cnn_model_1-Copy1

February 16, 2020

In [1]:
```
import numpy as np
import pandas as pd
import os
print(os.listdir("/home/mlab/Documents/brats_data"))
```

```
['val', 'train']
```

In [2]:
```
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
import random
import gc
```

In [3]:
```
train_dir = 'brats_data/train/high/'
train_dir_l = 'brats_data/train/low/'
train_high = [train_dir+str(i) for i in os.listdir(train_dir) if 'High' in i]
train_low = [train_dir_l+str(i) for i in os.listdir(train_dir_l) if 'Low' in i]
```

In [4]:
```
test_dir='brats_data/val/high/'
test_dir_l='brats_data/val/low/'
test_img_high=[test_dir+str(i) for i in os.listdir(test_dir) if 'High' in i]
test_img_low =[test_dir_l+str(i) for i in os.listdir(test_dir_l) if 'Low' in i]
train_imgs = train_high[:3000]+train_low[:3000]
# train_imgs = train_high+train_low
random.shuffle(train_imgs)
test_img=test_img_low[:500]+test_img_high[:500]
# test_img=test_img_low+test_img_high
# random.shuffle(test_img)
```

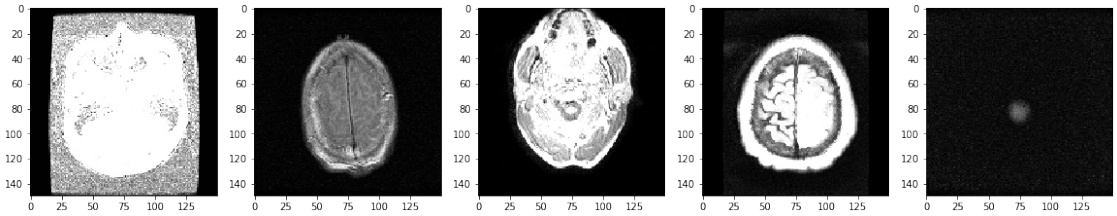
In [5]:
```
del train_high
del train_low
del test_img_high
del test_img_low
gc.collect()
```

Out[5]: 75

In [6]:
```
# a=5
# var = f'my name is {a}' can put list comprehension in here
# var
nrows = 150
ncolumns = 150
channels = 3

def read_and_process_image(list_of_images):
    X=[]
    y=[]
    for image in list_of_images:
        X.append(cv2.resize(cv2.imread(image,cv2.IMREAD_COLOR),(nrows,ncolumns),
                            interpolation=cv2.INTER_CUBIC))
        if 'High'in image:
            y.append(1)
        elif 'Low' in image:
            y.append(0)
    return X,y
```

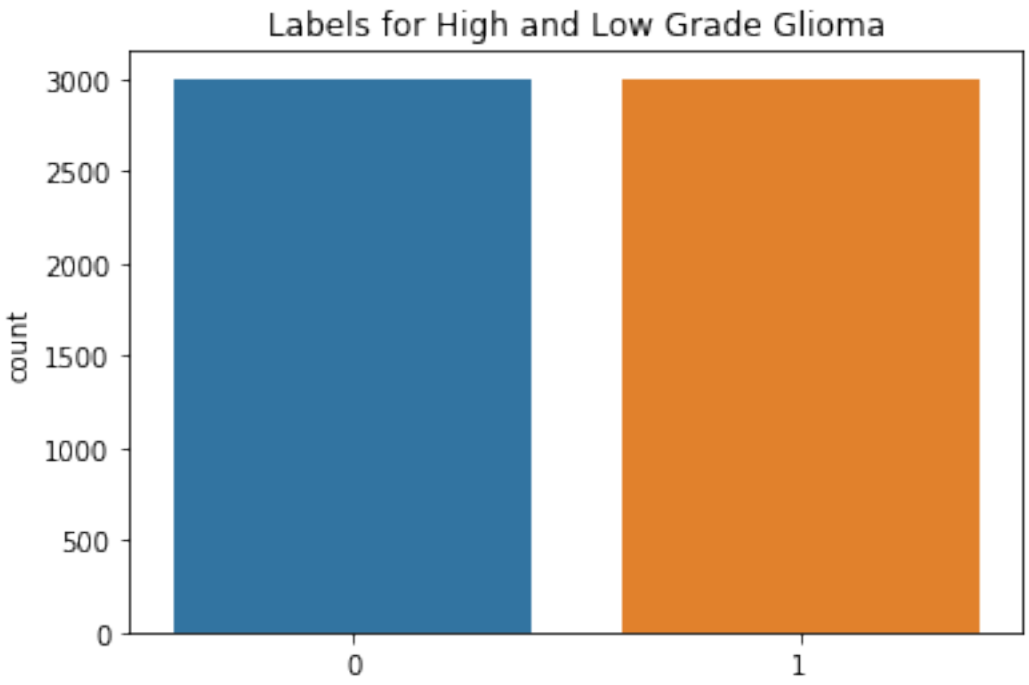
In [7]:
```
X, y = read_and_process_image(train_imgs)
```

In [8]:
```
plt.figure(figsize=(20,10))
columns = 5
for i in range(columns):
    plt.subplot(5/columns+1,columns,i+1)
    plt.imshow(X[i])
```



In [9]:
```
import seaborn as sns
del train_imgs

gc.collect
X = np.array(X)
y=np.array(y)

sns.countplot(y)
plt.title('Labels for High and Low Grade Glioma')
```

Out[9]: Text(0.5, 1.0, 'Labels for High and Low Grade Glioma')




```
In [10]: print("Shape of train images is: ",X.shape)
         print("Shape of labels is: ", y.shape)
```

```
Shape of train images is:  (6000, 150, 150, 3)
Shape of labels is:  (6000,)
```

```
In [11]: from sklearn.model_selection import train_test_split
         X_train,X_val,y_train,y_val = train_test_split(X,y,test_size=0.2,random_state=2)

         print("Shape of train image is: ",X_train.shape)
         print("Shape of validation images is: ", X_val.shape)
         print("Shape of labels is: ", y_train.shape)
         print("Shape of labels is: ", y_val.shape)
```

```
Shape of train image is:  (4800, 150, 150, 3)
Shape of validation images is:  (1200, 150, 150, 3)
Shape of labels is:  (4800,)
Shape of labels is:  (1200,)
```

```
In [12]: del X
         del y
         gc.collect()
```

```
ntrain = len(X_train)
nval = len(X_val)
#batch size should be a factor of 2.***4,8,16,32,64
batch_size = 128
```

In [13]:
```
from keras import layers
from keras import models
from keras import optimizers
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import img_to_array, load_img
```

```
Using TensorFlow backend.
```

In [14]:
```
#uses a smal vggnet
#32->64->128->512
model = models.Sequential()
model.add(layers.Conv2D(32,(3,3),activation='relu',input_shape=(150,150,3)))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64,(3,3),activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(128,(3,3),activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(512,(3,3),activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512,activation='relu'))
model.add(layers.Dense(1,activation='sigmoid')) #uses sigmoid at the end because we onlu have two classe
```

```
WARNING: Logging before flag parsing goes to stderr.
W0216 08:53:06.314436 140155455805248 deprecation_wrapper.py:119] From /home/mlab/anaconda3/lib/python

W0216 08:53:06.326355 140155455805248 deprecation_wrapper.py:119] From /home/mlab/anaconda3/lib/python

W0216 08:53:06.329110 140155455805248 deprecation_wrapper.py:119] From /home/mlab/anaconda3/lib/python

W0216 08:53:06.338634 140155455805248 deprecation_wrapper.py:119] From /home/mlab/anaconda3/lib/python

W0216 08:53:06.377693 140155455805248 deprecation_wrapper.py:119] From /home/mlab/anaconda3/lib/python

W0216 08:53:06.382473 140155455805248 deprecation.py:506] From /home/mlab/anaconda3/lib/python3.7/site-p
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
```

In [15]:
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
=================================================================
conv2d_1 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 15, 15, 512)       590336
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               12845568
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 513
=================================================================
Total params: 13,529,665
Trainable params: 13,529,665
Non-trainable params: 0
_________________________________________________________________
```

In [16]:
```
#We use the RMSprop optimizer with a learning rate of 0.0001
#We use ninary_crossentropy loss because its a binary classification
#optimizer = Adam(learning_rate=lr_schedule(0))
model.compile(loss='binary_crossentropy',optimizer=optimizers.Adam(lr=1e-4),metrics=['acc'])
```

```
W0216 08:53:06.432269 140155455805248 deprecation_wrapper.py:119] From /home/mlab/anaconda3/lib/python

W0216 08:53:06.435995 140155455805248 deprecation_wrapper.py:119] From /home/mlab/anaconda3/lib/python

W0216 08:53:06.439038 140155455805248 deprecation.py:323] From /home/mlab/anaconda3/lib/python3.7/site-p
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

In [17]:
```
train_datagen=ImageDataGenerator(rescale=1./255,
                                 rotation_range=40,
                                 width_shift_range=0.2,
```

```
                                    height_shift_range=0.2,
                                    shear_range=0.2,
                                    zoom_range=0.2,
                                    horizontal_flip=True)
        val_datagen=ImageDataGenerator(rescale=1./255)

In [18]: train_generator =train_datagen.flow(X_train,y_train,batch_size=batch_size)
        val_generator = val_datagen.flow(X_val,y_val,batch_size=batch_size)

In [19]: #the training part
        #train for 64 epochs with about 100 steps per epoch
        history = model.fit_generator(train_generator,
                                      steps_per_epoch=ntrain//batch_size,
                                      epochs=32,
                                      validation_data=val_generator,
                                      validation_steps=nval//batch_size)

Epoch 1/32
37/37 [==============================] - 71s 2s/step - loss: 0.6955 - acc: 0.5160 - val_loss: 0
Epoch 2/32
37/37 [==============================] - 68s 2s/step - loss: 0.6858 - acc: 0.5633 - val_loss: 0
Epoch 3/32
37/37 [==============================] - 68s 2s/step - loss: 0.6782 - acc: 0.5743 - val_loss: 0
Epoch 4/32
37/37 [==============================] - 68s 2s/step - loss: 0.6692 - acc: 0.5855 - val_loss: 0
Epoch 5/32
37/37 [==============================] - 68s 2s/step - loss: 0.6698 - acc: 0.5815 - val_loss: 0
Epoch 6/32
37/37 [==============================] - 68s 2s/step - loss: 0.6607 - acc: 0.5924 - val_loss: 0
Epoch 7/32
37/37 [==============================] - 68s 2s/step - loss: 0.6486 - acc: 0.6079 - val_loss: 0
Epoch 8/32
37/37 [==============================] - 68s 2s/step - loss: 0.6468 - acc: 0.6020 - val_loss: 0
Epoch 9/32
37/37 [==============================] - 68s 2s/step - loss: 0.6450 - acc: 0.6128 - val_loss: 0
Epoch 10/32
37/37 [==============================] - 68s 2s/step - loss: 0.6330 - acc: 0.6195 - val_loss: 0
Epoch 11/32
37/37 [==============================] - 68s 2s/step - loss: 0.6338 - acc: 0.6250 - val_loss: 0
Epoch 12/32
37/37 [==============================] - 69s 2s/step - loss: 0.6274 - acc: 0.6311 - val_loss: 0
Epoch 13/32
37/37 [==============================] - 68s 2s/step - loss: 0.6391 - acc: 0.6202 - val_loss: 0
Epoch 14/32
37/37 [==============================] - 68s 2s/step - loss: 0.6175 - acc: 0.6463 - val_loss: 0
Epoch 15/32
37/37 [==============================] - 68s 2s/step - loss: 0.6137 - acc: 0.6421 - val_loss: 0
Epoch 16/32
```

```
37/37 [==============================] - 68s 2s/step - loss: 0.6086 - acc: 0.6497 - val_loss: 0
Epoch 17/32
37/37 [==============================] - 68s 2s/step - loss: 0.6138 - acc: 0.6482 - val_loss: 0
Epoch 18/32
37/37 [==============================] - 68s 2s/step - loss: 0.6086 - acc: 0.6605 - val_loss: 0
Epoch 19/32
37/37 [==============================] - 68s 2s/step - loss: 0.6058 - acc: 0.6565 - val_loss: 0
Epoch 20/32
37/37 [==============================] - 68s 2s/step - loss: 0.6029 - acc: 0.6615 - val_loss: 0
Epoch 21/32
37/37 [==============================] - 68s 2s/step - loss: 0.5924 - acc: 0.6691 - val_loss: 0
Epoch 22/32
37/37 [==============================] - 68s 2s/step - loss: 0.5891 - acc: 0.6746 - val_loss: 0
Epoch 23/32
37/37 [==============================] - 68s 2s/step - loss: 0.6016 - acc: 0.6560 - val_loss: 0
Epoch 24/32
37/37 [==============================] - 68s 2s/step - loss: 0.5940 - acc: 0.6641 - val_loss: 0
Epoch 25/32
37/37 [==============================] - 68s 2s/step - loss: 0.5789 - acc: 0.6731 - val_loss: 0
Epoch 26/32
37/37 [==============================] - 68s 2s/step - loss: 0.5767 - acc: 0.6826 - val_loss: 0
Epoch 27/32
37/37 [==============================] - 68s 2s/step - loss: 0.5816 - acc: 0.6710 - val_loss: 0
Epoch 28/32
37/37 [==============================] - 68s 2s/step - loss: 0.5783 - acc: 0.6756 - val_loss: 0
Epoch 29/32
37/37 [==============================] - 68s 2s/step - loss: 0.5816 - acc: 0.6709 - val_loss: 0
Epoch 30/32
37/37 [==============================] - 67s 2s/step - loss: 0.5628 - acc: 0.6981 - val_loss: 0
Epoch 31/32
37/37 [==============================] - 67s 2s/step - loss: 0.5701 - acc: 0.6801 - val_loss: 0
Epoch 32/32
37/37 [==============================] - 63s 2s/step - loss: 0.5739 - acc: 0.6727 - val_loss: 0
```

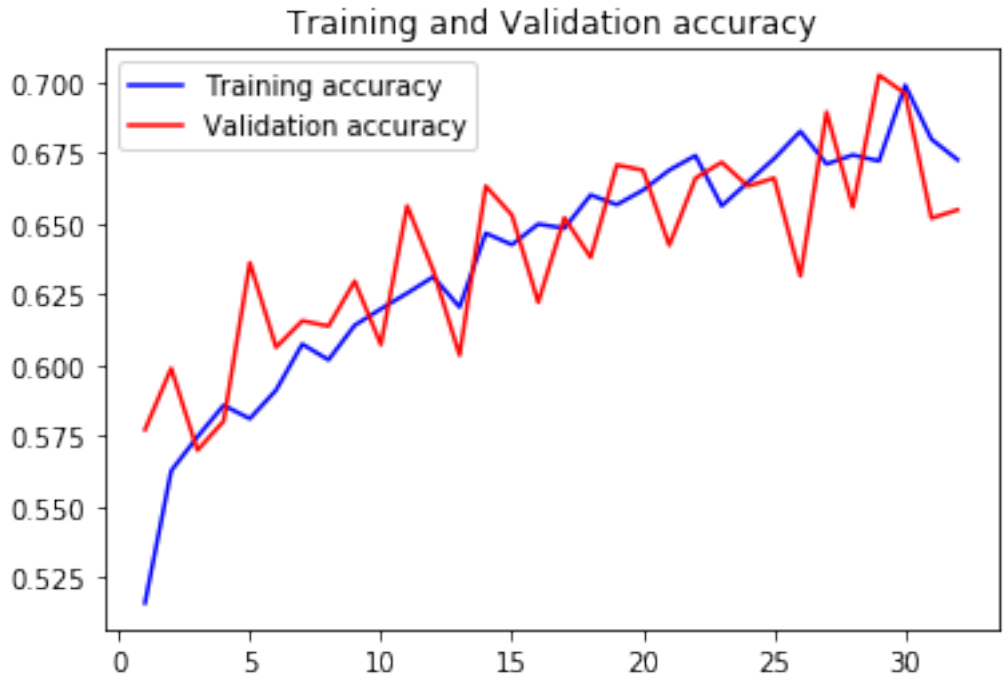
In [20]:
```
# #save the model
model.save_weights('model_weights.h5')
model.save('model_keras.h5')
```

In [21]:
```
_, accuracy = model.evaluate(X_train, y_train)
print('Accuracy training: %.2f' % (accuracy*100))
_, accuracy = model.evaluate(X_val, y_val)
print('Accuracy val: %.2f' % (accuracy*100))
```

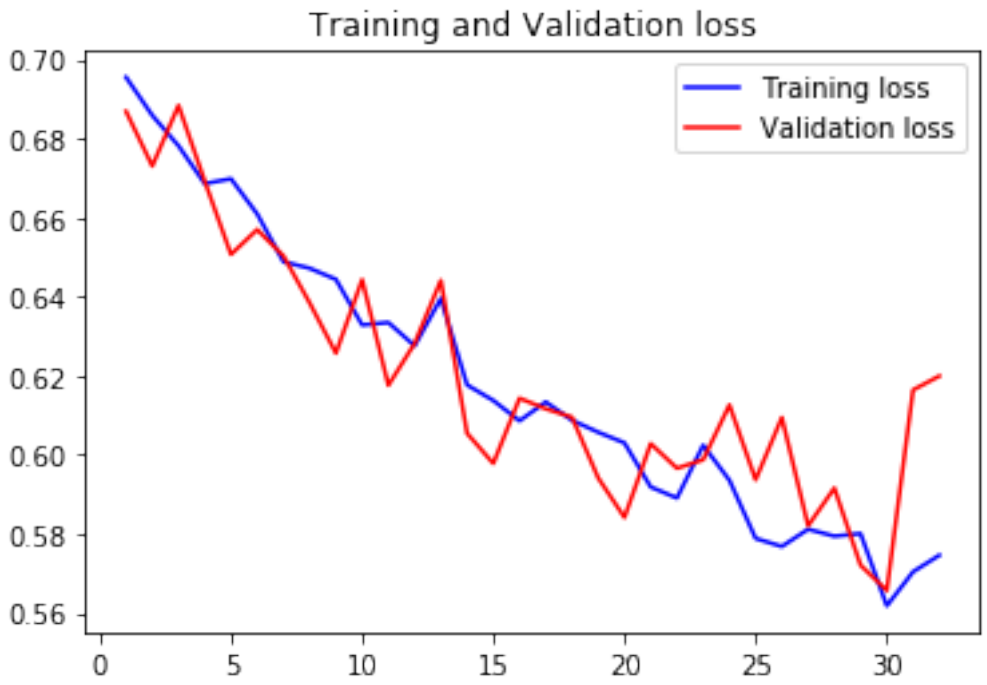
```
4800/4800 [==============================] - 12s 3ms/step
Accuracy training: 58.29
1200/1200 [==============================] - 3s 3ms/step
Accuracy val: 56.75
```

In [22]:
```
#plot the train and val curve
#get the details from the history object
acc = history.history['acc']
val_acc=history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1,len(acc)+1)

#train and validation accuracy
plt.plot(epochs,acc,'b',label='Training accuracy')
plt.plot(epochs,val_acc,'r',label='Validation accuracy')
plt.title('Training and Validation accuracy')
plt.legend()
```

Out[22]: <matplotlib.legend.Legend at 0x7f7823fa25c0>

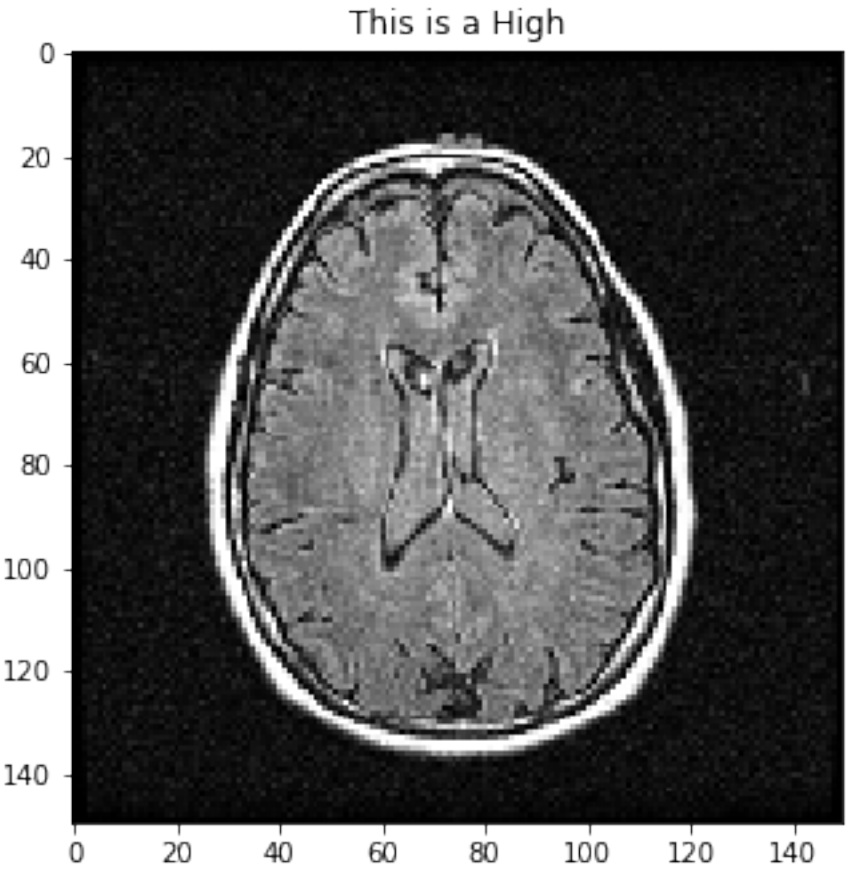


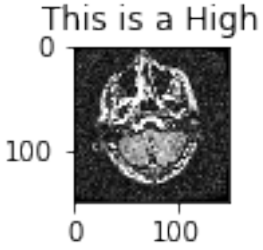
In [23]:
```
#train and validation loss
plt.plot(epochs, loss, 'b',label='Training loss')
plt.plot(epochs, val_loss, 'r',label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()
```

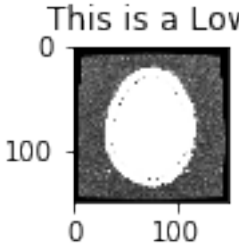


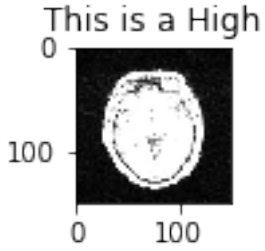
In [24]:
```
#predict on the first 10 images of the test set
X_test, y_test = read_and_process_image(test_img[:10]) #Y-test in this case will be empty
x=np.array(X_test)
test_datagen = ImageDataGenerator(rescale=1./255)
```

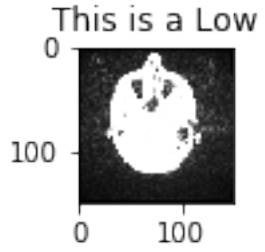
In [25]:
```
i = 0
text_labels = []
plt.figure(figsize=(30,20))
for batch in test_datagen.flow(x,batch_size=1):
    pred = model.predict(batch)
    if pred >0.5:
        text_labels.append('High')
    else:
        text_labels.append('Low')
    plt.subplot(5/columns + 1, columns, i+1)
    plt.title('This is a ' + text_labels[i])
    imgplot = plt.imshow(batch[0])
    i += 1
    if i % 10 == 0:
        break
    plt.show()
```

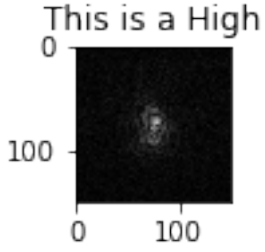

This is a High
0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140

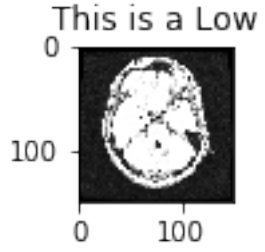

This is a High
0
100
0
100

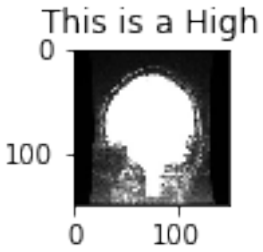

This is a Low
0
100
0
100

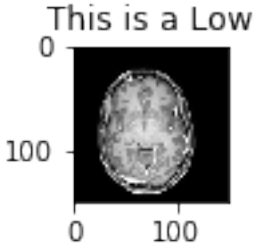

This is a High
0
100
0
100

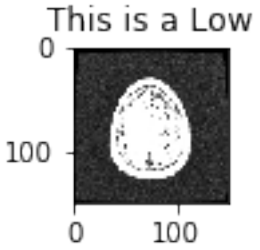

This is a Low
0
100
0
100


This is a High
0
100
0
100


This is a Low
0
100
0
100


This is a High
0
100
0
100


This is a Low
0
100
0
100


This is a Low
0
100
0
100