Deep Learning – Case Study

Title: Tumor classification

Name: Kantesariya Darshan

Enrollment Number: 18012011033

Batch: DL1

Model: Sequntial Model

1. Introduction

In this Keras project, we will build and train convolutional neural network for classifying images of Tumor. The input image will be analysed and then the output is predicted. The model that is implemented can be extended to a website or any mobile device as per the need. The brain_tumor dataset used in this project is available on Kaggle.

2. Tools & Libraries

In this topic we are going to see about tools and libraries that I am using to develop the project.

No	Tools & Library Name	Usage
1	os	os provides a portable way of using operating system-dependent functionality. The os and os.path modules include many functions to interact with the file system.
2	keras	Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load.
3	PIL	PIL provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.
4	Numpy	NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more
5	Pandas	pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
6	Maplotlib	Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python
7	Sklearn	Sklearn has Simple and efficient tools for predictive data analysis, accessible to everybody, and reusable in various contexts built on NumPy, SciPy, and matplotlib.

3. Architecture and Dataset

- **Architecture: -** Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 32)      416

conv2d_1 (Conv2D)            (None, 128, 128, 32)      4128

batch_normalization (BatchN  (None, 128, 128, 32)      128
ormalization)

max_pooling2d (MaxPooling2D  (None, 64, 64, 32)        0
)

dropout (Dropout)            (None, 64, 64, 32)        0

conv2d_2 (Conv2D)            (None, 64, 64, 64)        8256

conv2d_3 (Conv2D)            (None, 64, 64, 64)        16448

batch_normalization_1 (Batc  (None, 64, 64, 64)        256
hNormalization)

max_pooling2d_1 (MaxPooling  (None, 32, 32, 64)        0
2D)

dropout_1 (Dropout)          (None, 32, 32, 64)        0

flatten (Flatten)            (None, 65536)             0

dense (Dense)                (None, 512)               33554944

dropout_2 (Dropout)          (None, 512)               0
```

```
dense_1 (Dense)              (None, 2)                 1026

=================================================================
Total params: 33,585,602
Trainable params: 33,585,410
Non-trainable params: 192

_________________________________________________________________
None
```

- **Dataset: -** In this project we are using the "brain_tumor_dataset" dataset. This dataset have 2 folders, and almost 100 images. In this dataset two folders name is yes and no. in yes folder we have all tumor positive x-rays and no folder have all tumor negative x-rays. That means we can easily identify x-ray and show positive or negative output.

  https://drive.google.com/drive/folders/1Q2wk6OQdAlLdUF3f2HiJ4p87tWZBbAEP?usp=sharing

4. Code

Code is open source and published into GitHub with read-me file.

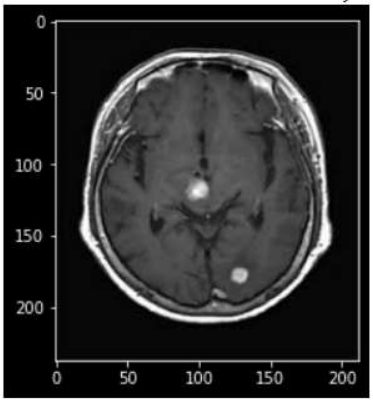
https://github.com/darshan-spyder/tumor_classification

5. Output

Some screenshots of our working application.

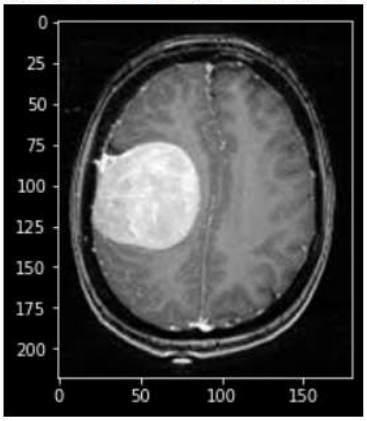
```
from matplotlib.pyplot import imshow
img = Image.open(r"/content/drive/MyDrive/brain_tumor_dataset/no/N16.jpg")
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
res = model.predict_on_batch(x)
classification = np.where(res == np.amax(res))[1][0]
imshow(img)
print(str(res[0][classification]*100) + '% Confidence This Is ' + names(classification))
```

```
100.0% Confidence This Is No, Its not a tumor
```



```
from matplotlib.pyplot import imshow
img = Image.open(r"/content/drive/MyDrive/brain_tumor_dataset/yes/Y1.jpg")
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
res = model.predict_on_batch(x)
classification = np.where(res == np.amax(res))[1][0]
imshow(img)
print(str(res[0][classification]*100) + '% Confidence This Is A ' + names(classification))
```

```
100.0% Confidence This Is A Its a Tumor
```



Conclusion – we learn about tumor classification means in any x-ray have tumor than output show positive othervise output is negative.