

In [1]:



```
1 # Import necessary modules
2
3 from tensorflow.keras.preprocessing.image import ImageDataGenerator
4 from tensorflow.keras.applications import VGG16
5 from tensorflow.keras.layers import Input
6 from tensorflow.keras.layers import Dense
7 from tensorflow.keras.layers import AveragePooling2D
8 from tensorflow.keras.layers import Dropout
9 from tensorflow.keras.layers import Flatten
10 from tensorflow.keras.models import Model
11 from tensorflow.keras.optimizers import Adam
12 from tensorflow.keras.utils import to_categorical
13
14 from sklearn.preprocessing import LabelBinarizer
15 from sklearn.model_selection import train_test_split
16 from sklearn.metrics import classification_report, confusion_matrix
17
18 from imutils import paths
19 import numpy as np
20 import matplotlib.pyplot as plt
21 import argparse
22 import os
23 import cv2
```

In [2]:



```
1 # Load the images directories
2 path = "D:/brain_tumor_dataset"
3 print(os.listdir(path))
4
5 image_paths = list(paths.list_images(path))
6 print(len(image_paths))
```

['no', 'yes']

253

In [3]:

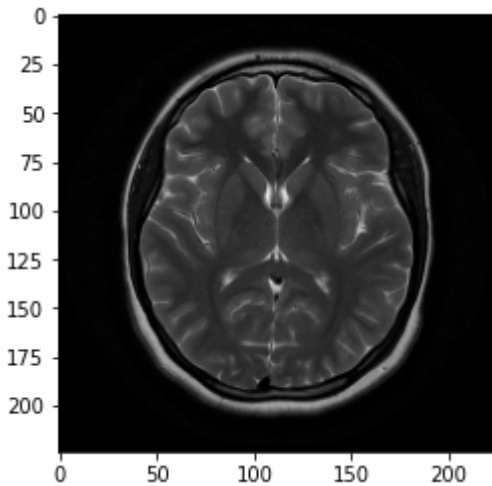


```
1 #
2 images = []
3 labels = []
4
5 for image_path in image_paths:
6     label = image_path.split(os.path.sep)[-2]
7     image = cv2.imread(image_path)
8     image = cv2.resize(image, (224, 224))
9
10     images.append(image)
11     labels.append(label)
```

In [4]:



```
1 # Plot an image
2 def plot_image(image):
3     plt.imshow(image)
4
5 plot_image(images[0])
```



In [5]:



```
1 # Convert into numpy arrays
2 images = np.array(images) / 255.0
3 labels = np.array(labels)
```

In [6]:



```
1 # Perform One-hot encoding
2 label_binarizer = LabelBinarizer()
3 labels = label_binarizer.fit_transform(labels)
4 labels = to_categorical(labels)
5
6 print(labels[0])
```

[1. 0.]

In [7]:



```
1 #Split the dataset
2 (train_X, test_X, train_Y, test_Y) = train_test_split(images,
3                                                         labels,
4                                                         test_size= 0.10,
5                                                         random_state= 42, stratify= 1
```

In [8]:

```
1 # Build the Image Data Generator
2 train_generator = ImageDataGenerator(fill_mode= 'nearest',
3                                     rotation_range= 15)
```

In [9]:

```
1 # Build the model
2 base_model = VGG16(weights= 'imagenet',
3                       input_tensor= Input(shape = (224, 224, 3)),
4                       include_top= False)
5 base_input = base_model.input
6 base_output = base_model.output
7 base_output = AveragePooling2D(pool_size=(4, 4))(base_output)
8 base_output = Flatten(name="flatten")(base_output)
9 base_output = Dense(64, activation="relu")(base_output)
10 base_output = Dropout(0.5)(base_output)
11 base_output = Dense(2, activation="softmax")(base_output)
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5 (https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5)

58892288/58889256 [=====] - 35s 1us/step

In [10]:

```
1 # Freeze the layers
2 for layer in base_model.layers:
3     layer.trainable = False
```

In [11]:

```
1 # Compile the model
2 model = Model(inputs = base_input, outputs = base_output)
3 model.compile(optimizer= Adam(learning_rate= 1e-3),
4               metrics= ['accuracy'], loss= 'binary_crossentropy')
```

In [12]:



```
1 # Let's see the architecture summary of our model
2 model.summary()
```

Model: "model"

Layer (type)	Output Shape	Param #
=====		
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
average_pooling2d (AveragePo	(None, 1, 1, 512)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 64)	32832
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 2)	130
=====		
Total params: 14,747,650		
Trainable params: 32,962		

Non-trainable params: 14,714,688

In [13]:

```
1 batch_size = 8
2 train_steps = len(train_X) // batch_size
3 validation_steps = len(test_X) // batch_size
4 epochs = 10
```

In [14]:

```
1 # Fit the model
2 history = model.fit_generator(train_generator.flow(train_X,
3                                                    train_Y,
4                                                    batch_size = batch_size),
5                               steps_per_epoch= train_steps,
6                               validation_data = (test_X, test_Y),
7                               validation_steps= validation_steps,
8                               epochs= epochs)
```

WARNING:tensorflow:From <ipython-input-14-261dfbd5bc3e>:2: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version.

Instructions for updating:

Please use Model.fit, which supports generators.

Epoch 1/10

28/28 [=====] - 82s 3s/step - loss: 0.6881 - accuracy: 0.6164 - val_loss: 0.6093 - val_accuracy: 0.6154

Epoch 2/10

28/28 [=====] - 74s 3s/step - loss: 0.6869 - accuracy: 0.5845 - val_loss: 0.5695 - val_accuracy: 0.6154

Epoch 3/10

28/28 [=====] - 81s 3s/step - loss: 0.6073 - accuracy: 0.6804 - val_loss: 0.5344 - val_accuracy: 0.8462

Epoch 4/10

28/28 [=====] - 78s 3s/step - loss: 0.6101 - accuracy: 0.6530 - val_loss: 0.4963 - val_accuracy: 0.9231

Epoch 5/10

28/28 [=====] - 76s 3s/step - loss: 0.5888 - accuracy: 0.6758 - val_loss: 0.4408 - val_accuracy: 0.9231

Epoch 6/10

28/28 [=====] - 79s 3s/step - loss: 0.5536 - accuracy: 0.7671 - val_loss: 0.4115 - val_accuracy: 0.9231

Epoch 7/10

28/28 [=====] - 77s 3s/step - loss: 0.5209 - accuracy: 0.7580 - val_loss: 0.3826 - val_accuracy: 0.9615

Epoch 8/10

28/28 [=====] - 79s 3s/step - loss: 0.5578 - accuracy: 0.7123 - val_loss: 0.3463 - val_accuracy: 0.9231

Epoch 9/10

28/28 [=====] - 69s 2s/step - loss: 0.5187 - accuracy: 0.7489 - val_loss: 0.3621 - val_accuracy: 0.9231

Epoch 10/10

28/28 [=====] - 70s 3s/step - loss: 0.4823 - accuracy: 0.7812 - val_loss: 0.3307 - val_accuracy: 0.9615

In [15]:



```

1 # Evaluate the model
2 predictions = model.predict(test_X, batch_size= batch_size)
3 predictions = np.argmax(predictions, axis= 1)
4 actuals = np.argmax(test_Y, axis= 1)

```

In [16]:



```

1 # Print Classification report and Confusion matrix
2 print(classification_report(actuals, predictions,
3                             target_names= label_binarizer.classes_))
4
5 cm = confusion_matrix(actuals, predictions)
6 print(cm)

```

	precision	recall	f1-score	support
no	0.91	1.00	0.95	10
yes	1.00	0.94	0.97	16
accuracy			0.96	26
macro avg	0.95	0.97	0.96	26
weighted avg	0.97	0.96	0.96	26

```

[[10  0]
 [ 1 15]]

```

In [17]:



```

1 # Final accuracy of our model
2 total = sum(sum(cm))
3 accuracy = (cm[0, 0] + cm[1, 1]) / total
4 print("Accuracy: {:.4f}".format(accuracy))

```

Accuracy: 0.9615

In [18]:

```
1 # Plot the losses and accuracies
2 N = epochs
3 plt.style.use("ggplot")
4 plt.figure()
5 plt.plot(np.arange(0, N), history.history["loss"],
6          label= "train_loss")
7 plt.plot(np.arange(0, N), history.history["val_loss"],
8          label= "val_loss")
9
10 plt.plot(np.arange(0, N), history.history["accuracy"],
11          label= "train_acc")
12 plt.plot(np.arange(0, N), history.history["val_accuracy"],
13          label= "val_acc")
14
15 plt.title("Training Loss and Accuracy on Brain Dataset")
16 plt.xlabel("Epoch")
17 plt.ylabel("Loss / Accuracy")
18 plt.legend(loc= "lower left")
19 plt.savefig("plot.jpg")
```

