**Data Augmentation**

**About the data:**  
The dataset contains 2 folders: yes and no which contains 253 Brain MRI Images. The folder yes contains 155 Brain MRI Images that are tumorous and the folder no contains 98 Brain MRI Images that are non-tumorous. You can find [here](https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection).

Since this is a small dataset, I used data augmentation in order to create more images.

Also, we could solve the data imbalance issue (since 61% of the data belongs to the tumorous class) using data augmentation.

**Import Necessary Modules**

In [1]:

**import** tensorflow **as** tf

**from** keras.preprocessing.image **import** ImageDataGenerator

**import** cv2

**import** imutils

**import** matplotlib.pyplot **as** plt

**from** os **import** listdir

**import** time

**%matplotlib** inline

Using TensorFlow backend.

In [2]:

*# Nicely formatted time string*

**def** hms\_string(sec\_elapsed):

h **=** int(sec\_elapsed **/** (60 **\*** 60))

m **=** int((sec\_elapsed **%** (60 **\*** 60)) **/** 60)

s **=** sec\_elapsed **%** 60

**return** f"{h}:{m}:{round(s,1)}"

In [3]:

**def** augment\_data(file\_dir, n\_generated\_samples, save\_to\_dir):

"""

Arguments:

file\_dir: A string representing the directory where images that we want to augment are found.

n\_generated\_samples: A string representing the number of generated samples using the given image.

save\_to\_dir: A string representing the directory in which the generated images will be saved.

"""

*#from keras.preprocessing.image import ImageDataGenerator*

*#from os import listdir*

data\_gen **=** ImageDataGenerator(rotation\_range**=**10,

width\_shift\_range**=**0.1,

height\_shift\_range**=**0.1,

shear\_range**=**0.1,

brightness\_range**=**(0.3, 1.0),

horizontal\_flip**=True**,

vertical\_flip**=True**,

fill\_mode**=**'nearest'

)

**for** filename **in** listdir(file\_dir):

*# load the image*

image **=** cv2**.**imread(file\_dir **+** '\\' **+** filename)

*# reshape the image*

image **=** image**.**reshape((1,)**+**image**.**shape)

*# prefix of the names for the generated sampels.*

save\_prefix **=** 'aug\_' **+** filename[:**-**4]

*# generate 'n\_generated\_samples' sample images*

i**=**0

**for** batch **in** data\_gen**.**flow(x**=**image, batch\_size**=**1, save\_to\_dir**=**save\_to\_dir,

save\_prefix**=**save\_prefix, save\_format**=**'jpg'):

i **+=** 1

**if** i **>** n\_generated\_samples:

**break**

Remember that 61% of the data (155 images) are tumorous. And, 39% of the data (98 images) are non-tumorous.  
So, in order to balance the data we can generate 9 new images for every image that belongs to 'no' class and 6 images for every image that belongs the 'yes' class.

In [4]:

start\_time **=** time**.**time()

augmented\_data\_path **=** 'augmented data/'

*# augment data for the examples with label equal to 'yes' representing tumurous examples*

augment\_data(file\_dir**=**yes\_path, n\_generated\_samples**=**6, save\_to\_dir**=**augmented\_data\_path**+**'yes')

*# augment data for the examples with label equal to 'no' representing non-tumurous examples*

augment\_data(file\_dir**=**no\_path, n\_generated\_samples**=**9, save\_to\_dir**=**augmented\_data\_path**+**'no')

end\_time **=** time**.**time()

execution\_time **=** (end\_time **-** start\_time)

print(f"Elapsed time: {hms\_string(execution\_time)}")

Elapsed time: 0:2:16.2

Let's see how many tumorous and non-tumorous examples after performing data augmentation:

In [5]:

**def** data\_summary(main\_path):

yes\_path **=** main\_path**+**'yes'

no\_path **=** main\_path**+**'no'

*# number of files (images) that are in the the folder named 'yes' that represent tumorous (positive) examples*

m\_pos **=** len(listdir(yes\_path))

*# number of files (images) that are in the the folder named 'no' that represent non-tumorous (negative) examples*

m\_neg **=** len(listdir(no\_path))

*# number of all examples*

m **=** (m\_pos**+**m\_neg)

pos\_prec **=** (m\_pos**\*** 100.0)**/** m

neg\_prec **=** (m\_neg**\*** 100.0)**/** m

print(f"Number of examples: {m}")

print(f"Percentage of positive examples: {pos\_prec}%, number of pos examples: {m\_pos}")

print(f"Percentage of negative examples: {neg\_prec}%, number of neg examples: {m\_neg}")

In [6]:

data\_summary(augmented\_data\_path)

Number of examples: 2065

Percentage of positive examples: 52.54237288135593%, number of pos examples: 1085

Percentage of negative examples: 47.45762711864407%, number of neg examples: 980

That's it for this notebook. Now, we can use the augmented data to train our convolutional neural network.

In [ ]: