UNIVERSITY PARTNER
UNIVERSITY OF WOLVERHAMPTON
iAcademy
Quality Education Improving Lives

HERALD COLLEGE
KATHMANDU

Artificial Intelligence and Machine learning

(6CS012)

Brain Tumor Classification

Full Name : Yogesh Shrestha

Student Number : NP03A190299 (2050215)

Award/Course : BSC (hons)

University Id : unv\2050214

Group : L6CG8

Date of Submission : 05/02/2022

Abstract

The brain tumor classification is used using convolutional neural network (CNN) technique. Its objective is to predict the images of a brain tumor using CNN algorithm with best technique MIR (Magnetic Resonance Imagining). In a stage of model's training the EfficientNetBO model was used and the 3 layers is used i.e. GlobalAveragePooling2D, DropOut, and Dense to gather weighted input data, perform a transformation of data and output data for comming layer. The hyper parameter of Dropout layer is set as rate:0.5 and Dense is set as activation: softmax. After optimizing model, to find the error categorical_crossentropy is used. Using 12 epoch, the lowest loss value, highest accuracy is found. And then , after using confusion matrix on different tumor types the image is successfully classified.

Contents

1: Introduction

About Data

Brain tumor is a disease that seen either they are children or adult. Brain tumor can be characterized as Benign Tumor, Malignant Tumor, Pituitary Tumor, etc.

The data in the directory is from an MRI scan. The photographs have already been divided into two directories: Training and Testing. There are 4 subdirectories in each directory. These directories contain MRIs of various tumor classes.

Aims and Objectives

- To classify the brain tumor symptoms.
- To locate the objects of the interest in an MRI image before using a basic CNN model by collecting the characteristics of the data after applying edge detection.
- To extract the characteristics of the image using CNN algorithm.
- To identify the brain tumor type using best technique which is MIR (Magnetic Resonance Imagining).

CNN

A convolutional neural network (CNN) is a type of neural network that has one or many convolutional layers that is used to handle image files, classifying the objects, and analyze correlation relevant information. Convolution, pooling, and fully linked networks are the three types of networks found in CNN. Typically, CNN is used to interpret Natural Language Processing, which aids in the smooth execution of tasks such as image categorization and feature extraction. Handwriting detection and digit classifications, for example, can be classified swiftly. Using CNN (Convolution Neural Network) to classify help to increase the performance of detection. (Yamashita, 2018)

The process of classifying image according to its groups, labeled, types etc. is known as image classification. for example, classifying image of dog and cat whether it is dog or cat then categorizing into dog group if it is dog, or into cat group if it is cat.

2: Methodology

Model Summary

The Transfer learning is used to train models in order to achieve extremely exact outcomes. While training the CNN deep network model it may takes long time in training each dataset. The EfficientNetBO model is used to train model. It takes weights of the ImageNet dataset. To construct our own final output based on our use cases, the top parameter is applied as false. There are many call back actions were used i.e. ModelCheckpotint, TensorBoard.

layers

There are 3 layers that were used in this model i.e. GlobalAveragePooling2D, Dropout, Dense. GlobalAveragePooling2D layer works in the same way to the Max Pooling in CNNs, with an exception that something in pools using Mean scores rather than Max values. It dramatically allows to increase the computational loading speed while training data.

Dropout this layer eliminates most of the neurons from the layers in every phase, allowing the neurons to be more independently. It reduces problems like overfitting. The rates variable is the probability of a neuron activation having adjusted to 0 and hence the neuron getting removed away.

Dense is the output layer that separates the photo into 1 of 4 categories. It involves use of the softmax layer, and that is a sigmoid function generalization.

Hyperparameter tuning

For the hyper parameter of the Dropout model rate is set as 0.5. for the hyper parameter of the Dense layer, activation is set as softmax.

Training the models

To reduce the algorithm's errors while optimization process of Neural Network like; gradient descent, loss function is used. It used to analyzed how well or poorly is performing in the model.

In our model training, categorical_crossentropy was used as loss function.

To optimize the algorithm, Adam was used as optimizer.

The 12 Epoch were used to train the model.

The extraction of image training and validation of loss against iteration is shown in bellow image.

```
]: CNN = model.fit(X_train,y_train,validation_split=0.1, epochs =12, verbose=1, batch_size=32,
                   callbacks=[tensorboard,checkpoint,reduce_lr])

Epoch 1/12
83/83 [==============================] - ETA: 0s - loss: 0.4697 - accuracy: 0.8157
Epoch 1: val_accuracy improved from -inf to 0.88776, saving model to effnet.h5
83/83 [==============================] - 151s 2s/step - loss: 0.4697 - accuracy: 0.8157 - val_loss: 0.4391 - val_accuracy: 0.88
78 - lr: 0.0010
Epoch 2/12
83/83 [==============================] - ETA: 0s - loss: 0.1931 - accuracy: 0.9330
Epoch 2: val_accuracy did not improve from 0.88776
83/83 [==============================] - 137s 2s/step - loss: 0.1931 - accuracy: 0.9330 - val_loss: 0.5545 - val_accuracy: 0.85
37 - lr: 0.0010
Epoch 3/12
83/83 [==============================] - ETA: 0s - loss: 0.1124 - accuracy: 0.9667
Epoch 3: val_accuracy improved from 0.88776 to 0.92177, saving model to effnet.h5
83/83 [==============================] - 143s 2s/step - loss: 0.1124 - accuracy: 0.9667 - val_loss: 0.3042 - val_accuracy: 0.92
18 - lr: 0.0010
Epoch 4/12
83/83 [==============================] - ETA: 0s - loss: 0.0975 - accuracy: 0.9690
Epoch 4: val_accuracy did not improve from 0.92177
83/83 [==============================] - 143s 2s/step - loss: 0.0975 - accuracy: 0.9690 - val_loss: 0.2925 - val_accuracy: 0.92
18 - lr: 0.0010
Epoch 5/12
83/83 [==============================] - ETA: 0s - loss: 0.0722 - accuracy: 0.9750
Epoch 5: val_accuracy did not improve from 0.92177

Epoch 5: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
83/83 [==============================] - 142s 2s/step - loss: 0.0722 - accuracy: 0.9750 - val_loss: 0.3225 - val_accuracy: 0.90
48 - lr: 0.0010
Epoch 6/12
83/83 [==============================] - ETA: 0s - loss: 0.0273 - accuracy: 0.9909
Epoch 6: val_accuracy improved from 0.92177 to 0.97279, saving model to effnet.h5
83/83 [==============================] - 147s 2s/step - loss: 0.0273 - accuracy: 0.9909 - val_loss: 0.1402 - val_accuracy: 0.97
28 - lr: 3.0000e-04
```

In this image, half of the epoch result is shown.

```
Epoch 7/12
83/83 [==============================] - ETA: 0s - loss: 0.0142 - accuracy: 0.9958
Epoch 7: val_accuracy improved from 0.97279 to 0.97619, saving model to effnet.h5
83/83 [==============================] - 146s 2s/step - loss: 0.0142 - accuracy: 0.9958 - val_loss: 0.1038 - val_accuracy: 0.97
62 - lr: 3.0000e-04
Epoch 8/12
83/83 [==============================] - ETA: 0s - loss: 0.0110 - accuracy: 0.9962
Epoch 8: val_accuracy did not improve from 0.97619
83/83 [==============================] - 146s 2s/step - loss: 0.0110 - accuracy: 0.9962 - val_loss: 0.0884 - val_accuracy: 0.97
28 - lr: 3.0000e-04
Epoch 9/12
83/83 [==============================] - ETA: 0s - loss: 0.0066 - accuracy: 0.9985
Epoch 9: val_accuracy did not improve from 0.97619

Epoch 9: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
83/83 [==============================] - 146s 2s/step - loss: 0.0066 - accuracy: 0.9985 - val_loss: 0.0946 - val_accuracy: 0.97
28 - lr: 3.0000e-04
Epoch 10/12
83/83 [==============================] - ETA: 0s - loss: 0.0066 - accuracy: 0.9977
Epoch 10: val_accuracy improved from 0.97619 to 0.98299, saving model to effnet.h5
83/83 [==============================] - 147s 2s/step - loss: 0.0066 - accuracy: 0.9977 - val_loss: 0.0860 - val_accuracy: 0.98
30 - lr: 9.0000e-05
Epoch 11/12
83/83 [==============================] - ETA: 0s - loss: 0.0055 - accuracy: 0.9985
Epoch 11: val_accuracy did not improve from 0.98299
83/83 [==============================] - 146s 2s/step - loss: 0.0055 - accuracy: 0.9985 - val_loss: 0.0878 - val_accuracy: 0.97
96 - lr: 9.0000e-05
Epoch 12/12
83/83 [==============================] - ETA: 0s - loss: 0.0075 - accuracy: 0.9970
Epoch 12: val_accuracy improved from 0.98299 to 0.98639, saving model to effnet.h5
83/83 [==============================] - 144s 2s/step - loss: 0.0075 - accuracy: 0.9970 - val_loss: 0.0829 - val_accuracy: 0.98
64 - lr: 9.0000e-05
```

In this image, second half of the epoch result is shown

Each loss function is decrease as applying epoch and the best accuracy show 0.9970. the epoch has val_loss 0.0829, val_accuracy 0.98. and it have lowest loss which is 0.0075.

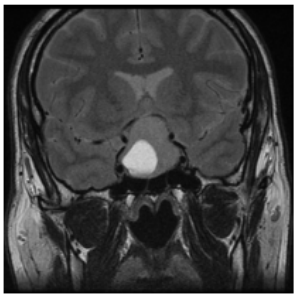
Result and Finding

For evaluation metrics, the sklearn's confusion_matrix was used to analyzed the performance models. I enable to compare between the original goal of the dataset's values and the predicted values . Confusion matrix was applied to Glioma Tumor, No Tumor, Meningioma Tumor, and Pituitary Tumor.

In [47]:
```
# Uploader button from widgets.
uploader = widgets.FileUpload()
display(uploader)
```

Upload (1)

In [50]:
```
# Displaying the uploaded Image.
for name, file_info in uploader.value.items():
    img = Image.open(io.BytesIO(file_info['content']))
scale = 0.2
display(img.resize(( int(img.width * scale), int(img.height * scale))))
```



In [51]:
```
# Predict Button for doing the prediction.
button = widgets.Button(description='Predict')
out = widgets.Output()

# Function from where the img_pred function is called when clicking the predict button.
def on_button_clicked(_):
    with out:
        clear_output()
        try:
            img_pred(uploader)

        except:
            print('No Image Uploaded.')
button.on_click(on_button_clicked)
widgets.VBox([button,out])
```

Predict

```
The Model predicts that the tumor type is a Pituitary Tumor
```

In this figure, a random image is selected to predict. After predicting image, it predicts it has pituitary tumor.

References

Yamashita, R. (2018). *Convolutional neural networks: an overview and application in radiology.* SpringerOpen.