Brain Tumor Classification of Magnetic Resonance Imaging (MRIs) Using Convolutional Neural Network(CNN) Deep Learning Algorithms

1.Introduction

Today, cancer is the leading cause of death with a rate of one in 6. The most common types are breast, lung, colon, rectum, and prostate cancers(World Health Organization(WHO), 2022).

Although the chance of getting brain and nervous-system-related cancer is not more than 1%, they are one of the most fatal cancers(Miller Kimberly, D., 2021). They are the the10th leading cause of death for men and women. In the U.S., the estimated number of deaths from primary cancerous brain and spinal tumors is 20,000. Worldwide, each year around 250,000 people die from primary cancerous brain and spinal tumors(cancer.net).

Early detection and rapid, accurate identification of the brain cancers can save so many lives. When the detection and identification is delayed or not made, the chance of getting cured is lower and the costs identified with the cure is higher. Accurate and fast detection with robust screening would eliminate the personal, societal and economic cost of cancer.

Today, the most common method to detect cancer is the MRI(Magnetic Resonance Imaging) scans after a physiological exam. Other advanced diagnostic methods are CT(Computerized Tomography) and PET(Positron Emission Tomography) scan followed by a biopsy when needed.

Deep learning methods extract crucial features automatically. These approaches have yielded outstanding results in various application domains, e.g., pedestrian detection, speech recognition and understanding, and brain tumor segmentation.

2.Data

The data source is Kaggle. The data is collected by several students of PES Modern College of Engineering(Pune,India) for a college project. They used multiple sources and let a physician examine and verify the images before publishing the dataset.

The folder contains 3264 MRI images in jpg.format. The data is already cleaned and organized by the owner. The images are already split into training and testing folders. Each folder has more four subfolders. These folders have MRIs of respective tumor classes: glioma, meningioma, pituitary tumor as well as a folder with brain images without a tumor.

3.Problem Statement

Comparing the MRI, PET and CT images, radiologists can detect abnormal mass/tissue growth and cancer with the help of computer-aided tools. However, manual detection is a slow process with a high error rate(A Sravanthi Peddinti, 2021).

On the other hand, the recent developments in Machine Learning and Deep Learning, enables the AI to detect patterns in MRI's or CT images at a very early stage of cancer which are not even visible to human-eye(Etemadi Mozziyar, MD., PhD., 2020).

These deep learning algorithms are categorized into 3 main classes in the scientific literature(Ranjbarzadeh, Ramin et. al., 2021):

"Machine learning approaches address these problems by mainly using hand-crafted features (or pre-defined features). As an initial step in this kind of segmentation, the key information is extracted from the input image using some feature extraction algorithm, and then a discriminative model is trained to recognize the tumor from normal tissues. The designed machine learning techniques generally employ hand-crafted features with various classifiers, such as random forest10, support vector machine (SVM)fuzzy clustering.The designed methods and features extraction algorithms have to extract features, edge-related details, and other necessary information—which is time-consuming. Moreover, when boundaries between healthy tissues and tumors are fuzzy/vague, these methods demonstrate poorer performances.

Multi-atlas registration (MAS) algorithms are based on the registration and label fusion of multiple normal brain atlases to a new image modality4. Due to the difficulties in registering normal brain atlases and the need for a large number of atlases, these MAS algorithms have not been successfully dealing with applications that require speed.

Deep learning methods extract crucial features automatically. These approaches have yielded outstanding results in various application domains, e.g., pedestrian detection, speech recognition and understanding, and brain tumor segmentation."

Convolutional Neural Network(CNN), which is in the third category above, is a very effective supervised learning neural networks algorithm for computer vision and image understanding, it outperforms human accuracy in most cases.

The application of deep learning on cancer research may benefit:

1.Hospitals

2.Doctors

3.Patients

4.Medical Imaging Companies

5.Researchers, scientists

6.Academia

7.Healthcare startups(telemedicine etc.)

8.Governments

9.Insurance companies

10.DS community

4.Pre-Processing

Pre-processing of a CNN modeling starts with importing the necessary Python packages: numpy, pandas, sklearn, tensorflow, keras models, seaborn, matplotlib.

The next steop is encoding the labels of tumors as 0, 1, 2, 3.

As a next step, I used the Image data generator for data augmentation. There was no need to rescale the images.I manipulated the following features of an image:

-rotation_range: rotation degree,

-width_shift_range: horizontal shift,

-height_shift_range: vertical shift,

-zoom_range: zoom,

-horizontal_flip=True: horizontal flip active,

-brightness_range=[0.2,1.2]): brightness(1.0 is the neutral brightness, needed to go above 1.0 to brighten),

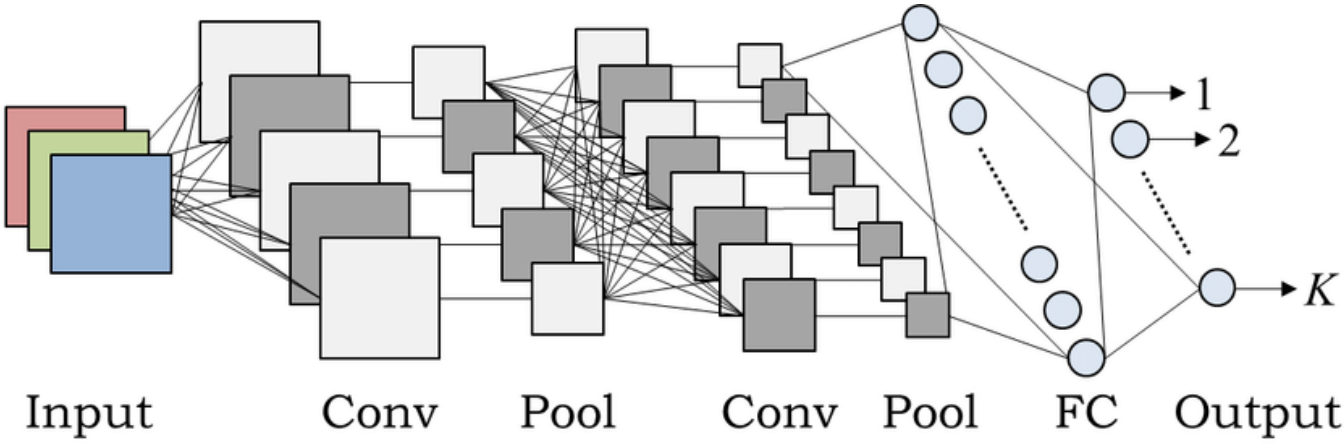
Following the augmentation, I splitted the dataset into training and test data.

5.Modeling

Convolutional Neural Network(CNN) is a very effective supervised learning and neural networks algorithm for computer vision and image understanding, it outperforms human accuracy in most cases.

What makes CNN special is that it is specifically designed to process images with the pixel format with a multi-layer perceptron, which means that the images pass through several back-to-back convolutional and pooling layers. Once the layers are fully connected, they end with an output layer which produces the probability of occurrence.

An example of CNN image processing as a diagram:



Sequential CNN Model

The structure where each layer connects only to the previous and following layers is called a "sequential CNN model". The Sequential model API is a way of creating deep learning models where an instance of the Sequential class is created and model layers are created and added to it.

To be able to create a complete layer, Keras needs the information about the shape of the input, number of neurons / units in the layer, initializers, regularizers, constraints, activations.

The common layers in a sequential CNN model architecture and their functions:

1.Conv2D(convolution layer): A filter or a kernel in a conv2D layer “slides” over the 2D input data, performing an elementwise multiplication. As a result, it will be summing up the results into a single output pixel. The kernel will perform the same operation for every location it slides over, transforming a 2D matrix of features into a different 2D matrix of features.

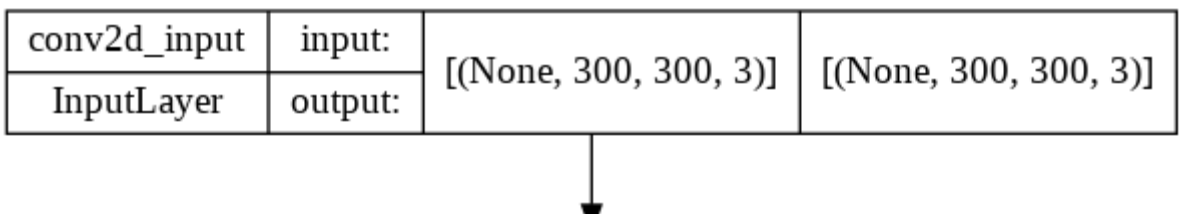
2.MaxPool2D layer: Applies a 2D max pooling(downsizing) over an input signal composed of several input planes, and takes the highest value.

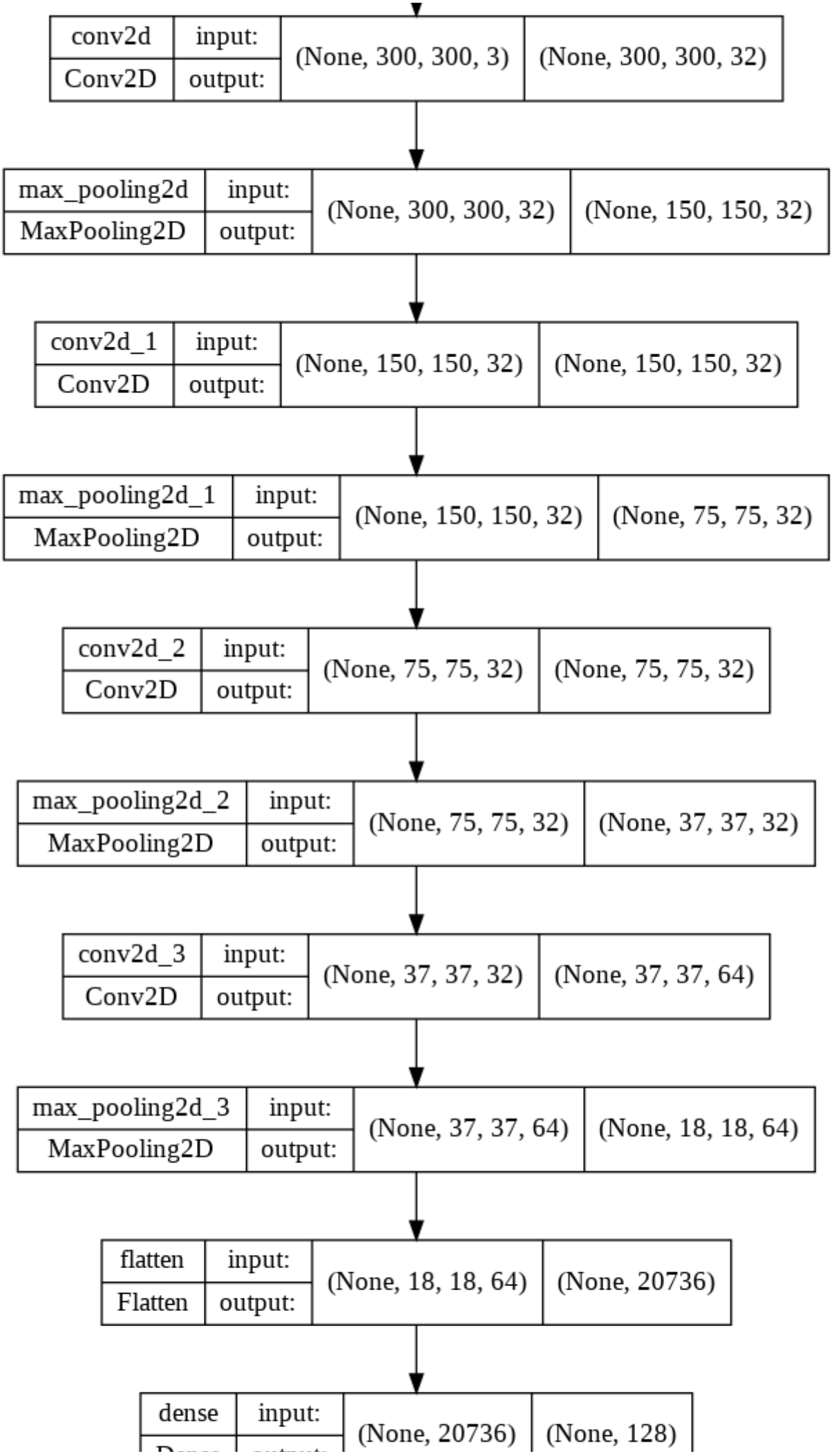
3.Flatten layer: Flattens the input, which returns a copy of the array in one dimensional rather than in 2-D or a multi-dimensional array.

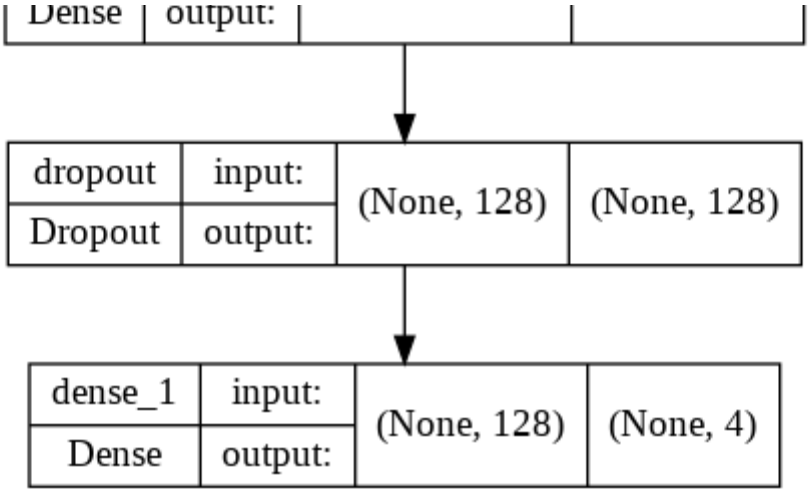
4.Dense(activation layer): Applies the sigmoid activation or relu. For small values (<-5), sigmoid returns a value close to zero, and for large values (>5) the result of the function gets close to 1(Note: Relu is faster than the sigmoid function).

5.Dropout layer(also called "dilution"): Dropout refers to randomly "dropping out", or excluding nodes during the training to regularize a neural network and prevent overfitting.

6.ReLU(Rectified Linear Activation Function): ReLU stands for “Rectified Linear Unit.” The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time.Mathematical formula is f(x)=max(0,x).




conv2d
Conv2D
input:
output:
(None, 300, 300, 3)
(None, 300, 300, 32)
max_pooling2d
MaxPooling2D
input:
output:
(None, 300, 300, 32)
(None, 150, 150, 32)
conv2d_1
Conv2D
input:
output:
(None, 150, 150, 32)
(None, 150, 150, 32)
max_pooling2d_1
MaxPooling2D
input:
output:
(None, 150, 150, 32)
(None, 75, 75, 32)
conv2d_2
Conv2D
input:
output:
(None, 75, 75, 32)
(None, 75, 75, 32)
max_pooling2d_2
MaxPooling2D
input:
output:
(None, 75, 75, 32)
(None, 37, 37, 32)
conv2d_3
Conv2D
input:
output:
(None, 37, 37, 32)
(None, 37, 37, 64)
max_pooling2d_3
MaxPooling2D
input:
output:
(None, 37, 37, 64)
(None, 18, 18, 64)
flatten
Flatten
input:
output:
(None, 18, 18, 64)
(None, 20736)
dense
input:
(None, 20736)
(None, 128)



6.Results

"Loss" refers to the loss value over the training data after each epoch. This is what the optimization process is trying to minimize with the training so, the lower, the better.

"Accuracy" refers to the ratio between correct predictions and the total number of predictions in the training data. The higher, the better. This is normally inversely correlated with the loss

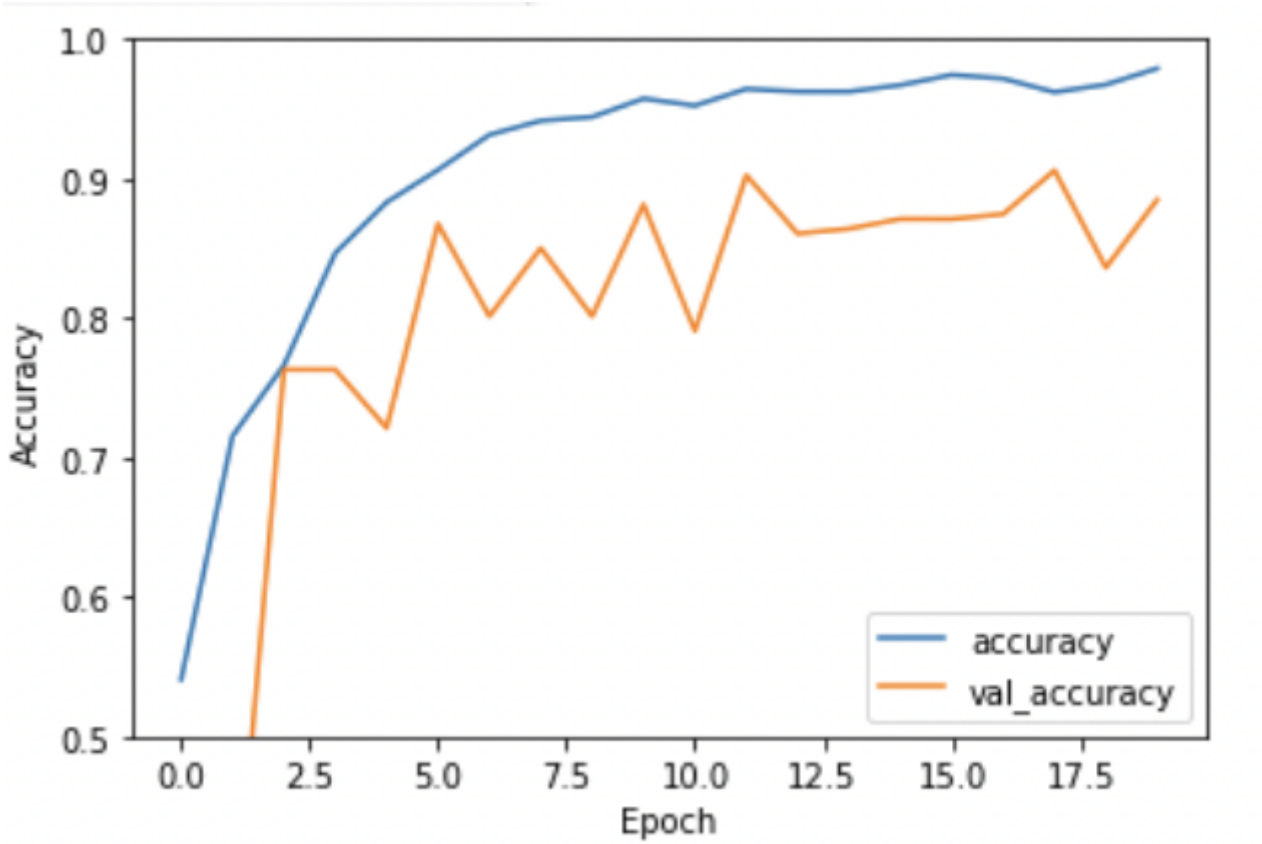
The "validation" is computed over the validation data, which is not used for training so "unseen" to your model. If the training loss and accuracy are good but the validation counterparts are bad, it is overfitting, as it can't generalize to unseen data.

```
Epoch 1/20
81/81 [==============================] - 24s 177ms/step - loss: 2.9952 - accuracy: 0.5408 - val_loss: 1.1610 - val_accuracy: 0.2300
Epoch 2/20
81/81 [==============================] - 12s 153ms/step - loss: 0.6817 - accuracy: 0.7154 - val_loss: 1.0648 - val_accuracy: 0.3380
Epoch 3/20
81/81 [==============================] - 12s 153ms/step - loss: 0.5599 - accuracy: 0.7662 - val_loss: 0.5260 - val_accuracy: 0.7631
Epoch 4/20
81/81 [==============================] - 12s 154ms/step - loss: 0.3911 - accuracy: 0.8467 - val_loss: 0.5835 - val_accuracy: 0.7631
Epoch 5/20
81/81 [==============================] - 12s 153ms/step - loss: 0.3249 - accuracy: 0.8831 - val_loss: 0.7780 - val_accuracy: 0.7213
Epoch 6/20
81/81 [==============================] - 12s 153ms/step - loss: 0.2577 - accuracy: 0.9063 - val_loss: 0.3786 - val_accuracy: 0.8676
Epoch 7/20
81/81 [==============================] - 12s 154ms/step - loss: 0.2017 - accuracy: 0.9315 - val_loss: 0.6202 - val_accuracy: 0.8014
Epoch 8/20
81/81 [==============================] - 13s 155ms/step - loss: 0.1574 - accuracy: 0.9415 - val_loss: 0.5790 - val_accuracy: 0.8502
Epoch 9/20
81/81 [==============================] - 12s 154ms/step - loss: 0.1432 - accuracy: 0.9443 - val_loss: 0.8409 - val_accuracy: 0.8014
Epoch 10/20
81/81 [==============================] - 13s 155ms/step - loss: 0.1202 - accuracy: 0.9574 - val_loss: 0.5037 - val_accuracy: 0.8815
Epoch 11/20
81/81 [==============================] - 12s 153ms/step - loss: 0.1417 - accuracy: 0.9524 - val_loss: 0.7585 - val_accuracy: 0.7909
Epoch 12/20
81/81 [==============================] - 12s 154ms/step - loss: 0.0991 - accuracy: 0.9644 - val_loss: 0.4188 - val_accuracy: 0.9024
Epoch 13/20
81/81 [==============================] - 12s 153ms/step - loss: 0.0985 - accuracy: 0.9624 - val_loss: 0.6095 - val_accuracy: 0.8606
Epoch 14/20
81/81 [==============================] - 13s 155ms/step - loss: 0.0967 - accuracy: 0.9624 - val_loss: 0.6721 - val_accuracy: 0.8641
Epoch 15/20
81/81 [==============================] - 12s 153ms/step - loss: 0.0857 - accuracy: 0.9671 - val_loss: 0.7784 - val_accuracy: 0.8711
Epoch 16/20
81/81 [==============================] - 12s 154ms/step - loss: 0.0798 - accuracy: 0.9744 - val_loss: 0.6494 - val_accuracy: 0.8711
Epoch 17/20
81/81 [==============================] - 12s 153ms/step - loss: 0.0808 - accuracy: 0.9717 - val_loss: 0.4505 - val_accuracy: 0.8746
Epoch 18/20
81/81 [==============================] - 13s 155ms/step - loss: 0.1188 - accuracy: 0.9621 - val_loss: 0.3938 - val_accuracy: 0.9059
Epoch 19/20
81/81 [==============================] - 12s 153ms/step - loss: 0.0791 - accuracy: 0.9675 - val_loss: 0.9683 - val_accuracy: 0.8362
Epoch 20/20
81/81 [==============================] - 12s 154ms/step - loss: 0.0644 - accuracy: 0.9791 - val_loss: 0.6048 - val_accuracy: 0.8850
```

Overall, the accuracy & val_accuracy increases as the number of epochs increases. The val_accuracy fluctuates in each epoch, but there is an upward trend on both accuracies.

It is expected from a healthy model that validation accuracy ends up slightly lesser than training accuracy because training data is something with which the model is already familiar with and validation data is a collection of new data points which is new to the model.

If the difference between training accuracy-val_accuracy were very large, we could speak of overfitting, or if it is the opposite, we could speak of underfitting, however, in this case, the model performs and generalizes very well.



7.Discussion

Although the model performance is satisfactory, I think pointing out certain drawbacks of the model and data is necessary.

Some drawbacks of the brain tumor MRI dataset are:

-There are a lot of abnormalities in the sizes and location of the brain tumor(s).

-This makes it really difficult for complete understanding of the nature of the tumor.

-A professional Neurosurgeon was required for MRI analysis, which makes the process time consuming, costly, and maybe not very accurate.

Some drawbacks of the CNN are:

-The number of the training data is limited.

-CNN do not encode the position and the orientation of the object for example it cannot handle tilt or rotation.

-CNN does not recognize the coordinate frame.

-The training will take a long time considering the computer’s and/or cloud's capabilities.

8.Recommendations

Some methods are worth exploring to further improve the algorithm's accuracy:

-Early Stopping (to avoid overtraining),

-Transfer Learning(using pre-trained CNN and transfer its knowledge to a smaller dataset),

-Data augmentation (improving the ImageDataGenerator,zoom in, zoom out, shift etc. and creating more, diverse data),

-Ensembling different models.