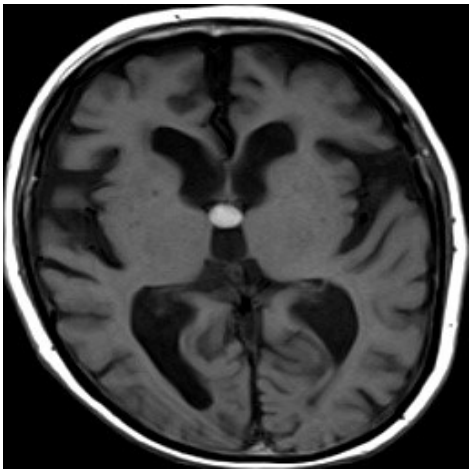
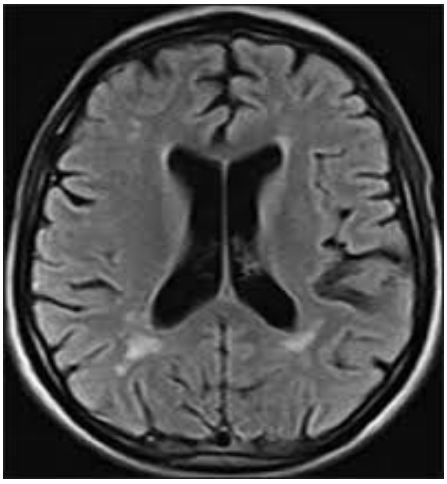
Brain Tumour Detection

using

Convolutional Neural Network



Brain Tumour



Healthy Brain

Report Prepared by

Anant Tyagi

Abstract:

This study presents a deep learning approach for the automatic detection of brain tumours using Convolutional Neural Networks (CNNs). The proposed model was trained on a large dataset of brain MRI images, and it was able to accurately identify the presence of tumours in new unseen images. The CNN architecture used in this study consisted of multiple convolutional layers, followed by pooling and fully connected layers. The model was trained using a combination of supervised learning and data augmentation techniques to increase its robustness to different types of brain tumours. The results showed that the proposed approach achieved high accuracy, sensitivity, and specificity, indicating its potential as a reliable tool for assisting radiologists in the diagnosis of brain tumours. This study demonstrates the potential of deep learning methods to improve the accuracy and efficiency of brain tumour detection, which could ultimately lead to better patient outcomes.

Dataset:

https://www.kaggle.com/datasets/preetviradiya/brian-tumor-dataset

Brain Tumor Dataset(2 directory)

- Brain Tumour
- Healthy

Context

This dataset consists of the scanned images of brain of patient diagnosed of brain tumour.

Introduction:

One of the essential organs in the human body, the brain has billions of cells. Uncontrolled cell division results in the formation of an aberrant cell group, generally known as a tumour. There are two types of brain tumours: low grade (grades 1 and 2) and high grade (grades 3 and 4). Benign brain tumours are those that are low grade. A high-grade tumour is referred to as malignant in a similar way. Cancerous tumours are not benign tumours. As a result, it doesn't spread to other brain regions. The cancerous tumour, however, is a malignant tumour. As a result, it easily spreads to other regions of the body and spreads quickly with no restrictions. That results in instant death.

The primary applications of brain MRI images are tumour detection and tumour progression modelling. The main applications of this knowledge are in the diagnosis and treatment of tumours. Compared to a CT or ultrasound image, an MRI scan provides more information about a specific medical image. An MRI picture can be used to detect anomalies in the brain tissue and provide extensive information on brain structure.

Now since the detection of tumour can be cured at the initial stages so we are trying to develop a model using Convolutional Neural Network and Transfer Learning techniques to automate prediction of a brain tumour using the MRI image.

Implementation:

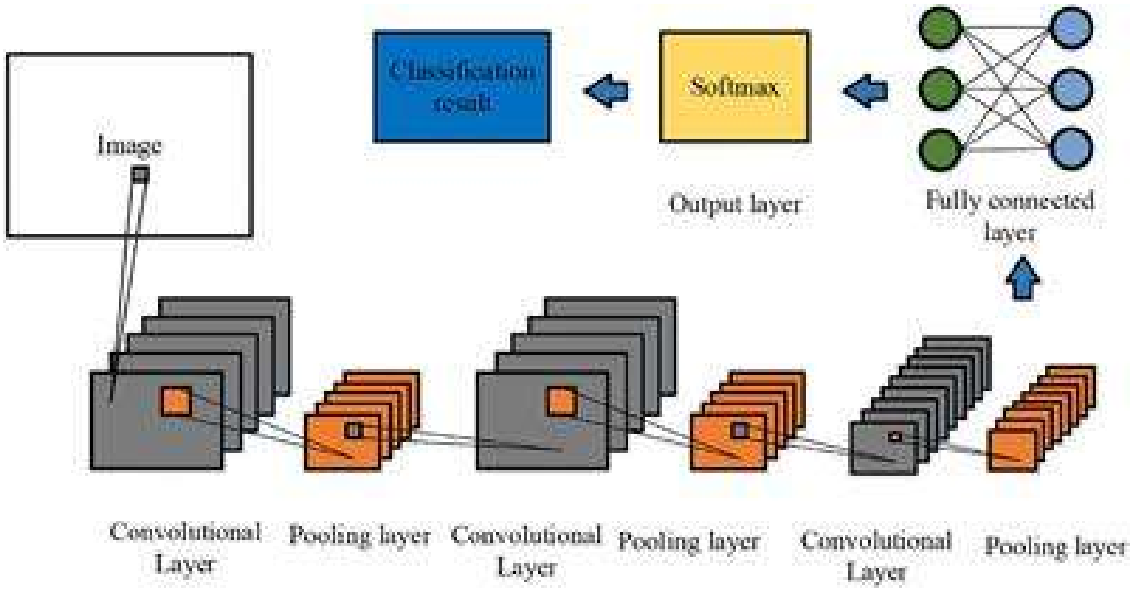
Packages/Tools used:

1. NumPy: To calculate various calculations related to arrays
2. Os: The OS module in Python provides functions for interacting with the operating system
3. Keras.layers: To import all the layers required
4. Keras.model: To import the models
5. Matplotlib: To plot the graphs
6. Shutil: Transfer the data from root directory to respective directory

What is CNN?

A Convolutional Neural Network (CNN) is a type of deep learning algorithm that excels at image recognition and processing. It has several layers, including convolutional layers, pooling layers, and fully connected layers.

The key component of a CNN is the convolutional layers, which apply filters to the input image to extract features such as edges, textures, and shapes. The convolutional layers' output is then passed through pooling layers, which are used to down-sample the feature maps, reducing the spatial dimensions while retaining the most important information. The pooling layers' output is then passed through one or more fully connected layers, which are used to predict or classify the image.



CNN are used for images because of their ability to extract local features, hierarchical representation, parameter sharing, pooling and striding, and high accuracy, CNNs are ideal for image processing. Convolutional layers are used to capture local features, hierarchical representations for progressively complex features are learned, parameters are shared for efficient memory usage, and pooling and striding are used for robustness. CNNs have been shown to achieve state-of-the-art image recognition performance by learning from large datasets, making them well-suited for image processing and achieving high accuracy in tasks such as image classification, object detection, and image segmentation.

Methodology:

Step1 Import all the dependencies required for the project.

Step2 Pre-Processing the Dataset:

The given dataset is initially present in the dataset repository and then we use os module to train set, test set and validation sets.The distribution is as follows:

- Training Set=70%
- Test Set= 15%
- Validation Set=15%

Step3 Model Building:

We are trying to develop a sequential model based on Convolutional Neural Network for that we use

```
#Model Building
from keras.layers import Conv2D,MaxPool2D,Dropout,Flatten,Dense,BatchNormalization,GlobalAvgPool2D
from keras.models import Sequential
from keras_preprocessing.image import load_img,img_to_array
```

✓ 0.0s

Model Structure

```
#CNN MODEL
model=Sequential()
model.add(Conv2D(filters=16,kernel_size=(3,3),activation='relu',input_shape=(224,224,3)))
model.add(Conv2D(filters=36, kernel_size=(3,3),activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Conv2D(filters=64, kernel_size=(3,3),activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Conv2D(filters=128, kernel_size=(3,3),activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(units=64,activation='relu'))
model.add(Dropout(rate=0.25))
model.add(Dense(units=1,activation='sigmoid'))
model.summary()
```

✓ 1.6s

Step4 Compile The model:

```
model.compile(optimizer='adam',loss=keras.losses.binary_crossentropy, metrics=['accuracy'])
```

✓ 0.2s

Step5 Model Training:

Before beginning the model training, we need to convert the training set into the way it is to be fed to the model. So, we use image generator to convert the images to the model understandable form.

We also use Early Stopping and model checkpoint to stop the model from over training and preventing over use of resources.

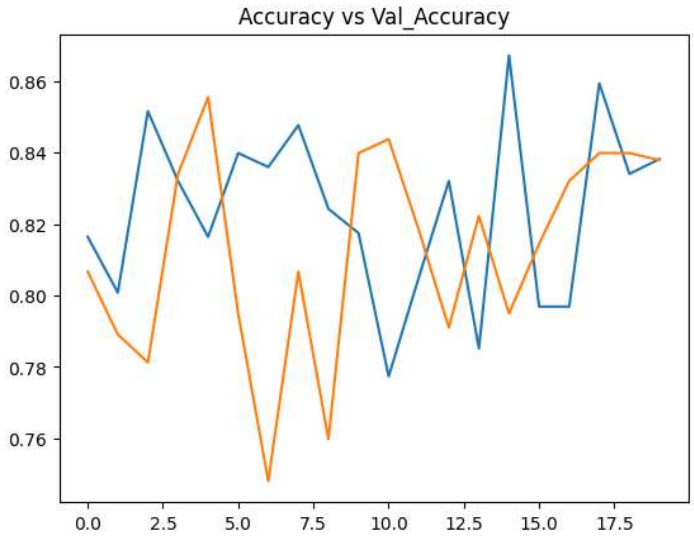
```
#Early Stopping and model checkpoint
from keras.callbacks import ModelCheckpoint,EarlyStopping
#early stopping
es=EarlyStopping(monitor='val_accuracy',min_delta=0.01, patience=15,verbose=1, mode='auto')
#Model Checkpoint
mc=ModelCheckpoint(monitor='val_accuracy',filepath="./bestmodel.h5", verbose=1,save_best_only=True,mode='auto')

cd=[es,mc]
```
✓ 0.0s

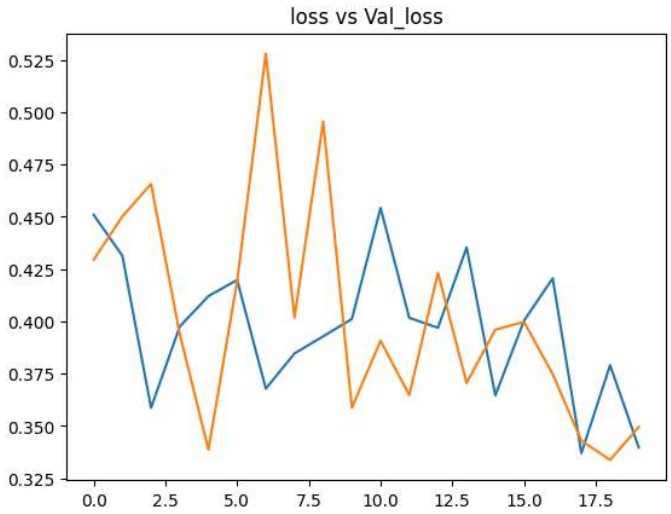
Then we finally train the model

```
#Model Training
hs=model.fit_generator(generator=train_data,steps_per_epoch=8, epochs=50,verbose=1,validation_data=validation_data,validation_steps=16,
                       callbacks=cd)
```

Tracing the accuracy and losses of this CNN model



X-axis No. of epochs and Y-axis represent the accuracy



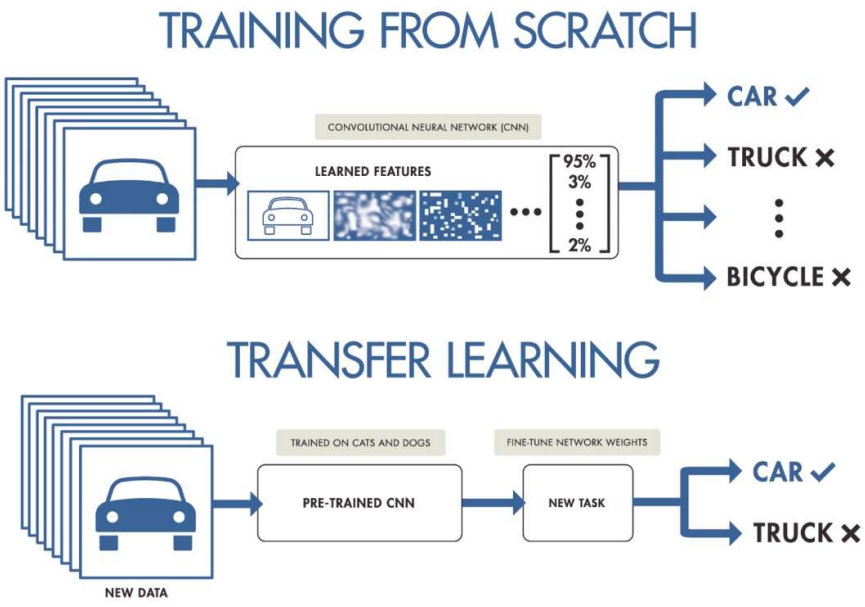
X-axis No. of epochs and Y-axis represent the losses

Accuracy: The current accuracy of the model is 83.80809426307678

Since, the accuracy of the current model is very low so we would use transfer learning to improve the accuracy of this model.

What is Transfer Learning?

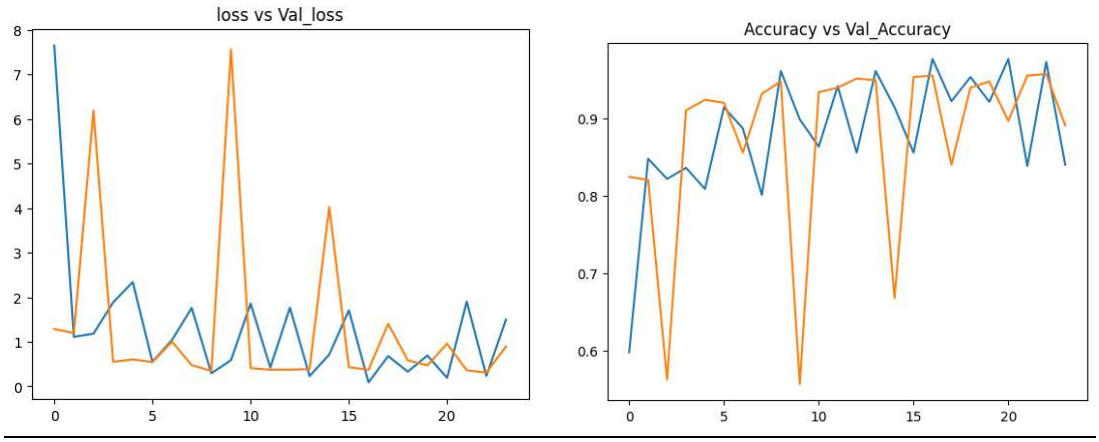
Transfer learning is a **research problem in machine learning** that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. This technique is applicable to many machine learning models, including deep learning models like artificial neural networks and reinforcement models.



Step 6 Transfer Learning:

We use inbuilt CNN architecture to transfer learn the model. Here we have used mobilenet to make it compatible with Mobile phones

This raises the accuracy to 95.652174949646



Tracing the accuracy and losses of this CNN model

To run the bestmodel.h5 file

```
#  Transfer Model Accuracy
from keras.models import load_model
model_transfer=load_model("bestmodel.h5")
```

```
testing_path="D:\Internship Code Clause\Task1 Brain Tumour Detection\Dataset\Brain Tumor\Cancer (230).jpg"

img_load=load_img(testing_path,target_size=(224,224))
```

```
input_array=img_to_array(img_load)/255
input_array.shape
```

```
(224, 224, 3)
```

```
input_array=np.expand_dims(input_array,axis=0)
```

```
pred=model.predict(input_array)[0][0]
```

Conclusion:

Finally, the study presents a novel approach for brain tumour prediction using Convolutional Neural Networks (CNNs). The research demonstrates CNNs' potential as a powerful tool for accurate and efficient brain tumour classification from medical images. The findings show promising results in terms of accuracy, sensitivity, and specificity, indicating that the proposed method has the potential to detect and diagnose brain tumours early. The research paper also discusses how advanced image processing techniques and deep learning algorithms can improve the accuracy and reliability of brain tumour prediction. However, larger dataset studies and validations are required to establish the generalizability and robustness of the proposed CNN-based approach.Nonetheless, the study lays the groundwork for future advances in brain tumour prediction using CNNs, and it has the potential to have a significant impact on clinical practise by facilitating early detection and timely intervention for patients with brain tumours.

Github: https://github.com/Ananttyagi004/Brain_Tumour_Detection