out2,1)
        out2=tf.expand_dims(out2,1)
        out3=self.conv2(out2)
        out4=self.bn(out3)
        out_Y=self.sigmoid(out4)

        out=self.mul([out_X,out_Y])
        FSM_Conv_T=self.up(out)
```

```
        return FSM_Conv_T


class agcn(tf.keras.layers.Layer):
    def __init__(self, name="global_conv_net"):
        super().__init__(name=name)
        self.conv1 = tf.keras.layers.Conv2D(32,kernel_size=(7,1),padding='same')
        self.conv2 = tf.keras.layers.Conv2D(32,kernel_size=(1,7),padding='same')
        self.conv3 = tf.keras.layers.Conv2D(32,kernel_size=(1,7),padding='same')
        self.conv4 = tf.keras.layers.Conv2D(32,kernel_size=(7,1),padding='same')
        self.add1 = tf.keras.layers.Add()
        self.add2 = tf.keras.layers.Add()
        self.conv5 = tf.keras.layers.Conv2D(32,kernel_size=(3,3),padding='same')

    def call(self, X):

        out=self.conv1(X)
        out2=self.conv2(out)

        out_parallel=self.conv3(X)
        out2_parallel=self.conv4(out_parallel)

        out_sum=self.add1([out2,out2_parallel])
        out_x=self.conv5(out_sum)

        X=self.add2([out_x,out_sum])

        return X


# write the complete architecutre
 #self.avg=tf.keras.layers.AveragePooling2D(pool_size=(2, 2), strides=2, padding='valid')
  #      self.conv1 = tf.keras.layers.Conv2D(32,kernel_size=(3,3),padding='same')
class mymodel(tf.keras.Model):
    def __init__(self):
        super().__init__()

        self.s1=stage1()
        self.C1=convolutional_block(kernel=3,  filters=[4,4,8], stride=2, name="conv_block
        self.C2=convolutional_block(kernel=3,  filters=[8,8,16], stride=2, name="conv_bloc
        self.C3=convolutional_block(kernel=3,  filters=[16,16,32], stride=1, name="conv_bl
        self.C4=convolutional_block(kernel=3,  filters=[32,32,64], stride=1, name="conv_bl
        self.I1=identity_block(kernel=3,  filters=[4,4,8], name="identity_block1")
        self.I2=identity_block(kernel=3,  filters=[8,8,16], name="identity_block2")
        self.I3=identity_block(kernel=3,  filters=[8,8,16], name="identity_block3")
        self.I4=identity_block(kernel=3,  filters=[16,16,32], name="identity_block4")
        self.I5=identity_block(kernel=3,  filters=[16,16,32], name="identity_block5")
        self.I6=identity_block(kernel=3,  filters=[16,16,32], name="identity_block6")
        self.I7=identity_block(kernel=3,  filters=[32,32,64], name="identity_block7")
        self.I8=identity_block(kernel=3,  filters=[32,32,64], name="identity_block8")
        self.I9=identity_block(kernel=3,  filters=[32,32,64], name="identity_block9")
        self.I10=identity_block(kernel=3,  filters=[32,32,64], name="identity_block10")

        self.global_f=global_flow(name="global_flow")
        self.context1=context_flow(name="context_flow1")
```

```
        self.context1=context_flow(name="context_flow1")
        self.context2=context_flow(name="context_flow2")
        self.context3=context_flow(name="context_flow3")

        self.add1 = tf.keras.layers.Add()

        self.f_sm=fsm(name="feature_selection")
        self.ag_cn=agcn(name="global_conv_net")

        self.conv = tf.keras.layers.Conv2D(4,kernel_size=(3,3),padding='same')
        self.up=tf.keras.layers.UpSampling2D(size=(4,4),interpolation='bilinear')

        self.softmax=tf.keras.layers.Activation('softmax')

    def call(self, X):

        #stage1
        op_s1=self.s1(X)

        op=self.C1(op_s1)
        op_id=self.I1(op)

        op2=self.C2(op_id)
        op_id2=self.I2(op2)
        op_id3=self.I3(op_id2)

        op3=self.C3(op_id3)
        op_id4=self.I4(op3)
        op_id5=self.I5(op_id4)
        op_id6=self.I6(op_id5)

        op4=self.C4(op_id6)
        op_id7=self.I7(op4)
        op_id8=self.I8(op_id7)
        op_id9=self.I9(op_id8)
        op_c4=self.I10(op_id9)

        op_g=self.global_f(op_c4)
        op_cx1=self.context1([op_c4,op_g])
        op_cx2=self.context2([op_c4,op_cx1])
        op_cx3=self.context3([op_c4,op_cx2])

        op_add=self.add1([op_g,op_cx1,op_cx2,op_cx3])

        op_fsm=self.f_sm(op_add)

        op_agcn=self.ag_cn(op_id)
        op_concat=tf.concat([op_agcn,op_fsm],axis=-1)
        op_conv=self.conv(op_concat)
        op_up=self.up(op_conv)
        Y=self.softmax(op_up)

        return Y
```

```
model=mymodel()


inputs=tf.keras.layers.Input(shape=(224,224,3))


outputs=model(inputs)


outputs
```

```
<tf.Tensor 'mymodel_1/activation_167/truediv:0' shape=(None, 224, 224, 4) dtype=float
```

```
model_canet = tf.keras.Model(inputs = inputs, outputs = outputs)

model_canet.summary()
```

```
Model: "functional_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
mymodel_1 (mymodel)          (None, 224, 224, 4)       277724
=================================================================
Total params: 277,724
Trainable params: 275,116
Non-trainable params: 2,608
_________________________________________________________________
```

```
def dice_coef(y_true, y_pred, epsilon=0.00001):
    """
    Dice = (2*|X & Y|)/ (|X|+ |Y|)
         =  2*sum(|A*B|)/(sum(A^2)+sum(B^2))
    ref: https://arxiv.org/pdf/1606.04797v1.pdf

    """
    axis = (0,1,2)
    dice_numerator = 2. * K.sum(y_true * y_pred, axis=axis) + epsilon
    dice_denominator = K.sum(y_true*y_true, axis=axis) + K.sum(y_pred*y_pred, axis=axis) +
    return K.mean((dice_numerator)/(dice_denominator))

def dice_coef_loss(y_true, y_pred):
    return 1-dice_coef(y_true, y_pred)


learning_rate=0.00000001
model_canet.compile(optimizer=Adam(lr=learning_rate), loss=dice_coef_loss, metrics=[dice_c



# classes for data loading and preprocessing
#to create masks for each patch
classes=[0,1,2,4]
import numpy as np
```

```
from random import sample
class Dataset:
    """Read images, apply augmentation and preprocessing transformations.

    Args:
        images_dir : path to images folder (directories of all sequences)
        masks_dir : path to segmentation masks folder
        classes : values of classes to extract from segmentation mask


    """

    def __init__(
            self,
            flair_paths,t1_paths,t2_paths,t1ce_paths,mask_paths,

            augmentation=None,
            classes=classes,
    ):
        self.images_flair =flair_paths
        self.images_t1 =t1_paths
        self.images_t2 =t2_paths
        self.images_t1ce =t1ce_paths
        self.masks_fps =mask_paths
        self.classes=classes
        self.augmentation = augmentation

    def __getitem__(self, i):

        # read data
        #print(self.images_flair[i])
        #print(i)
        image_f= np.load(str (self.images_flair[i]))
        image_flair=image_f[8:232,8:232]
        image_1 = np.load(str (self.images_t1[i]))
        image_t1=image_1[8:232,8:232]
        image_1ce = np.load(str (self.images_t1ce[i]))
        image_t1ce=image_1ce[8:232,8:232]
        image_2 = np.load(str (self.images_t2[i]))
        image_t2=image_2[8:232,8:232]
        m = np.load(str (self.masks_fps[i]))
        mask=m[8:232,8:232]


        # extract certain classes from mask (e.g. cars)
        masks = [(mask == v) for v in self.classes]
        mask = np.stack(masks, axis=-1).astype('float')
        l=[image_flair,image_t1,image_t1ce,image_t2]
        v=[0,1,2,3]
        x,y,z=sample(v,3)

        image=np.stack((l[x],l[y],l[z]), axis=-1).astype('float')

        return image, mask
```

```
    def __len__(self):
      return len(self.masks_fps)



import pandas as pd
df=pd.DataFrame()
df['flair']=flair_l
df['t1']=t1_l
df['t1ce']=t1ce_l
df['t2']=t2_l
df['mask']=mask_l



min_thresh=10005

l=[]
for index,file in enumerate(df['flair']):
  read_file=np.load(file)
  if np.count_nonzero(read_file)>=min_thresh:
    l.append(index)


new_df=df.iloc[l,:]
new_df.shape

     (2890, 5)


from sklearn.model_selection import train_test_split
X=new_df.drop(["mask"],axis=1)
y=new_df["mask"]


X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=10)


train_data_canet= Dataset(
    flair_paths=list(X_train["flair"]),t1_paths=list(X_train["t1"]),t1ce_paths=list(X_trai
    classes=classes

)

test_data_canet= Dataset(
    flair_paths=list(X_test["flair"]),t1_paths=list(X_test["t1"]),t1ce_paths=list(X_test["
    classes=classes

)


train_dataloader_canet = Dataloder(train_data_canet, batch_size=4, shuffle=True)
valid_dataloader_canet = Dataloder(test_data_canet, batch_size=1, shuffle=True)

# check shapes for errors
assert train_dataloader_canet[0][0].shape == (4, 224, 224, 3)
```

```
assert train_dataloader_canet[0][1].shape == (4, 224, 224, 4)


# define callbacks for learning rate scheduling and best checkpoints saving
checkpoint_reducelr = [
    tf.keras.callbacks.ModelCheckpoint('/content/drive/My Drive/cannet2/weights-{epoch:02d
    tf.keras.callbacks.ReduceLROnPlateau(
    monitor='val_iou_score', factor=0.8, patience=3, verbose=0, mode='max',
    min_delta=0.0001, cooldown=0, min_lr=0
),
]


%load_ext tensorboard
import datetime

# Clear any logs from previous runs
!rm -rf ./logs/

     The tensorboard extension is already loaded. To reload it, use:
       %reload_ext tensorboard


log_dir="logs/fit/basic-canet" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
#tensorboard_callback = keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, writ


model_canet.load_weights("/content/drive/My Drive/cannet2/model_cannet500-1.h5")


history = model_canet.fit_generator(train_dataloader, steps_per_epoch=len(train_dataloader
```

```
Train for 108 steps, validate for 385 steps
Epoch 1/10
108/108 [==============================] - 26s 236ms/step - loss: 0.8271 - iou_score: 0.2454 - val_loss: 0.3888 - val_iou_score: 0.6855
Epoch 2/10
108/108 [==============================] - 26s 239ms/step - loss: 0.8271 - iou_score: 0.2454 - val_loss: 0.3888 - val_iou_score: 0.6855
Epoch 3/10
108/108 [==============================] - 26s 237ms/step - loss: 0.8271 - iou_score: 0.2454 - val_loss: 0.3888 - val_iou_score: 0.6855
Epoch 4/10
108/108 [==============================] - 26s 238ms/step - loss: 0.8271 - iou_score: 0.2454 - val_loss: 0.3888 - val_iou_score: 0.6855
Epoch 5/10
108/108 [==============================] - 26s 239ms/step - loss: 0.8271 - iou_score: 0.2454 - val_loss: 0.3888 - val_iou_score: 0.6855
Epoch 6/10
108/108 [==============================] - 26s 238ms/step - loss: 0.8271 - iou_score: 0.2454 - val_loss: 0.3888 - val_iou_score: 0.6855
Epoch 7/10
108/108 [==============================] - 26s 237ms/step - loss: 0.8271 - iou_score: 0.2454 - val_loss: 0.3888 - val_iou_score: 0.6855
Epoch 8/10
108/108 [==============================] - 25s 236ms/step - loss: 0.8271 - iou_score: 0.2454 - val_loss: 0.3888 - val_iou_score: 0.6855
Epoch 9/10
108/108 [==============================] - 26s 237ms/step - loss: 0.8271 - iou_score: 0.2454 - val_loss: 0.3888 - val_iou_score: 0.6855
Epoch 10/10
108/108 [==============================] - 26s 237ms/step - loss: 0.8271 - iou_score: 0.2454 - val_loss: 0.3888 - val_iou_score: 0.6855
```

```
#here by using the basic cannet we can observe there is no change in loss,clearly indicati
#so next experiment i've used is by removing the identity blocks so to reduce some layers
```

CANNET MODEL-WITHOUT IDENTITY BLOCKS

```
#cannet model-Removed identity blocks

class mymodel(tf.keras.Model):
```

```
class MyModel(tf.keras.Model):
    def __init__(self):
        super().__init__()

        self.s1=stage1()
        self.C1=convolutional_block(kernel=3,  filters=[4,4,8], stride=2, name="conv_block
        self.C2=convolutional_block(kernel=3,  filters=[8,8,16], stride=2, name="conv_bloc
        self.C3=convolutional_block(kernel=3,  filters=[16,16,32], stride=1, name="conv_bl
        self.C4=convolutional_block(kernel=3,  filters=[32,32,64], stride=1, name="conv_bl
        self.I1=identity_block(kernel=3,  filters=[4,4,8], name="identity_block1")
        self.I2=identity_block(kernel=3,  filters=[8,8,16], name="identity_block2")
        #self.I3=identity_block(kernel=3,  filters=[32,32,64], name="identity_block3")
        self.I4=identity_block(kernel=3,  filters=[16,16,32], name="identity_block4")
        #self.I5=identity_block(kernel=3,  filters=[64,64,128], name="identity_block5")
        #self.I6=identity_block(kernel=3,  filters=[64,64,128], name="identity_block6")
        self.I7=identity_block(kernel=3,  filters=[32,32,64], name="identity_block7")
        #self.I8=identity_block(kernel=3,  filters=[128,128,256], name="identity_block8")
        #self.I9=identity_block(kernel=3,  filters=[128,128,256], name="identity_block9")
        self.I10=identity_block(kernel=3,  filters=[32,32,64], name="identity_block10")

        self.global_f=global_flow(name="global_flow")
        self.context1=context_flow(name="context_flow1")
        self.context2=context_flow(name="context_flow2")
        self.context3=context_flow(name="context_flow3")

        self.add1 = tf.keras.layers.Add()

        self.f_sm=fsm(name="feature_selection")
        self.ag_cn=agcn(name="global_conv_net")

        self.conv = tf.keras.layers.Conv2D(4,kernel_size=(3,3),padding='same')
        self.up=tf.keras.layers.UpSampling2D(size=(4,4),interpolation='bilinear')

        self.softmax=tf.keras.layers.Activation('softmax')

    def call(self, X):

        #stage1
        op_s1=self.s1(X)

        op=self.C1(op_s1)
        op_id=self.I1(op)

        op2=self.C2(op_id)
        op_id2=self.I2(op2)
        #op_id3=self.I3(op_id2)

        op3=self.C3(op_id2)
        op_id4=self.I4(op3)
        #op_id5=self.I5(op_id4)
        #op_id6=self.I6(op_id5)

        op4=self.C4(op_id4)
        op_id7=self.I7(op4)
        #op_id8=self.I8(op_id7)
        #op_id9=self.I9(op_id8)
```

```
        #op_id9=self.I9(op_id8)
        op_c4=self.I10(op_id7)

        op_g=self.global_f(op_c4)
        op_cx1=self.context1([op_c4,op_g])
        op_cx2=self.context2([op_c4,op_cx1])
        op_cx3=self.context3([op_c4,op_cx2])

        op_add=self.add1([op_g,op_cx1,op_cx2,op_cx3])

        op_fsm=self.f_sm(op_add)

        op_agcn=self.ag_cn(op_id)
        op_concat=tf.concat([op_agcn,op_fsm],axis=-1)
        op_conv=self.conv(op_concat)
        op_up=self.up(op_conv)
        Y=self.softmax(op_up)

        return Y



model=mymodel()


inputs=tf.keras.layers.Input(shape=(224,224,3))


outputs=model(inputs)


outputs
```

```
<tf.Tensor 'mymodel_2/activation_213/truediv:0' shape=(None, 224, 224, 4) dtype=float
```

```
model_canet_no_id = tf.keras.Model(inputs = inputs, outputs = outputs)

model_canet_no_id.summary()
```

```
Model: "functional_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
mymodel_2 (mymodel)          (None, 224, 224, 4)       241532
=================================================================
Total params: 241,532
Trainable params: 239,756
Non-trainable params: 1,776
_________________________________________________________________
```

```
learning_rate=0.00000001
model_canet_no_id.compile(optimizer=Adam(lr=learning_rate), loss=dice_coef_loss, metrics=[
```

```
train_dataloader_canet_no_id = Dataloder(train_data_canet, batch_size=4, shuffle=True)
valid_dataloader_canet_no_id = Dataloder(test_data_canet, batch_size=1, shuffle=True)

# check shapes for errors
assert train_dataloader_canet_no_id[0][0].shape == (4, 224, 224, 3)
assert train_dataloader_canet_no_id[0][1].shape == (4, 224, 224, 4)


# define callbacks for learning rate scheduling and best checkpoints saving
checkpoint_reducelr = [
    tf.keras.callbacks.ModelCheckpoint('/content/drive/My Drive/cannet-weights/weights-{ep
    tf.keras.callbacks.ReduceLROnPlateau(
    monitor='val_loss', factor=0.08, patience=3, verbose=0, mode='min',
    min_delta=0.0001, cooldown=0, min_lr=0
),tf.keras.callbacks.EarlyStopping( monitor='val_loss', min_delta=0.0001, patience=2, verb
]


#model_canet_no_id.load_weights("/content/drive/My Drive/cannet-weights")


%load_ext tensorboard
import datetime

# Clear any logs from previous runs
!rm -rf ./logs/

     The tensorboard extension is already loaded. To reload it, use:
       %reload_ext tensorboard


log_dir="logs/fit/canet-no-id" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
#tensorboard_callback = keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, writ


history = model_canet_no_id.fit_generator(train_dataloader, steps_per_epoch=len(train_data
```

```
Epoch 00220: val_iou_score did not improve from 0.59727
578/578 [==============================] - 26s 44ms/step - loss: 0.2750 - iou_score: 0.6548 - val_loss: 0.3909 - val_iou_score: 0.5897
Epoch 221/500
576/578 [============================>.] - ETA: 0s - loss: 0.2770 - iou_score: 0.6530
Epoch 00221: val_iou_score did not improve from 0.59727
578/578 [==============================] - 26s 45ms/step - loss: 0.2764 - iou_score: 0.6535 - val_loss: 0.3918 - val_iou_score: 0.5892
Epoch 222/500
577/578 [============================>.] - ETA: 0s - loss: 0.2720 - iou_score: 0.6570
Epoch 00222: val_iou_score did not improve from 0.59727
578/578 [==============================] - 26s 44ms/step - loss: 0.2720 - iou_score: 0.6570 - val_loss: 0.3880 - val_iou_score: 0.5912
Epoch 223/500
576/578 [============================>.] - ETA: 0s - loss: 0.2744 - iou_score: 0.6557
Epoch 00223: val_iou_score did not improve from 0.59727
578/578 [==============================] - 25s 44ms/step - loss: 0.2747 - iou_score: 0.6554 - val_loss: 0.3924 - val_iou_score: 0.5888
Epoch 224/500
577/578 [============================>.] - ETA: 0s - loss: 0.2772 - iou_score: 0.6528
Epoch 00224: val_iou_score did not improve from 0.59727
578/578 [==============================] - 25s 44ms/step - loss: 0.2774 - iou_score: 0.6526 - val_loss: 0.3913 - val_iou_score: 0.5888
Epoch 225/500
577/578 [============================>.] - ETA: 0s - loss: 0.2783 - iou_score: 0.6522
Epoch 00225: val_iou_score did not improve from 0.59727
578/578 [==============================] - 25s 44ms/step - loss: 0.2781 - iou_score: 0.6525 - val_loss: 0.3926 - val_iou_score: 0.5882
Epoch 226/500
576/578 [============================>.] - ETA: 0s - loss: 0.2747 - iou_score: 0.6551
Epoch 00226: val_iou_score did not improve from 0.59727
578/578 [==============================] - 26s 44ms/step - loss: 0.2750 - iou_score: 0.6547 - val_loss: 0.3888 - val_iou_score: 0.5908
Epoch 227/500
577/578 [============================>.] - ETA: 0s - loss: 0.2710 - iou_score: 0.6591
Epoch 00227: val_iou_score did not improve from 0.59727
578/578 [==============================] - 26s 44ms/step - loss: 0.2711 - iou_score: 0.6590 - val_loss: 0.3901 - val_iou_score: 0.5900
```

```
#validation dice loss got stuck at 0.3900 for around 15 epochs the model is not learning m
#all the parameters are not able to change accordingly so that loss is not converging even
```

EXPERIMENTATION OF BACKBONE NETWORKS

RESNET152

```
#Here resnet152 network is used for classificaion task so to check if the model performanc
#if the performance is good that represents context information of image is well captured


IMAGE_SIZE=[224,224]
res=tf.keras.applications.ResNet152(
    include_top=False,
    weights="imagenet",
    input_tensor=None,
    input_shape=IMAGE_SIZE + [3],
    pooling=None,
    classes=2
)#without fc layers

from tensorflow.keras.layers import Dense,Input,Conv2D,MaxPool2D,Activation,Dropout,Flatte
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
import tensorflow as tf
import matplotlib.pyplot as plt
import random as rn

for layer in res.layers:
  layer.trainable = False

#Conv Layer
Conv = Conv2D(filters=256,kernel_size=(3,3),strides=(1,1),padding='same',
              activation='relu',kernel_initializer=tf.keras.initializers.he_normal(seed=32
#MaxPool Layer
Pool = MaxPool2D(pool_size=(2,2),strides=(1,1),padding='valid',name='Pool')(Conv)


#Flatten
flatten = Flatten(name='Flatten')(Pool)

#FC1 layer
FC1 = Dense(units=128,activation='relu',kernel_initializer=tf.keras.initializers.glorot_no

#FC2 layer
FC2 = Dense(units=64,activation='relu',kernel_initializer=tf.keras.initializers.glorot_nor

#FC3 layer
FC3 = Dense(units=32,activation='relu',kernel_initializer=tf.keras.initializers.glorot_nor
```

```
#FC4 layer
#FC4 = Dense(units=16,activation='relu',kernel_initializer=tf.keras.initializers.glorot_no

#output layer
Out = Dense(units=1,activation='sigmoid',kernel_initializer=tf.keras.initializers.glorot_n

#Creating a model
model_res = Model(inputs=res.input,outputs=Out)


model_res.compile(optimizer=tf.keras.optimizers.Adam(lr=0.00001),loss='binary_crossentropy
model_res.summary()
```

```
Model: "functional_15"
__________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==========================================================================================
input_7 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________
conv1_pad (ZeroPadding2D)       (None, 230, 230, 3)  0           input_7[0][0]
__________________________________________________________________________________________
conv1_conv (Conv2D)             (None, 112, 112, 64) 9472        conv1_pad[0][0]
__________________________________________________________________________________________
conv1_bn (BatchNormalization)   (None, 112, 112, 64) 256         conv1_conv[0][0]
__________________________________________________________________________________________
conv1_relu (Activation)         (None, 112, 112, 64) 0           conv1_bn[0][0]
__________________________________________________________________________________________
pool1_pad (ZeroPadding2D)       (None, 114, 114, 64) 0           conv1_relu[0][0]
__________________________________________________________________________________________
pool1_pool (MaxPooling2D)       (None, 56, 56, 64)   0           pool1_pad[0][0]
__________________________________________________________________________________________
conv2_block1_1_conv (Conv2D)    (None, 56, 56, 64)   4160        pool1_pool[0][0]
__________________________________________________________________________________________
conv2_block1_1_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block1_1_co
__________________________________________________________________________________________
conv2_block1_1_relu (Activation (None, 56, 56, 64)   0           conv2_block1_1_bn
__________________________________________________________________________________________
conv2_block1_2_conv (Conv2D)    (None, 56, 56, 64)   36928       conv2_block1_1_re
__________________________________________________________________________________________
conv2_block1_2_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block1_2_co
__________________________________________________________________________________________
conv2_block1_2_relu (Activation (None, 56, 56, 64)   0           conv2_block1_2_bn
__________________________________________________________________________________________
conv2_block1_0_conv (Conv2D)    (None, 56, 56, 256)  16640       pool1_pool[0][0]
__________________________________________________________________________________________
conv2_block1_3_conv (Conv2D)    (None, 56, 56, 256)  16640       conv2_block1_2_re
__________________________________________________________________________________________
conv2_block1_0_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block1_0_co
__________________________________________________________________________________________
conv2_block1_3_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block1_3_co
__________________________________________________________________________________________
conv2_block1_add (Add)          (None, 56, 56, 256)  0           conv2_block1_0_bn
                                                                 conv2_block1_3_bn
__________________________________________________________________________________________
conv2_block1_out (Activation)   (None, 56, 56, 256)  0           conv2_block1_add[(
__________________________________________________________________________________________
conv2_block2_1_conv (Conv2D)    (None, 56, 56, 64)   16448       conv2_block1_out[(
__________________________________________________________________________________________
conv2_block2_1_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block2_1_co
__________________________________________________________________________________________
```

```
conv2_block2_1_relu (Activation (None, 56, 56, 64)   0          conv2_block2_1_bn
__________________________________________________________________________________
conv2_block2_2_conv (Conv2D)    (None, 56, 56, 64)   36928      conv2_block2_1_re
__________________________________________________________________________________
conv2_block2_2_bn (BatchNormali (None, 56, 56, 64)   256        conv2_block2_2_co
__________________________________________________________________________________
conv2_block2_2_relu (Activation (None, 56, 56, 64)   0          conv2_block2_2_bn
__________________________________________________________________________________
conv2_block2_3_conv (Conv2D)    (None, 56, 56, 256)  16640      conv2_block2_2_re
__________________________________________________________________________________
conv2_block2_3_bn (BatchNormali (None, 56, 56, 256)  1024       conv2_block2_3_co
```

```
history = model_res.fit_generator(train_dataloader, steps_per_epoch=len(train_dataloader),
```

```
Epoch 18/500
769/769 [==============================] - 58s 75ms/step - loss: 0.5073 - accuracy: 0.7383 - val_loss: 0.5407 - val_accuracy: 0.7156
Epoch 19/500
769/769 [==============================] - 58s 75ms/step - loss: 0.4977 - accuracy: 0.7630 - val_loss: 0.5354 - val_accuracy: 0.7312
Epoch 20/500
769/769 [==============================] - 58s 75ms/step - loss: 0.4964 - accuracy: 0.7458 - val_loss: 0.5129 - val_accuracy: 0.7455
Epoch 21/500
769/769 [==============================] - 58s 75ms/step - loss: 0.4918 - accuracy: 0.7633 - val_loss: 0.5313 - val_accuracy: 0.7364
Epoch 22/500
769/769 [==============================] - 57s 74ms/step - loss: 0.4848 - accuracy: 0.7568 - val_loss: 0.5365 - val_accuracy: 0.7234
Epoch 23/500
769/769 [==============================] - 56s 73ms/step - loss: 0.4816 - accuracy: 0.7679 - val_loss: 0.5141 - val_accuracy: 0.7377
Epoch 24/500
769/769 [==============================] - 56s 73ms/step - loss: 0.4735 - accuracy: 0.7653 - val_loss: 0.4949 - val_accuracy: 0.7571
Epoch 25/500
769/769 [==============================] - 56s 73ms/step - loss: 0.4707 - accuracy: 0.7633 - val_loss: 0.4993 - val_accuracy: 0.7506
Epoch 26/500
769/769 [==============================] - 56s 73ms/step - loss: 0.4681 - accuracy: 0.7796 - val_loss: 0.4922 - val_accuracy: 0.7494
```



```
#model is converging well,loss is reducing slowly but there is convergence we can try this
```

VGG16

```
from tensorflow.keras.layers import Dense,Input,Conv2D,MaxPool2D,Activation,Dropout,Flatte
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
import tensorflow as tf
import matplotlib.pyplot as plt
import random as rn


IMAGE_SIZE=[224,224]
vg=tf.keras.applications.VGG16(
    include_top=False,
    weights="imagenet",
    input_tensor=None,
    input_shape=IMAGE_SIZE + [3],
    pooling=None,
    classes=2
)#without fc layers
```

```
for layer in vg.layers:
  layer.trainable = False

#Conv Layer
Conv = Conv2D(filters=256,kernel_size=(3,3),strides=(1,1),padding='same',
              activation='relu',kernel_initializer=tf.keras.initializers.he_normal(seed=32
#MaxPool Layer
Pool = MaxPool2D(pool_size=(2,2),strides=(1,1),padding='valid',name='Pool')(Conv)


#Flatten
flatten = Flatten(name='Flatten')(Pool)

#FC1 layer
FC1 = Dense(units=128,activation='relu',kernel_initializer=tf.keras.initializers.glorot_no

#FC2 layer
FC2 = Dense(units=64,activation='relu',kernel_initializer=tf.keras.initializers.glorot_nor

#FC3 layer
FC3 = Dense(units=32,activation='relu',kernel_initializer=tf.keras.initializers.glorot_nor

#FC4 layer
#FC4 = Dense(units=16,activation='relu',kernel_initializer=tf.keras.initializers.glorot_no

#output layer
Out = Dense(units=1,activation='sigmoid',kernel_initializer=tf.keras.initializers.glorot_n

#Creating a model
model_vg = Model(inputs=res.input,outputs=Out)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vg
58892288/58889256 [==============================] - 0s 0us/step
```

```
model_vg.compile(optimizer=tf.keras.optimizers.Adam(lr=0.00001),loss='binary_crossentropy'
model_vg.summary()
```

```
Model: "functional_17"
_________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================
input_7 (InputLayer)         [(None, 224, 224, 3) 0
_________________________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, 230, 230, 3)  0           input_7[0][0]
_________________________________________________________________________________
conv1_conv (Conv2D)          (None, 112, 112, 64) 9472        conv1_pad[0][0]
_________________________________________________________________________________
conv1_bn (BatchNormalization)(None, 112, 112, 64) 256         conv1_conv[0][0]
_________________________________________________________________________________
conv1_relu (Activation)      (None, 112, 112, 64) 0           conv1_bn[0][0]
_________________________________________________________________________________
pool1_pad (ZeroPadding2D)    (None, 114, 114, 64) 0           conv1_relu[0][0]
_________________________________________________________________________________
pool1_pool (MaxPooling2D)    (None, 56, 56, 64)   0           pool1_pad[0][0]
_________________________________________________________________________________
conv2_block1_1_conv (Conv2D) (None, 56, 56, 64)   4160        pool1_pool[0][0]
```

```
conv2_block1_1_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block1_1_co
conv2_block1_1_relu (Activation (None, 56, 56, 64)   0           conv2_block1_1_bn
conv2_block1_2_conv (Conv2D)    (None, 56, 56, 64)   36928       conv2_block1_1_re
conv2_block1_2_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block1_2_co
conv2_block1_2_relu (Activation (None, 56, 56, 64)   0           conv2_block1_2_bn
conv2_block1_0_conv (Conv2D)    (None, 56, 56, 256)  16640       pool1_pool[0][0]
conv2_block1_3_conv (Conv2D)    (None, 56, 56, 256)  16640       conv2_block1_2_re
conv2_block1_0_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block1_0_co
conv2_block1_3_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block1_3_co
conv2_block1_add (Add)          (None, 56, 56, 256)  0           conv2_block1_0_bn
                                                                 conv2_block1_3_bn
conv2_block1_out (Activation)   (None, 56, 56, 256)  0           conv2_block1_add[(
conv2_block2_1_conv (Conv2D)    (None, 56, 56, 64)   16448       conv2_block1_out[(
conv2_block2_1_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block2_1_co
conv2_block2_1_relu (Activation (None, 56, 56, 64)   0           conv2_block2_1_bn
conv2_block2_2_conv (Conv2D)    (None, 56, 56, 64)   36928       conv2_block2_1_re
conv2_block2_2_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block2_2_co
conv2_block2_2_relu (Activation (None, 56, 56, 64)   0           conv2_block2_2_bn
conv2_block2_3_conv (Conv2D)    (None, 56, 56, 256)  16640       conv2_block2_2_re
conv2_block2_3_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block2_3_co
```

```
history = model_vg.fit_generator(train_dataloader, steps_per_epoch=len(train_dataloader),e
```

```
Epoch 1/500
578/578 [==============================] - 38s 66ms/step - loss: 0.5528 - accuracy: 0.7102 - val_loss: 0.5103 - val_accuracy: 0.7370
Epoch 2/500
578/578 [==============================] - 21s 37ms/step - loss: 0.4846 - accuracy: 0.7535 - val_loss: 0.4449 - val_accuracy: 0.7699
Epoch 3/500
578/578 [==============================] - 22s 37ms/step - loss: 0.4138 - accuracy: 0.8067 - val_loss: 0.4047 - val_accuracy: 0.8426
Epoch 4/500
578/578 [==============================] - 22s 38ms/step - loss: 0.3683 - accuracy: 0.8378 - val_loss: 0.3746 - val_accuracy: 0.8218
Epoch 5/500
```

#model is converging very quickly better than resnet so vgg pretrained layer weights captu
#so we can add vgg as backbone without training

```
578/578 [==============================] - 22s 38ms/step - loss: 0.2091 - accuracy: 0.9425 - val_loss: 0.2785 - val_accuracy: 0.8581
```

CANNET WITH VGG BACKBONE

```
578/578 [==============================] - 22s 37ms/step - loss: 0.1358 - accuracy: 0.9697 - val_loss: 0.2189 - val_accuracy: 0.9014
```

#Replacining Conv blocks and identity blocks fromm cannet architecture with vgg retrained
#global blocks and context flow modules and using augmentation techinques

```
578/578 [==============================] - 22s 37ms/step - loss: 0.0794 - accuracy: 0.9862 - val_loss: 0.1949 - val_accuracy: 0.9221
```



```
class global_flow(tf.keras.layers.Layer):
    def __init__(self, name="global_flow"):
        super().__init__(name=name)
        self.glob=tf.keras.layers.GlobalAveragePooling2D()
        self.conv=tf.keras.layers.Conv2D(64,kernel_size=(1,1))
        self.bn=tf.keras.layers.BatchNormalization()
        self.act=tf.keras.layers.Activation('relu')
        self.up=tf.keras.layers.UpSampling2D(size=(7,7),interpolation='bilinear')

    def call(self, X):
        # implement the global flow operatiom
        out=self.glob(X)
        out=tf.expand_dims(out,1)
        out=tf.expand_dims(out,1)
        out1=self.bn(out)
        out2=self.act(out1)
        out3=self.conv(out2)
        X=self.up(out3)


        return X


class context_flow(tf.keras.layers.Layer):
    def __init__(self, name="context_flow"):
        super().__init__(name=name)

        self.avg=tf.keras.layers.AveragePooling2D(pool_size=(2, 2), strides=1, padding='sa
        self.conv1 = tf.keras.layers.Conv2D(64,kernel_size=(3,3),padding='same')
        self.conv2 = tf.keras.layers.Conv2D(64,kernel_size=(3,3),padding='same')
        self.conv3 = tf.keras.layers.Conv2D(64,kernel_size=(1,1))
        self.conv4 = tf.keras.layers.Conv2D(64,kernel_size=(1,1))
        self.act=tf.keras.layers.Activation('relu')
        self.add = tf.keras.layers.Add()
        self.mul=tf.keras.layers.Multiply()
        self.sigmoid=tf.keras.layers.Activation('relu')
        #self.up=tf.keras.layers.UpSampling2D(size=(2,2),interpolation='bilinear')
    def call(self, X):
        # here X will a list of two elements
```

```
        #INP, FLOW = X[0], X[1]
        concat=tf.concat([X[0], X[1]],-1)

        #print("concat",concat.shape)
        out1=self.avg(concat)
        #print("out1",out1.shape)
        out2=self.conv1(out1)
        X1=self.conv2(out2)

        output1=self.conv3(X1)
        output2=self.act(output1)
        output3=self.conv4(output2)
        output4=self.sigmoid(output3)
        #print("x1",X1.shape)
        #print("oup4",output4.shape)
        out_mul=self.mul([X1,output4])
        Y=self.add([out_mul,X1])
        #f=self.up(Y)
        #print("up",f.shape)


        # implement the context flow as mentioned in the above cell
        return Y


class fsm(tf.keras.layers.Layer):
    def __init__(self, name="feature_selection"):
        super().__init__(name=name)
        self.conv1 = tf.keras.layers.Conv2D(64,kernel_size=(3,3),padding='same')
        self.glob=tf.keras.layers.GlobalAveragePooling2D()
        self.conv2 = tf.keras.layers.Conv2D(64,kernel_size=(1,1))
        self.bn=tf.keras.layers.BatchNormalization()
        self.sigmoid=tf.keras.layers.Activation('sigmoid')
        self.mul=tf.keras.layers.Multiply()
        self.up=tf.keras.layers.UpSampling2D(size=(16,16),interpolation='bilinear')
    def call(self, X):
        # implement the FSM modules based on image in the above cells

        out_X=self.conv1(X)
        out2=self.glob(out_X)
        out2=tf.expand_dims(out2,1)
        out2=tf.expand_dims(out2,1)
        out3=self.conv2(out2)
        out4=self.bn(out3)
        out_Y=self.sigmoid(out4)

        out=self.mul([out_X,out_Y])
        FSM_Conv_T=self.up(out)

        return FSM_Conv_T


class agcn(tf.keras.layers.Layer):
    def __init__(self, name="global_conv_net"):
        super().__init__(name=name)
        self.conv1 = tf.keras.layers.Conv2D(64,kernel_size=(7,1),padding='same')
```

```
        self.conv2 = tf.keras.layers.Conv2D(64,kernel_size=(1,7),padding='same')
        self.conv3 = tf.keras.layers.Conv2D(64,kernel_size=(1,7),padding='same')
        self.conv4 = tf.keras.layers.Conv2D(64,kernel_size=(7,1),padding='same')
        self.add1 = tf.keras.layers.Add()
        self.add2 = tf.keras.layers.Add()
        self.conv5 = tf.keras.layers.Conv2D(64,kernel_size=(3,3),padding='same')

    def call(self, X):

        out=self.conv1(X)
        out2=self.conv2(out)

        out_parallel=self.conv3(X)
        out2_parallel=self.conv4(out_parallel)

        out_sum=self.add1([out2,out2_parallel])
        out_x=self.conv5(out_sum)

        X=self.add2([out_x,out_sum])

        return X



        IMAGE_SIZE=[224,224]
        vgg=VGG16(include_top=False,weights="imagenet",input_shape=IMAGE_SIZE + [3]) #with

        for layer in vgg.layers:
          layer.trainable = False

        #train layer 14
        vgg.get_layer('block5_conv3').trainable=True

        c1 = vgg.get_layer('block2_conv2').output
        vgg_op=vgg.get_layer('block5_conv3').output

        op_g=global_flow(name="global_flow")(vgg_op)

        op_cx1=context_flow(name="context_flow1")([vgg.output,op_g])
        op_cx2=context_flow(name="context_flow2")([vgg.output,op_cx1])
        op_cx3=context_flow(name="context_flow3")([vgg.output,op_cx2])

        op_add=tf.keras.layers.Add()([op_g,op_cx1,op_cx2,op_cx3])

        op_fsm=fsm(name="feature_selection")(op_add)



        op_agcn=agcn(name="global_conv_net")(c1)

        op_concat=tf.concat([op_agcn,op_fsm],axis=-1)
        op_conv=tf.keras.layers.Conv2D(4,kernel_size=(3,3),padding='same')(op_concat)
        op_up=tf.keras.layers.UpSampling2D(size=(2,2),interpolation='bilinear')(op_conv)
        Y_vgg_aug=tf.keras.layers.Activation('softmax')(op_up)
```

```
Y_vgg_aug.shape
```

```
TensorShape([None, 224, 224, 4])
```

```
model_vgg_aug = tf.keras.Model(inputs = vgg.input,outputs=Y_vgg_aug)
```

```
model_vgg_aug.summary()
```

```
Model: "functional_19"
```

Layer (type)	Output Shape	Param #	Connected to
input_10 (InputLayer)	[(None, 224, 224, 3)	0	
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792	input_10[0][0]
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928	block1_conv1[0][0
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0	block1_conv2[0][0
block2_conv1 (Conv2D)	(None, 112, 112, 128	73856	block1_pool[0][0]
block2_conv2 (Conv2D)	(None, 112, 112, 128	147584	block2_conv1[0][0
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0	block2_conv2[0][0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168	block2_pool[0][0]
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080	block3_conv1[0][0
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080	block3_conv2[0][0
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0	block3_conv3[0][0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160	block3_pool[0][0]
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808	block4_conv1[0][0
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808	block4_conv2[0][0
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0	block4_conv3[0][0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808	block4_pool[0][0]
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808	block5_conv1[0][0
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808	block5_conv2[0][0
global_flow (global_flow)	(None, 7, 7, 64)	34880	block5_conv3[0][0
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0	block5_conv3[0][0
context_flow1 (context_flow)	(None, 7, 7, 64)	377088	block5_pool[0][0] global_flow[0][0]
context_flow2 (context_flow)	(None, 7, 7, 64)	377088	block5_pool[0][0] context_flow1[0][(
context_flow3 (context_flow)	(None, 7, 7, 64)	377088	block5_pool[0][0]

```
                                                                 context_flow2[0][(
__________________________________________________________________________________________________
add_59 (Add)                    (None, 7, 7, 64)     0           global_flow[0][0]
                                                                 context_flow1[0][(
                                                                 context_flow2[0][(
                                                                 context_flow3[0][(
__________________________________________________________________________________________________
```

```
learning_rate=0.0000001
model_vgg_aug.compile(optimizer=tf.keras.optimizers.Adam(lr=learning_rate), loss=dice_coef
```

```
tf.compat.v1.enable_eager_execution()
```

```
import pandas as pd
df=pd.DataFrame()
df['flair']=flair_l
df['t1']=t1_l
df['t1ce']=t1ce_l
df['t2']=t2_l
df['mask']=mask_l
```

```
min_thresh=10005

l=[]
for index,file in enumerate(df['flair']):
  read_file=np.load(file)
  if np.count_nonzero(read_file)>=min_thresh:
    l.append(index)
```

```
new_df=df.iloc[l,:]
```

```
import imgaug.augmenters as iaa
#import imgaug

aug1 = iaa.Flipud(1)
aug2 = iaa.DirectedEdgeDetect(alpha=(0.5))
```

```
# classes for data loading and preprocessing
#to create masks for each patch
classes=[0,1,2,4]
import numpy as np

from random import sample
class Dataset:
    """Read images, apply augmentation and preprocessing transformations.

    Args:
        images_dir : path to images folder (directories of all sequences)
        masks_dir : path to segmentation masks folder
```

```
    masks_dir : path to segmentation masks folder
    classes : values of classes to extract from segmentation mask


"""

def __init__(
        self,
        flair_paths,t1_paths,t1ce_paths,mask_paths,

        augmentation=None,
        classes=classes,
):
    self.images_flair =flair_paths
    self.images_t1 =t1_paths
    #self.images_t2 =t2_paths
    self.images_t1ce =t1ce_paths
    self.masks_fps =mask_paths
    self.classes=classes
    self.augmentation = augmentation

def __getitem__(self, i):

    # read data
    #print(self.images_flair[i])
    #print(i)
    image_f= np.load(str (self.images_flair[i]))
    image_flair=image_f[8:232,8:232]
    image_1 = np.load(str (self.images_t1[i]))
    image_t1=image_1[8:232,8:232]
    image_1ce = np.load(str (self.images_t1ce[i]))
    image_t1ce=image_1ce[8:232,8:232]
    #image_2 = np.load(str (self.images_t2[i]))
    #image_t2=image_2[8:232,8:232]
    m = np.load(str (self.masks_fps[i]))
    mask=m[8:232,8:232]
    #mask=self.masks_fps[i]


    # extract certain classes from mask (e.g. cars)
    masks = [(mask == v) for v in self.classes]
    mask = np.stack(masks, axis=-1).astype('float')
    #l=[image_flair,image_t1,image_t1ce]
    #v=[0,1,2,3]
    #x,y,z=sample(v,3)

    image=np.stack((image_flair,image_t1,image_t1ce), axis=-1).astype('float')

    # apply augmentations

    image = aug1.augment_image(image)
    mask = aug1.augment_image(mask)

    image = aug2.augment_image(image)

    return image,mask
```

```
        return image,mask


    def __len__(self):
      return len(self.masks_fps)



mask_class=[]
for i in new_df['mask']:
  if np.load(i).any()>0:
    mask_class.append(1)
  else:
    mask_class.append(0)

new_df['mask_class']=mask_class
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:8: SettingWithCopyWarnir
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
```

```
from sklearn.model_selection import train_test_split
X=new_df.drop(["mask","t2"],axis=1)
y=new_df["mask"]


X_train,X_test,y_train,y_test=train_test_split(X,y,stratify=mask_class,test_size=0.2,rando


train_data_vgg_aug= Dataset(
    flair_paths=list(X_train["flair"]),t1_paths=list(X_train["t1"]),t1ce_paths=list(X_trai
    classes=classes

)

test_data_vgg_aug= Dataset(
    flair_paths=list(X_test["flair"]),t1_paths=list(X_test["t1"]),t1ce_paths=list(X_test["
    classes=classes

)


train_dataloader_vgg_aug = Dataloder(train_data_vgg_aug, batch_size=4, shuffle=True)
valid_dataloader_vgg_aug = Dataloder(test_data_vgg_aug, batch_size=1, shuffle=True)

# check shapes for errors
assert train_dataloader_vgg_aug[0][0].shape == (4, 224, 224, 3)
assert train_dataloader_vgg_aug[0][1].shape == (4, 224, 224, 4)


# define callbacks for learning rate scheduling and best checkpoints saving
checkpoint_reducelr = [
    tf.keras.callbacks.ModelCheckpoint('/content/drive/My Drive/cannet-vgg-14train-edge-lg
]
```

]

#model_vgg_aug.load_weights("/content/drive/My Drive/cannet-vgg-14train-edge-lgg/weights-1

```
Epoch 328/1000
578/578 [==============================] - ETA: 0s - loss: 0.3537 - dice_coef: 0.6463
Epoch 00328: val_dice_coef did not improve from 0.59458
578/578 [==============================] - 93s 161ms/step - loss: 0.3537 - dice_coef: 0.6463 - val_loss: 0.4057 - val_dice_coef: 0.5943
Epoch 329/1000
578/578 [==============================] - ETA: 0s - loss: 0.3536 - dice_coef: 0.6464
Epoch 00329: val_dice_coef did not improve from 0.59458
578/578 [==============================] - 93s 161ms/step - loss: 0.3536 - dice_coef: 0.6464 - val_loss: 0.4071 - val_dice_coef: 0.5929
Epoch 330/1000
578/578 [==============================] - ETA: 0s - loss: 0.3533 - dice_coef: 0.6467
Epoch 00330: val_dice_coef did not improve from 0.59458
578/578 [==============================] - 93s 161ms/step - loss: 0.3533 - dice_coef: 0.6467 - val_loss: 0.4061 - val_dice_coef: 0.5939
Epoch 331/1000
578/578 [==============================] - ETA: 0s - loss: 0.3537 - dice_coef: 0.6463
Epoch 00331: val_dice_coef did not improve from 0.59458
578/578 [==============================] - 93s 161ms/step - loss: 0.3537 - dice_coef: 0.6463 - val_loss: 0.4056 - val_dice_coef: 0.5944
Epoch 332/1000
578/578 [==============================] - ETA: 0s - loss: 0.3526 - dice_coef: 0.6474
Epoch 00332: val_dice_coef improved from 0.59458 to 0.59470, saving model to /content/drive/My Drive/cannet-vgg-14train-edge-lgg/weights-332-0.5947.hdf5
578/578 [==============================] - 93s 161ms/step - loss: 0.3526 - dice_coef: 0.6474 - val_loss: 0.4053 - val_dice_coef: 0.5947
Epoch 333/1000
578/578 [==============================] - ETA: 0s - loss: 0.3526 - dice_coef: 0.6474
Epoch 00333: val_dice_coef did not improve from 0.59470
578/578 [==============================] - 93s 161ms/step - loss: 0.3526 - dice_coef: 0.6474 - val_loss: 0.4055 - val_dice_coef: 0.5945
Epoch 334/1000
578/578 [==============================] - ETA: 0s - loss: 0.3524 - dice_coef: 0.6476
Epoch 00334: val_dice_coef improved from 0.59470 to 0.59498, saving model to /content/drive/My Drive/cannet-vgg-14train-edge-lgg/weights-334-0.5950.hdf5
578/578 [==============================] - 94s 162ms/step - loss: 0.3524 - dice_coef: 0.6476 - val_loss: 0.4050 - val_dice_coef: 0.5950
Epoch 335/1000
```


#using augmentation tecnique the model has dice coeff as 0.6273 as already we are using pr
#so we can check without augmentation technique

CANNET USING VGG BACKBONE-WITHOUT AUGMENTATION TECHNIQUE

```
IMAGE_SIZE=[224,224]
vgg=VGG16(include_top=False,weights="imagenet",input_shape=IMAGE_SIZE + [3]) #with

for layer in vgg.layers:
  layer.trainable = False

c1 = vgg.get_layer('block2_conv2').output
vgg_op=vgg.get_layer('block5_conv3').output

op_g=global_flow(name="global_flow")(vgg_op)

op_cx1=context_flow(name="context_flow1")([vgg.output,op_g])
op_cx2=context_flow(name="context_flow2")([vgg.output,op_cx1])
op_cx3=context_flow(name="context_flow3")([vgg.output,op_cx2])

op_add=tf.keras.layers.Add()([op_g,op_cx1,op_cx2,op_cx3])

op_fsm=fsm(name="feature_selection")(op_add)
```

```
    op_agcn=agcn(name="global_conv_net")(c1)

    op_concat=tf.concat([op_agcn,op_fsm],axis=-1)
    op_conv=tf.keras.layers.Conv2D(4,kernel_size=(3,3),padding='same')(op_concat)
    op_up=tf.keras.layers.UpSampling2D(size=(2,2),interpolation='bilinear')(op_conv)
    Y_vgg=tf.keras.layers.Activation('softmax')(op_up)


Y_vgg.shape
```

```
    TensorShape([None, 224, 224, 4])
```

```
model_vgg = tf.keras.Model(inputs = vgg.input,outputs=Y_vgg)
learning_rate=0.000001
model_vgg.compile(optimizer=tf.keras.optimizers.Adam(lr=learning_rate), loss=dice_coef_los
model_vgg.summary()
```

```
    Model: "functional_21"
    _________________________________________________________________________________
    Layer (type)                 Output Shape              Param #     Connected to
    =================================================================================
    input_11 (InputLayer)        [(None, 224, 224, 3) 0
    _________________________________________________________________________________
    block1_conv1 (Conv2D)        (None, 224, 224, 64) 1792        input_11[0][0]
    _________________________________________________________________________________
    block1_conv2 (Conv2D)        (None, 224, 224, 64) 36928       block1_conv1[0][0
    _________________________________________________________________________________
    block1_pool (MaxPooling2D)   (None, 112, 112, 64) 0           block1_conv2[0][0
    _________________________________________________________________________________
    block2_conv1 (Conv2D)        (None, 112, 112, 128 73856       block1_pool[0][0]
    _________________________________________________________________________________
    block2_conv2 (Conv2D)        (None, 112, 112, 128 147584      block2_conv1[0][0
    _________________________________________________________________________________
    block2_pool (MaxPooling2D)   (None, 56, 56, 128)  0           block2_conv2[0][0
    _________________________________________________________________________________
    block3_conv1 (Conv2D)        (None, 56, 56, 256)  295168      block2_pool[0][0]
    _________________________________________________________________________________
    block3_conv2 (Conv2D)        (None, 56, 56, 256)  590080      block3_conv1[0][0
    _________________________________________________________________________________
    block3_conv3 (Conv2D)        (None, 56, 56, 256)  590080      block3_conv2[0][0
    _________________________________________________________________________________
    block3_pool (MaxPooling2D)   (None, 28, 28, 256)  0           block3_conv3[0][0
    _________________________________________________________________________________
    block4_conv1 (Conv2D)        (None, 28, 28, 512)  1180160     block3_pool[0][0]
    _________________________________________________________________________________
    block4_conv2 (Conv2D)        (None, 28, 28, 512)  2359808     block4_conv1[0][0
    _________________________________________________________________________________
    block4_conv3 (Conv2D)        (None, 28, 28, 512)  2359808     block4_conv2[0][0
    _________________________________________________________________________________
    block4_pool (MaxPooling2D)   (None, 14, 14, 512)  0           block4_conv3[0][0
    _________________________________________________________________________________
    block5_conv1 (Conv2D)        (None, 14, 14, 512)  2359808     block4_pool[0][0]
    _________________________________________________________________________________
    block5_conv2 (Conv2D)        (None, 14, 14, 512)  2359808     block5_conv1[0][0
    _________________________________________________________________________________
    block5_conv3 (Conv2D)        (None, 14, 14, 512)  2359808     block5_conv2[0][0
    _________________________________________________________________________________
    global_flow (global_flow)    (None, 7, 7, 64)     34880       block5_conv3[0][0
```

```
block5_pool (MaxPooling2D)      (None, 7, 7, 512)    0          block5_conv3[0][0
context_flow1 (context_flow)    (None, 7, 7, 64)     377088     block5_pool[0][0]
                                                                global_flow[0][0]
context_flow2 (context_flow)    (None, 7, 7, 64)     377088     block5_pool[0][0]
                                                                context_flow1[0][(
context_flow3 (context_flow)    (None, 7, 7, 64)     377088     block5_pool[0][0]
                                                                context_flow2[0][(
add_65 (Add)                    (None, 7, 7, 64)     0          global_flow[0][0]
                                                                context_flow1[0][(
                                                                context_flow2[0][(
                                                                context_flow3[0][(
```

```
tf.compat.v1.enable_eager_execution()


# classes for data loading and preprocessing
#to create masks for each patch
classes=[0,1,2,4]
import numpy as np

from random import sample
class Dataset:
    """Read images, apply augmentation and preprocessing transformations.

    Args:
        images_dir : path to images folder (directories of all sequences)
        masks_dir : path to segmentation masks folder
        classes : values of classes to extract from segmentation mask


    """

    def __init__(
            self,
            flair_paths,t1_paths,t1ce_paths,mask_paths,

            augmentation=None,
            classes=classes,
    ):
        self.images_flair =flair_paths
        self.images_t1 =t1_paths
        #self.images_t2 =t2_paths
        self.images_t1ce =t1ce_paths
        self.masks_fps =mask_paths
        self.classes=classes
        self.augmentation = augmentation

    def __getitem__(self, i):

        # read data
```

```
        #print(self.images_flair[i])
        #print(i)
        image_f= np.load(str (self.images_flair[i]))
        image_flair=image_f[8:232,8:232]
        image_1 = np.load(str (self.images_t1[i]))
        image_t1=image_1[8:232,8:232]
        image_1ce = np.load(str (self.images_t1ce[i]))
        image_t1ce=image_1ce[8:232,8:232]
        #image_2 = np.load(str (self.images_t2[i]))
        #image_t2=image_2[8:232,8:232]
        m = np.load(str (self.masks_fps[i]))
        mask=m[8:232,8:232]
        #mask=self.masks_fps[i]


        # extract certain classes from mask (e.g. cars)
        masks = [(mask == v) for v in self.classes]
        mask = np.stack(masks, axis=-1).astype('float')
        #l=[image_flair,image_t1,image_t1ce]
        #v=[0,1,2,3]
        #x,y,z=sample(v,3)

        image=np.stack((image_flair,image_t1,image_t1ce), axis=-1).astype('float')

        return image,mask


    def __len__(self):
      return len(self.masks_fps)



from sklearn.model_selection import train_test_split
X=new_df.drop(["mask","t2"],axis=1)
y=new_df["mask"]


X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=10)


train_data_vgg= Dataset(
    flair_paths=list(X_train["flair"]),t1_paths=list(X_train["t1"]),t1ce_paths=list(X_trai
    classes=classes

)

test_data_vgg= Dataset(
    flair_paths=list(X_test["flair"]),t1_paths=list(X_test["t1"]),t1ce_paths=list(X_test["
    classes=classes

)


train_dataloader_vgg = Dataloder(train_data_vgg, batch_size=4, shuffle=True)
valid_dataloader_vgg = Dataloder(test_data_vgg, batch_size=1, shuffle=True)
```

```
# check shapes for errors
assert train_dataloader_vgg[0][0].shape == (4, 224, 224, 3)
assert train_dataloader_vgg[0][1].shape == (4, 224, 224, 4)


# define callbacks for learning rate scheduling and best checkpoints saving
checkpoint_reducelr = [
    tf.keras.callbacks.ModelCheckpoint('/content/drive/My Drive/cannet-vgg-14/weights-{epo
]


model_vgg.load_weights("/content/drive/My Drive/cannet-vgg-14/weights-331-0.7672.hdf5")


history = model_vgg.fit_generator(train_dataloader, steps_per_epoch=len(train_dataloader),
```

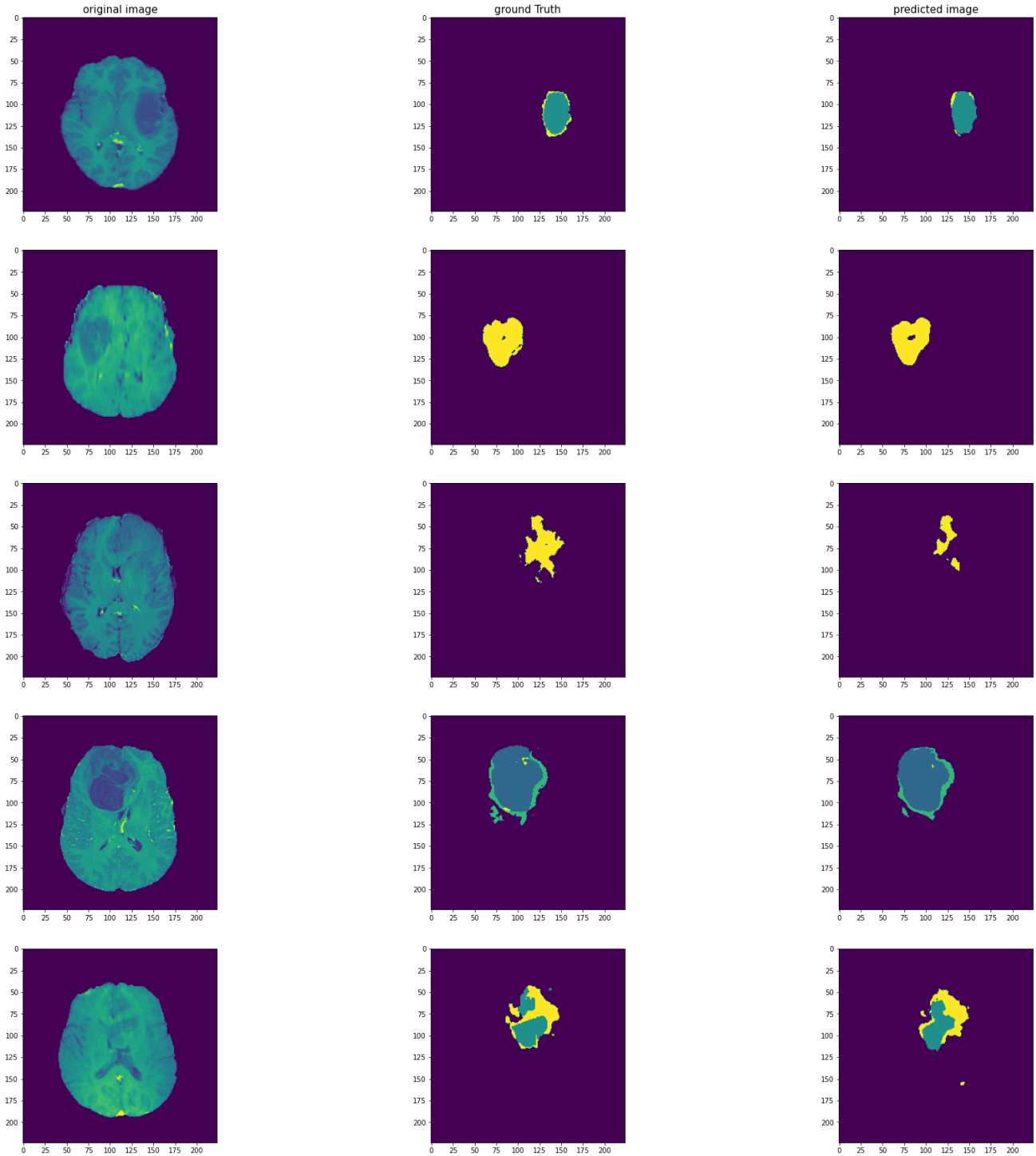
```
Epoch 2/500
578/578 [==============================] - ETA: 0s - loss: 0.1400 - dice_coef: 0.8600
Epoch 00002: val_dice_coef did not improve from 0.72951
578/578 [==============================] - 43s 74ms/step - loss: 0.1400 - dice_coef: 0.8600 - val_loss: 0.2986 - val_dice_coef: 0.7014
Epoch 3/500
578/578 [==============================] - ETA: 0s - loss: 0.1401 - dice_coef: 0.8599
Epoch 00003: val_dice_coef improved from 0.72951 to 0.73988, saving model to /content/drive/My Drive/cannet-vgg-14/weights-03-0.7399.hdf5
578/578 [==============================] - 44s 76ms/step - loss: 0.1401 - dice_coef: 0.8599 - val_loss: 0.2601 - val_dice_coef: 0.7399
Epoch 4/500
578/578 [==============================] - ETA: 0s - loss: 0.1400 - dice_coef: 0.8600
Epoch 00004: val_dice_coef improved from 0.73988 to 0.74042, saving model to /content/drive/My Drive/cannet-vgg-14/weights-04-0.7404.hdf5
578/578 [==============================] - 45s 78ms/step - loss: 0.1400 - dice_coef: 0.8600 - val_loss: 0.2596 - val_dice_coef: 0.7404
Epoch 5/500
578/578 [==============================] - ETA: 0s - loss: 0.1392 - dice_coef: 0.8608
Epoch 00005: val_dice_coef did not improve from 0.74042
578/578 [==============================] - 43s 74ms/step - loss: 0.1392 - dice_coef: 0.8608 - val_loss: 0.2692 - val_dice_coef: 0.7308
Epoch 6/500
578/578 [==============================] - ETA: 0s - loss: 0.1391 - dice_coef: 0.8609
Epoch 00006: val_dice_coef did not improve from 0.74042
578/578 [==============================] - 42s 73ms/step - loss: 0.1391 - dice_coef: 0.8609 - val_loss: 0.2914 - val_dice_coef: 0.7086
Epoch 7/500
578/578 [==============================] - ETA: 0s - loss: 0.1391 - dice_coef: 0.8609
Epoch 00007: val_dice_coef did not improve from 0.74042
578/578 [==============================] - 42s 73ms/step - loss: 0.1391 - dice_coef: 0.8609 - val_loss: 0.2601 - val_dice_coef: 0.7399
Epoch 8/500
578/578 [==============================] - ETA: 0s - loss: 0.1396 - dice_coef: 0.8604
Epoch 00008: val_dice_coef improved from 0.74042 to 0.74895, saving model to /content/drive/My Drive/cannet-vgg-14/weights-08-0.7490.hdf5
578/578 [==============================] - 43s 74ms/step - loss: 0.1396 - dice_coef: 0.8604 - val_loss: 0.2510 - val_dice_coef: 0.7490
```



```
#Out of all the experiments cannet with pretrained vgg backbone got generelized well and a
# also trainable parameters are also very low having less computational cost
#Major advantage of using this network is basline unet has 138 million parameter where as


#prediction
def predict_(slice_):
  #image
  img_data=valid_dataloader_vgg[slice_]
  image=img_data[0]# original image


  #ground truth image
  ground_truth=img_data[1][0]
  ground_truth=np.argmax(ground_truth,axis=-1)
  print("unique_classes in ground truth :",np.unique(ground_truth))


  #predicted image

  pred_img=model_vgg.predict(image)
  pred_img=np.argmax(pred_img[0],axis=-1)
```

```
  print("unique_classes in predicted image :",np.unique(pred_img))

  return image,ground_truth,pred_img



#prediction of cannet model-vgg backbone
k=1
plt.figure(figsize=(30,30))
slice_no=[140,196,116,126,19]
for i in slice_no:
  image,groundtruth,pred_image=predict_(i)
  l=[image[0][:,:,2],groundtruth,pred_image] #taken for a single modality
  for i in l:
    plt.subplot(5,3,k)
    plt.imshow(i)
    if(k==1):
      plt.title("original image",fontdict={"fontsize":15})
    if(k==2):
      plt.title("ground Truth",fontdict={"fontsize":15})
    if(k==3):
      plt.title("predicted image",fontdict={"fontsize":15})

    k=k+1
```

```
unique_classes in ground truth : [0 1 2]
unique_classes in predicted image : [0 1 2]
unique_classes in ground truth : [0 2]
unique_classes in predicted image : [0 2]
unique_classes in ground truth : [0 2]
unique_classes in predicted image : [0 2]
unique_classes in ground truth : [0 1 2 3]
unique_classes in predicted image : [0 1 2 3]
unique_classes in ground truth : [0 1 2]
unique_classes in predicted image : [0 1 2]
```



```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Model","trainable parameters","dice coeff"]


x.add_row(["Baseline-Unet Model",  "138m" , 0.7679])
x.add_row(["Basic Cannet Model", "275k" , "vanishing gradients"])
x.add_row(["Cannet without Identity Blocks", "239k", 0.68])
x.add_row(["Cannet using VGG Backbone-With Augmentation", "3m" , 0.6273])
x.add_row(["Cannet using VGG Backbone-Without Augmentation", "1.4m",0.7672])

print(x)
```

```
+-----------------------------------------------+----------------------+-----------
|                     Model                     | trainable parameters |      dice c
+-----------------------------------------------+----------------------+-----------
|              Baseline-Unet Model              |         138m         |        0.76
|               Basic Cannet Model              |         275k         | vanishing g
|         Cannet without Identity Blocks        |         239k         |          0.6
|  Cannet using VGG Backbone-With Augmentation  |          3m          |        0.62
| Cannet using VGG Backbone-Without Augmentation |         1.4m         |        0.76
+-----------------------------------------------+----------------------+-----------
```

Here compared to all the models, final architecture which I've used for segmenting the tumor is CANNET Architecture using VGG16 as Backbone network without the augmentation techniques