```
!pip install tensorflow==2.3.0
```

```
Requirement already satisfied: tensorflow==2.3.0 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: gast==0.3.3 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: google-pasta>=0.1.8 in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: tensorflow-estimator<2.4.0,>=2.3.0 in /usr/local/lib/p
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: scipy==1.4.1 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: h5py<2.11.0,>=2.10.0 in /usr/local/lib/python3.6/dist-
Requirement already satisfied: grpcio>=1.8.6 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: wrapt>=1.11.1 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: keras-preprocessing<1.2,>=1.1.1 in /usr/local/lib/pyth
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: astunparse==1.6.3 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.6/dist-pack
Requirement already satisfied: numpy<1.19.0,>=1.16.0 in /usr/local/lib/python3.6/dist
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: tensorboard<3,>=2.3.0 in /usr/local/lib/python3.6/dist
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (
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: google-auth<2,>=1.6.3 in /usr/local/lib/python3.6/dist
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/pyt
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.6/dist-packa
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: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: rsa<5,>=3.1.4; python_version >= "3" in /usr/local/lib
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.6/dis
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.6/dist
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.6/d
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 SimpleITK
```

```
Collecting SimpleITK
  Downloading https://files.pythonhosted.org/packages/f3/cb/a15f4612af8e37f3627fc7fb2
     |████████████████████████████████| 44.9MB 69kB/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
from keras.models import load_model


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

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

Entire pipeline function for segmenting the tumor

```
def function_(patient_dirs):
  """
  This function takes patients  image paths and performs the following operation:

  1.Bias Field correction
  2.pre-processing of corrected image modalities
  3.create slices of the pre-processed images
  4.predicts the tumor regions in slices using trained best model-Cannet with vgg as backb

  Function parameters:
  image_path: patients MRI volumes path

  Returns:
  X:patients image
  predictions:predicted label

  """
  def bias_field_correction(image_p):
    """
    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
    image segmentation algorithms to process the images

    Function Parameters:
    --------------------
    image_path: path of the image

    Returns:

    img_c:corrected image

    """
    img = sitk.ReadImage(image_p)
    img_mask = sitk.OtsuThreshold(img)
    img = sitk.Cast(img, sitk.sitkFloat32)
    corrector = sitk.N4BiasFieldCorrectionImageFilter()
    img_c = corrector.Execute(img, img_mask)
    corrected_image=sitk.GetArrayFromImage(img_c)


    return corrected_image
```

```
flair=[]
t1=[]
t1ce=[]
t2=[]
mask=[]

for i in patient_dirs:
  modalities=os.listdir(i)
  for seq in modalities:
      if 'flair' in seq:
          c=bias_field_correction(i+'/'+seq)
          flair.append(c)
      if ('t1' in seq) and ('t1ce' not in seq):
          c=bias_field_correction(i+'/'+seq)
          t1.append(c)
      if ('t1ce' in seq):
          c=bias_field_correction(i+'/'+seq)
          t1ce.append(c)
      if ('seg' in seq):
          c=bias_field_correction(i+'/'+seq)
          mask.append(c)


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
```

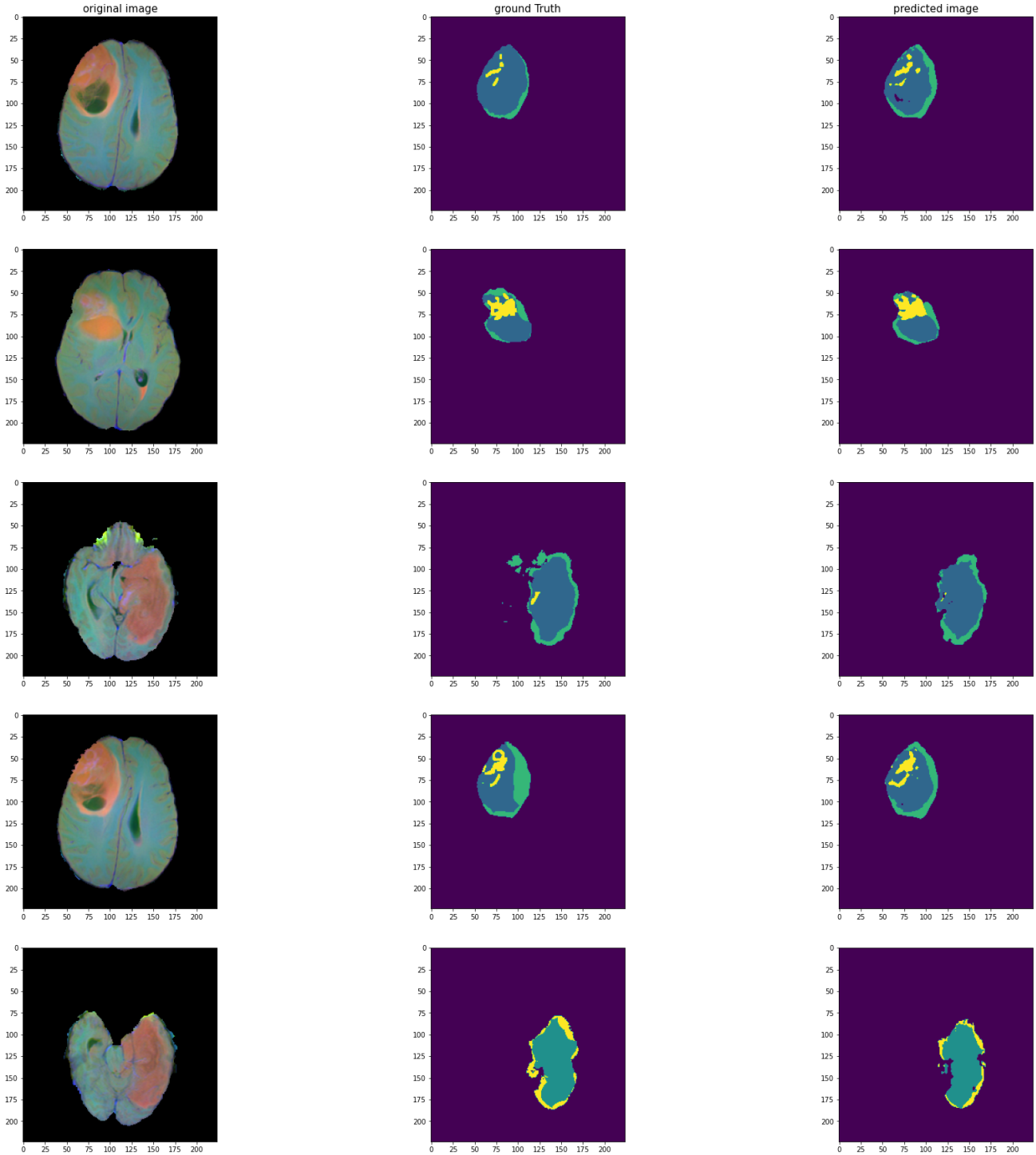
```
pre_flair=[]
pre_t1=[]
pre_t1ce=[]
for a,b,c in zip(flair,t1,t1ce):
  pre_flair.append(preprocessing(a))
  pre_t1.append(preprocessing(b))
  pre_t1ce.append(preprocessing(c))


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

  """
  This function create slices of an image volume
  Function parameters:
  ---------------------
   grade_type: list of image volumes of type of grade
   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:
   --------
   b:slice indexes
   slices:slices of given modalities

  """

  count=0
  slices=[]
  for image in grade_type:
      #print(image.shape[0])

      if (collected == False):
          v=[]

          for slice_ in range(image.shape[0]):
              if (slice_%2==1) and (slice_<141):
                  if (np.count_nonzero(image[slice_])>=7359):
                      v.append(slice_)
                      slices.append(image[slice_].astype('float32'))

          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[slice_other_modality]

                    slices.append(val.astype('float32'))
                count+=1
                break

    return b,slices

  indexes_f,flair_slices=create_slices(pre_flair,collected=False,b=[])
  indexes_t1,t1_slices=create_slices(pre_t1,collected=True,b=indexes_f)
  indexes_t1ce,t1ce_slices=create_slices(pre_t1ce,collected=True,b=indexes_f)
  indexes_seg,mask_slices=create_slices(mask,collected=True,b=indexes_f)



#min no.of non zero pixels
  min_thresh=10005

  l=[]
  for index,Slice in enumerate(flair_slices):
    if np.count_nonzero(Slice)>=min_thresh:
        l.append(index)
  Data_flair=[]
  Data_t1=[]
  Data_t1ce=[]
  Data_t2=[]
  Data_mask=[]

  for i in l:
    Data_flair.append(flair_slices[i])
    Data_t1.append(t1_slices[i])
    Data_t1ce.append(t1ce_slices[i])
    Data_t2.append(t2_slices[i])
    Data_mask.append(mask_slices[i])



    # classes for data loading and preprocessing
    #to create masks for each slice
  classes=[0,1,2,4]
  X=[]
  y=[]
  for a,b,c,d in zip(Data_flair,Data_t1,Data_t1ce,Data_mask):
    a1=a[8:232,8:232]
    b1=b[8:232,8:232]
    c1=c[8:232,8:232]
    d1=d[8:232,8:232]
    image=np.stack((a1,b1,c1),axis=-1)
    masks = [(d1 == v) for v in classes]
    mask = np.stack(masks, axis=-1).astype('float')
    X.append(image)
    y.append(mask)
 #Loading pretrained cannet model with vgg backbone without augmentation for predictions
  model_canet=load_model("/content/drive/My Drive/dl_cannet/cannet_weights.h5")
  predictions=model_canet.predict(np.array(X))
```

```
  return X,predictions



#patient modalities
patient_dirs=['/content/Brats18_TCIA09_255_1','/content/Brats18_TCIA13_650_1','/content/Br


X,predictions=function_(patient_dirs)
```

Segmentation of Tumor for 5 patients -Using CANNET Architecture with VGG16 as backbone without Augmentation techniques

```
#plot predictions
k=1
plt.figure(figsize=(30,30))
slice_no=[120,107,144,118,138]
for i in slice_no:
  image,groundtruth,pred_image=X[i],y[i],predictions[i]
  l=[image,np.argmax(groundtruth,axis=-1),np.argmax(pred_image,axis=-1)] #taken for a sing
  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
```


original image
ground Truth
predicted image