PREDICTING BRAIN TUMORS FROM MRI SCANS

Final Report

Table of Contents

1 Abstract

This project employs two of the well-known architectures of Convolutional Neural Networks, namely VGG-16 and Inception in an experimental study of Brain Tumor Identification from the MRI Scans. The experiments show that the networks, which are fine-tuned using custom-sized inputs, obtain a very high recognition performance where a sequential model like VGG-16 achieves an accuracy of 86.56%. Moreover, a best accuracy of 96.25% was achieved using the Inception Architecture for CNN.

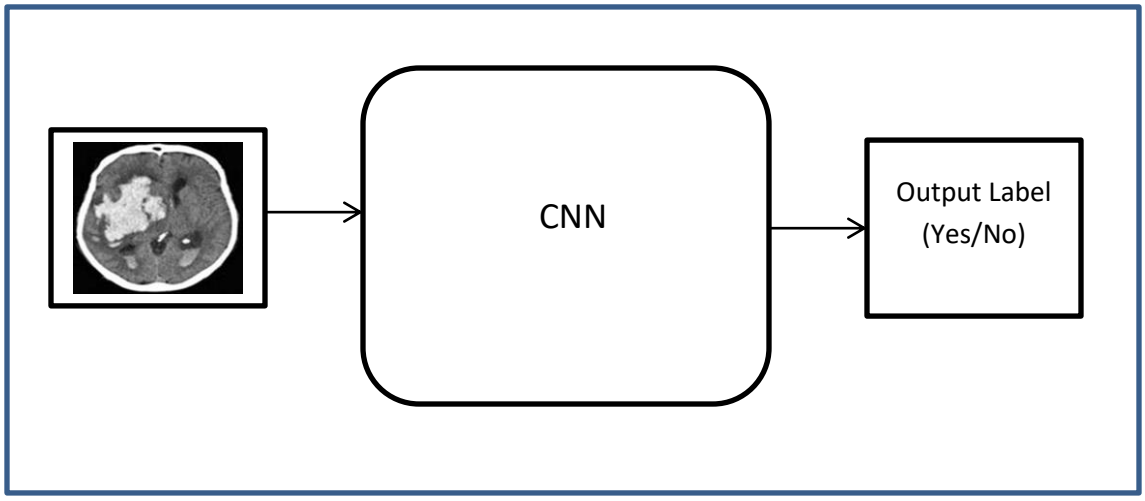

Figure 1. *Basic Flow of the Model*

2 Introduction

Cancer is the second leading cause of death globally, according to the World Health Organization (WHO). Early detection of cancer can prevent death, but this is not always possible. Unlike cancer, a tumor could be benign, pre-carcinoma, or malign. Benign tumors differ from malign in that benign generally do not spread to other organs and tissues and can be surgically removed. A tumor is an abnormal tissue in the brain which causes damage to the functioning of the cell. Therefore, brain tumor detection is an incredibly tricky task.

The most common method for differential diagnostics of tumor type is magnetic resonance imaging (MRI). 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.

Early brain–tumor detection mostly depends on the experience of the radiologist. Often times in developing countries the lack of skillful radiologist and lack of knowledge about tumors makes it really challenging and time-consuming to generate reports from MRIs. Thus, there is a need for automated brain tumor detection systems. The well-timed detection of the tumor can add to accurate treatment and can increase the survival rate of patients.

In recent years, deep learning algorithms and more specifically deep Convolutional Neural Networks (CNNs) have led to breakthroughs in many application domains including image classification, object detection and biometric recognition. These improvements are the result of several factors including the availability of tremendous amounts of labeled data, powerful hardware (i.e. GPUs) for accelerating computations, well-designed deep network architectures, effective optimization techniques and the technical improvement in training deep networks. Besides being scalable supervised learning techniques, deep CNNs perform the feature extraction and classification by training the entire system in an end-to-end manner and obviate the manual feature extraction.

The biggest problem with classifying and segmenting the MRI images with some neural networks lies in the number of images in the database. In addition, MRI images are acquired in different planes, so the option of using all the available planes could enlarge the database. As this could generally affect the classification output by over fitting, pre-processing is required before feeding the images into the neural network. However, one of the known advantages of convolutional neural networks (CNN) is that the pre-processing and the feature engineering do not have to be performed.

The aim of this project is to differentiate the brain with Tumors from that without. The dataset used here has around 3000 images. In this project, I have used two of the most famous CNN architectures available today, VGG-16 and Inception, to showcase how efficient and powerful their vanilla implementations can be. The network performance was tested through 10-fold cross-validation method. The results are presented as resulting classified images.

3 Work Done

3.1 Image Database

The image database used in this project contains 3525 contrast enhanced MRI Images obtained from different sources on Kaggle.com. There are two types of images: With Tumor (2929) and without Tumor (596). The examples of images present in dataset are shown below in Figure 2.

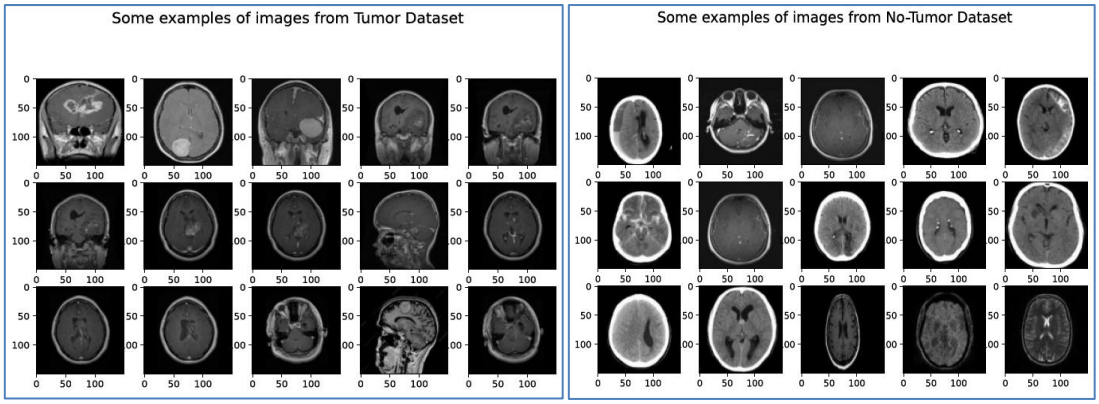

Figure 2. *MRI Scans with Tumor (left).MRI Scans without Tumor (right).*

3.2 Data Cleaning

MRI Scan dataset contained images in two different formats: .JPG & .PNG. Hence, PNG files were converted to JPG files so that all the images share a single common format. The images were renamed in the following format: Y_<sequence #> and N_<sequence #>. Here, Y- with tumor and N- without tumor.

3.3 Exploratory Data Analysis (EDA)

After Data Cleaning, I performed EDA on this clean dataset to find how the data looks like and if there is any kind of pattern in the Images. The dataset had images all of which had 3 layers- RGB and a large number of images had a size of 700x700 or smaller (Figure 3).

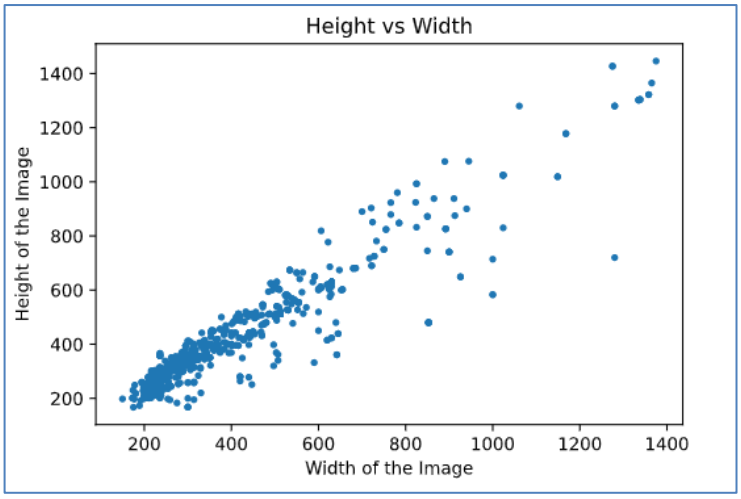

Figure 3. *Height Vs. Width distribution of Images.*

3.4 Data Preprocessing

As per EDA, MRI Scans are of varying sizes with the smallest image had dimensions of 150x167 hence; the whole Image dataset was reshaped to a common size of 150x150 pixels. After resizing, the whole dataset was broken into Train-Test set in 80-20 ratio. Figure 4, shows the final size of train and test dataset.

```
Train Yes: 2344
Train No: 477
Test Yes: 585
Test No: 119
```

Figure 4. *Train and Test Set sizes with Yes (tumor) and No (no tumor).*

3.5 Network Architectures

Tumor Classification was performed using Convolutional Neural Networks. The CNNs used for this particular project are the well-known state-of-the-art architectures of VGG-16 and Inception. The description of both the networks is provided below:

3.5.1 VGG-16

The VGG convolutional neural network architecture, named for the Visual Geometry Group at Oxford, was an important milestone in the use of deep learning methods for computer vision. The architecture was described in the 2014 paper titled "Very Deep Convolutional Networks for Large-Scale Image Recognition" by Karen Simonyan and Andrew Zisserman and achieved top results in the LSVRC-2014 computer vision competition.

The key innovation in this architecture was the definition and repetition of what we will refer to as VGG-blocks. These are groups of convolutional layers that use small filters (e.g. 3×3 pixels) followed by a max pooling layer.

A convolutional neural network with VGG-blocks is a sensible starting point when developing a new model from scratch as it is easy to understand, easy to implement, and very effective at extracting features from images.

Table 1 below shows the VGG-16 architecture implemented for this project.

Layer No.	Layer Name	Properties
1	Input Layer	150X150X3 Image
2	Convolutional	Filters = 64, kernel size = 3x3, padding = same, activation = Rectified Linear Unit
3	Max Pooling	Pool size = 2x2, stride = 2x2
4	Convolutional	Filters = 128, kernel size = 3x3, padding = same, activation = Rectified Linear Unit
5	Convolutional	Filters = 128, kernel size = 3x3, padding = same, activation = Rectified Linear Unit
6	Max Pooling	Pool size = 2x2, stride = 2x2
7	Convolutional	Filters = 256, kernel size = 3x3, padding = same, activation = Rectified Linear Unit
8	Convolutional	Filters = 256, kernel size = 3x3, padding = same, activation = Rectified Linear Unit
9	Convolutional	Filters = 256, kernel size = 3x3, padding = same, activation = Rectified Linear Unit
10	Max Pooling	Pool size = 2x2, stride = 2x2
11	Convolutional	Filters = 512, kernel size = 3x3, padding = same, activation = Rectified Linear Unit
12	Convolutional	Filters = 512, kernel size = 3x3, padding = same, activation = Rectified Linear Unit
13	Convolutional	Filters = 512, kernel size = 3x3, padding = same, activation = Rectified Linear Unit
14	Max Pooling	Pool size = 2x2, stride = 2x2
15	Convolutional	Filters = 512, kernel size = 3x3, padding = same, activation = Rectified Linear Unit
16	Convolutional	Filters = 512, kernel size = 3x3, padding = same, activation = Rectified Linear Unit
17	Convolutional	Filters = 512, kernel size = 3x3, padding = same, activation = Rectified Linear Unit
18	Max Pooling	Pool size = 2x2, stride = 2x2
19	Flattening	Flatten the resulting output
20	Fully Connected	4096 hidden neurons in Fully Connected (FC) layer, activation = Rectified Linear Unit
21	Fully Connected	4096 hidden neurons in Fully Connected (FC) layer, activation = Rectified Linear Unit
22	Fully Connected	2 hidden neurons in Fully Connected (FC) layer, activation = Softmax
23	Classification Output	2 output classes, "1" for Tumor, "0" for No Tumor.

Table 1. *VGG-16 Model Architecture*

3.5.2 Inception

The inception module was described and used in the GoogLeNet model in the 2015 paper by Christian Szegedy, et al. titled “Going Deeper with Convolutions.”

The key innovation on the inception model is called the inception module. This is a block of parallel convolutional layers with different sized filters (e.g. 1×1, 3×3, 5×5) and a 3×3 max pooling layer, the results of which are then concatenated. This is a very simple and powerful architectural unit that allows the model to learn not only parallel filters of the same size, but parallel filters of differing sizes, allowing learning at multiple scales.

This is a very simple and powerful architectural unit that allows the model to learn not only parallel filters of the same size, but parallel filters of differing sizes, allowing learning at multiple scales.

Figure 5 shows the schematic Diagram of the Implemented Inception Model and Table 2 shows the Model Layers with properties.

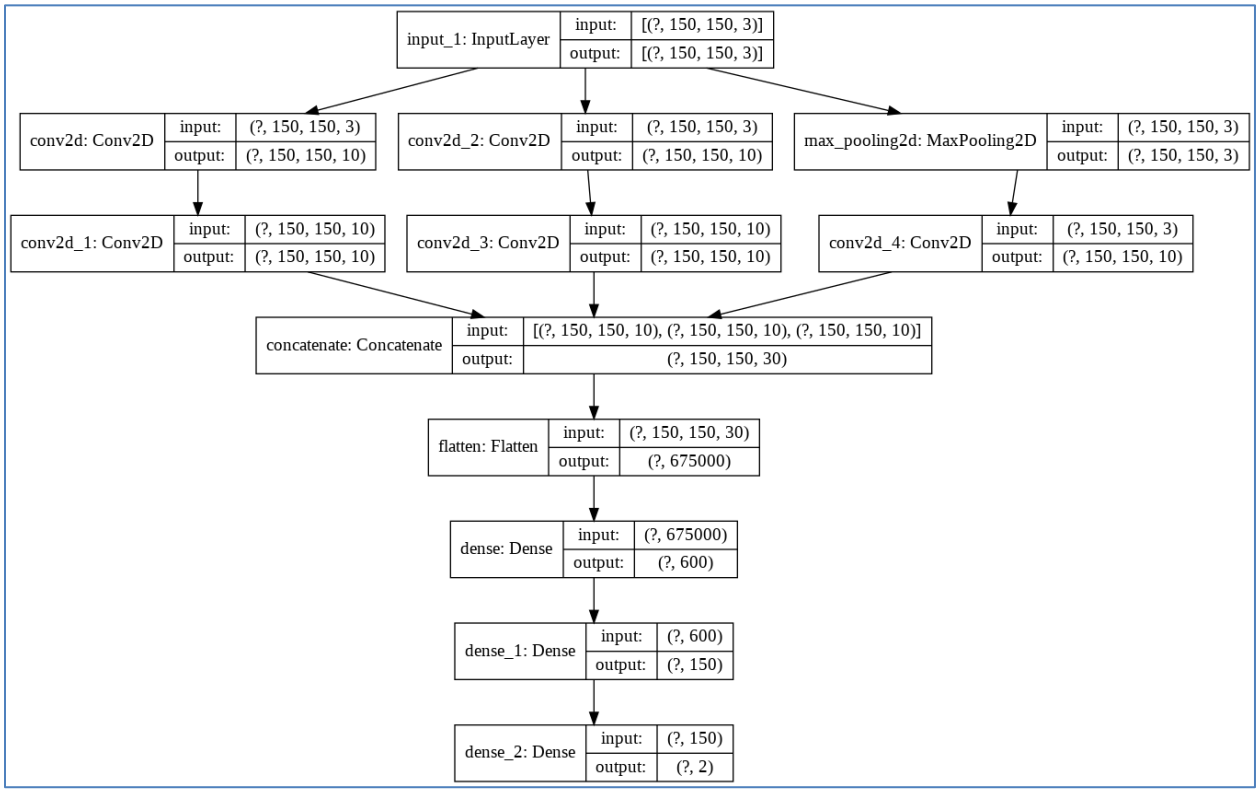

Figure 5. *A diagram of the implemented Inception Architecture*

Layer No.	Layer Name	Properties
1	Input Layer	150X150X3 Image
2	Convolutional	Filters = 10, kernel size = 1x1, padding = same, activation = Rectified Linear Unit, connected to Input Layer
3	Convolutional	Filters = 10, kernel size = 3x3, padding = same, activation = Rectified Linear Unit, connected to Layer #2
4	Convolutional	Filters = 10, kernel size = 1x1, padding = same, activation = Rectified Linear Unit, connected to Input Layer
5	Convolutional	Filters = 10, kernel size = 5x5, padding = same, activation = Rectified Linear Unit, connected to Layer #4
6	Max Pooling	Pool size = 3x3, stride = 1x1, padding = same, connected to Input Layer
7	Convolutional	Filters = 10, kernel size = 1x1, padding = same, activation = Rectified Linear Unit, connected to Layer #6
8	Concatenate	Concatenate the Layers- 3, 5, 7
9	Flattening	Flatten the resulting output
10	Fully Connected	600 hidden neurons in Fully Connected (FC) layer, activation = Rectified Linear Unit
11	Fully Connected	150 hidden neurons in Fully Connected (FC) layer, activation = Rectified Linear Unit
12	Fully Connected	2 hidden neurons in Fully Connected (FC) layer, activation = Softmax
13	Classification Output	2 output classes, "1" for Tumor, "0" for No Tumor.

Table 2. *Inception Model Architecture*

3.6 Training Network

The dataset was divided in proportions of 8:2, Train to Validation data to train the model and validate it against the Validation Data. A separate set of 40 Images were kept for Prediction from the trained model.

The Networks were trained using "Adam Optimizer" with Accuracy as metric. The Loss function used for VGG-16 was "Binary Cross Entropy" while for Inception was "Categorical Cross Entropy". The early-stop condition that affects when the process of network training will stop corresponds to twenty epochs (of a total of 50 epochs). More specifically, it was tuned to finish the training process after twenty epochs, if the validation set accuracy doesn't improve.

4 Results & Discussion

The model trained using the VGG-16 Architecture had an accuracy of 86.56% while the one trained using the Inception Architecture had an Accuracy of 96.25%. Let's discuss the results of both.

4.1 VGG-16

VGG-16 was trained on 80% of the available dataset. A total of 50 epochs were used to train this model. The Final Accuracy achieved by the Validation Set was of 86.56% (Figure 6). The average time taken to train this model was ~1100s/epoch. This model was then used on a completely new Dataset of 40 images (20 with Tumor and 20 without Tumor) to check how many of the predictions made by the model were correct. In this case the model predicted all 40 images correctly (Figure 7). Model Accuracy when plotted with the Loss function shows that the Accuracy and Loss do not change a lot after each epoch (Figure 8).

```
Epoch 48/50
50/50 [==============================] - ETA: 0s - loss: 2.6326 - accuracy: 0.8284
Epoch 00048: val_accuracy did not improve from 0.86563
50/50 [==============================] - 1073s 21s/step - loss: 2.6326 - accuracy: 0.8284 - val_loss: 2.3964 - val_accuracy: 0.8438
Epoch 49/50
50/50 [==============================] - ETA: 0s - loss: 2.5114 - accuracy: 0.8363
Epoch 00049: val_accuracy did not improve from 0.86563
50/50 [==============================] - 1104s 22s/step - loss: 2.5114 - accuracy: 0.8363 - val_loss: 2.3485 - val_accuracy: 0.8469
Epoch 50/50
50/50 [==============================] - ETA: 0s - loss: 2.6840 - accuracy: 0.8250
Epoch 00050: val_accuracy did not improve from 0.86563
50/50 [==============================] - 1092s 22s/step - loss: 2.6840 - accuracy: 0.8250 - val_loss: 2.5881 - val_accuracy: 0.8313
```

Figure 6.*VGG-16 Training results*

```
Correct Classification: 40
Incorrect Classification: 0
```

Figure 7.*VGG-16 Prediction Results*

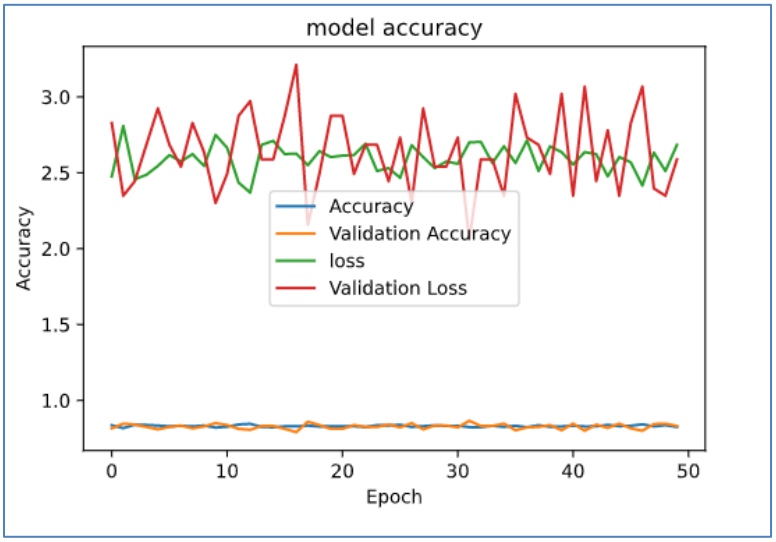

Figure 8.*VGG-16 Change in Accuracy and Loss with each Epochs*

4.2 Inception

Inception was trained on 80% of the available dataset. A total of 50 epochs were used to train this model. The Final Accuracy achieved by the Validation Set was of 96.25% (Figure 9). The average time taken to train this model was ~260s/epoch. This model was then used on a completely new Dataset of 40 images (20 with Tumor and 20 without Tumor) to check how many of the predictions made by the model were correct. In this case the model predicted all 38 images correctly (Figure 10). Model Accuracy when plotted with the Loss function shows that the Loss decreases with each epoch (Figure 11).

```
Epoch 00039: val_accuracy did not improve from 0.96250
50/50 [==============================] - 259s 5s/step - loss: 0.0520 - accuracy: 0.9981 - val_loss: 13.8782 - val_accuracy: 0.9469
Epoch 40/50
50/50 [==============================] - ETA: 0s - loss: 0.4631 - accuracy: 0.9955
Epoch 00040: val_accuracy did not improve from 0.96250
50/50 [==============================] - 258s 5s/step - loss: 0.4631 - accuracy: 0.9955 - val_loss: 21.1055 - val_accuracy: 0.9000
Epoch 41/50
50/50 [==============================] - ETA: 0s - loss: 0.5353 - accuracy: 0.9917
Epoch 00041: val_accuracy did not improve from 0.96250
50/50 [==============================] - 257s 5s/step - loss: 0.5353 - accuracy: 0.9917 - val_loss: 21.0188 - val_accuracy: 0.9344
Epoch 00041: early stopping
```

Figure 9.*Inception Training results*

```
Correct Classification: 38
Incorrect Classification: 2
```

Figure 10.*Inception Prediction Results*

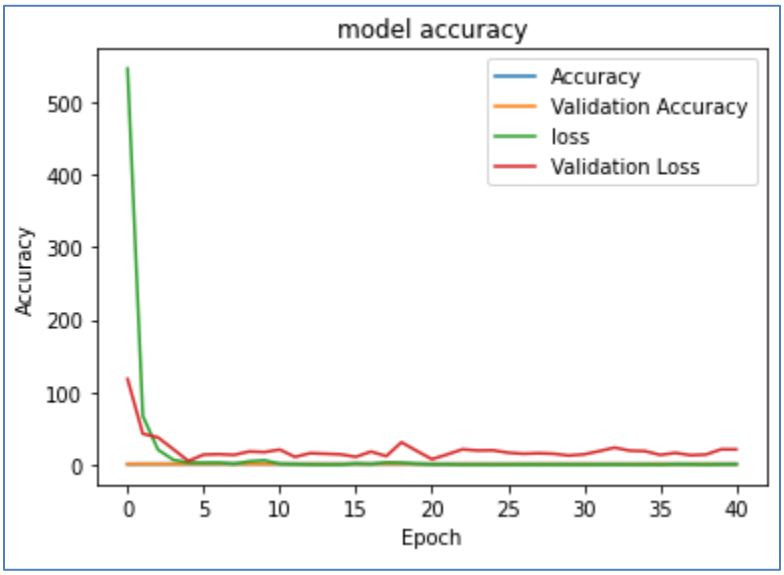

Figure 11.*Inception Change in Accuracy and Loss with each Epochs*

5 Conclusion

This project used two of the most famous CNN Architectures (VGG-16 and Inception) for Image Classification. The classification was performed using a contrast-enhanced MRI Image database which contains both types of images- 1) Having Tumor, and 2) Having No Tumor. As input, we used whole images in JPG format and resized them to 150x150 pixels. Both the architectures showed a high Accuracy in predictions but the time taken in training VGG-16 was 4-times the time taken to Train Inception. Also, Inception showed more promise over VGG-16 in terms of Accuracy as well as Inception had a Validation Accuracy of 96% while VGG-16 had around 86%. These networks were trained on an average Personal Computer with 8GB of RAM and i3 Processors hence, these architectures require less number of resources as well to train.

These results show that Inception has a good generalization capability and good execution speed, so it could be used as an effective decision-support tool for radiologists in medical diagnostics.

Regarding further work, other well-known CNN approaches like ResNet can be implemented on the Dataset to see how there results fare with the results obtained using VGG-16 and Inception. An Ensemble of all the above architectures can also be implemented with a polling system to decide the outcome as we have already seen that an Ensemble of different Methods can lead to better prediction accuracies.

In future, this Model can also be generalized to predict the type of Brain Tumors i.e, Glioma, Meningioma, and Pituitary Tumors. One of the main improvements will be adjusting the architecture so that it could be used during brain surgery, classifying and accurately locating the tumor. Detecting the tumors in the operating room should be performed in real-time and real-world conditions; thus, in that case, the improvement would also involve adapting the network to a 3D system.

6 References

1. Classification of Brain Tumors from MRI Images Using a Convolutional Neural Network by Milica M. Badža and Marko C. Barjaktarovi´c.
2. World Health Organization.
3. Step by step VGG16 implementation in Keras for beginners by Rohit Thakur.
4. Dataset Sources:
    a. https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection?
    b. https://www.kaggle.com/sartajbhuvaji/brain-tumor-classification-mri?