Brain Tumor MRI Classification

Tanya Jain, Kavya Anantha Rao, Pranjal Jain
University at Buffalo, SUNY

***Abstract*— Early detection of the brain tumor has become a necessity in research to facilitate appropriate treatment to the patients. Magnetic Resonance Imaging has been used for identifying the tumor and its modeling process. Brain tumors not only possess high variance in appearance but also possess similarity between the normal and tumor tissues, thereby making the classification process challenging [1]. The significance of classifying the tumors accurately has become a major aim of radiologists to reduce the false negatives, which when undetected might deteriorate the condition of the patients. The dataset used for study under consideration consisted of images with various intensities, size and shape. The main objective of the proposed work is to build and compare the convolutional neural networks to accurately classify the images as to having tumor or not [2]. The best accuracy among the three models was 98.08% obtained using CNN implemented in Keras.**

Keywords—tumor, classification, convolutional, neural networks, confusion matrix, VGG16.

I. INTRODUCTION

Medical imaging methods are used to generate images of the structures and organs inside the human body. Computer-Aided diagnostic technology has become invaluable in the medical field as it assists in diagnosis and prognosis of treatment [3]. Brain tumor is due the occurrence or presence of abnormal cells inside the brain. They are of two types - benign and malignant. A benign tumor is non-cancerous tumor and is localized whereas malignant tumors are cancerous and grows very fast and spreads to surrounding parts and thereby affect the central nervous system.

Early detection of the brain tumor has become a necessity in research to facilitate appropriate treatment to the patients. Magnetic Resonance Imaging (MRI) used to detect a brain tumor and the Magnetic Resonance (MR) images of the brain are primarily used for identifying the tumor and its modeling process. Brain tumor segmentation from MRI is the most significant and challenging task medical image processing as the tumors do not have a well-defined shape. MRI images are used to detect the presence of tumor, their localization and severity.

The significance of classifying the tumors accurately has become a major aim of radiologists to reduce the false negatives, which when undetected might deteriorate the condition of the patients. Current trend has been to incorporate machine learning techniques to classify the images, and strengthen the diagnosis by the healthcare professional. The dependency on the radiologists to analyze the MR scan necessitates automatic classification methods to improve the detection accuracy.

The objective is to build a model that classifies the brain MR images with high accuracy to detect the presence of tumors. The images can be classified using classical machine learning methods such as SVM classifier and Adaboosting as well as deep learning techniques such as convolutional neural networks. In this proposed work, we have discussed the implementation of Convolutional Neural Network (CNN) for classifying the images as to having tumors or not. Deep CNN architectures are designed using small kernels. The CNN framework with discrimination mechanism is effective to achieve the goals in the problem statement. The system being built aims to perform automatic classification and will be an invaluable tool in the real-world scenario for health sciences.

II. PROPOSED METHODOLOGY

In the proposed method, each of the stages are implemented to collectively constitute a brain MRI classifier and discussed in detail as shown in figure 1. The coding is performed in Python programming language with the significant modules being - Pytorch, Keras, VGG16, OpenCV, Torchvision, Tensorflow, etc. The outline of the system is as follows: dataset description, preprocessing the images, classification of images as to having brain tumors or not using CNN, performance measurement and evaluation.

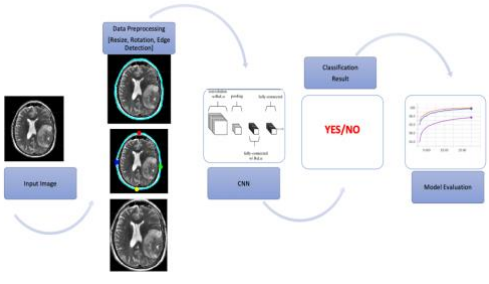


Figure 1: Proposed methodology for brain tumor classification system

A. *Dataset Description:*

The dataset under consideration is obtained from the following link: https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection

The dataset has 251 images and it is split into train and validation sets in the ratio of 80:20. It has images with different intensities, sizes, orientations and shapes. The tumors do not have a well-defined shape and thus makes it a challenging task in medical image processing. Thus, image preprocessing is a significant task in building a good classifier.

B. *Data Preprocessing:*

Appropriate segmentation of the tumor will generate accurate results and accurate detection allows better feature extraction and thus will provide good classification. The images may be distorted due to defects during the acquisition process. Data preprocessing is performed to get the robust learning outcomes and thereby enhance the precision and interpretability of an image. It is used to improve the available image data by enhancing important features and suppressing the unnecessary distortions.

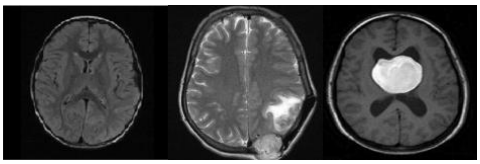


Figure 2: Images obtained from the MRI system

Figure 2 shows the MRI brain samples with and without tumor. There is a need to remove spurious intensity variations caused by inhomogeneity of the magnetic fields and coils. Obtaining preprocessed and well-annotated data is a crucial step to achieve good performance.

Image Resize:

In order to have the base size of the images' constant, this preprocessing step is considered so that all the images are scaled to have a standard width and height.

Image Crop:

Images are cropped through the contours to obtain clarity of tumors within the boundaries of the brain. Non-trivial blocks of the image are 'Gaussian Blurred' and dilations with several erosions are performed to remove small regions of noise.

Image Rotation:

The images are rotated with in either the clockwise or counterclockwise direction with the range of rotation is specified. Rotation is necessary as the slope of the image may be different and may be dependent on the data.

Edge Detection:

It is a technique of determining the edges or boundaries of objects within an image. It works by detecting drastic discontinuities or changes in brightness in the image. It helps to focus on the significant parts of interest while ignoring the others. It helps to determine the shape or structure of the object of interest.

C. *Methods*

Brain tumor classification is achieved using Convolutional Neural Network (CNN) classification. Deep CNN architectures are designed using small kernels. CNN architecture consists of blocks of different 2-D CNN networks. These blocks contain sequential layers constituting the structure of CNN. This framework with discrimination mechanism will be effective to achieve the goals of the proposed system.

Mathematically, convolution sums up the element-wise product of two matrices. For images, it basically convolves the image (matrix 1) using a filter (matrix 2). Figure 3 shows an example of a convolutional neural network and the description of layers that are used in a CNN follow.

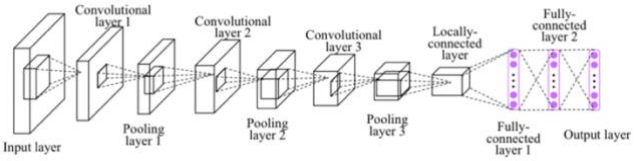


Figure 3: Example of a convolutional neural network [4]

Few terminologies

Stride:

The number of steps to be moved in convolution (default=1) that is the number of pixels' shifts over the input matrix.

Padding:

It is a method of adding zeros across all the edges of the image so the details from the edges of the actual image are not compromised and the size of the output is maintained as that of input.

CNN uses the spatial information between the input image pixels using two basic processes, convolution and pooling (max, min or average). These processes

will run in the background with each consecutive layer of the network.

Convolution layer:

It is the main one in the algorithm. It extracts key features from the input image (brain image from the MRI scans) where we would use filters or kernels of variable sizes. It outputs a scalar value which is an artificial neuron.

Pooling layers:

It performs spatial down-sampling and preserves the most relevant information thereby reducing the size of representation. It speeds up the computation which is necessary for a system to be time efficient and more robust.

Activation function:

(RELU, sigmoid or tanh) will be used to eliminate the non-linearity and produce feature maps of the images. Softmax function will be used to calculate the probability value for each classification label of our predictions for the flattened output of the pooling layer.

The layers components seen in the network are defined at first followed by wiring the network in the required order.

Fully Connected Layer:

This layer changes the previous layer's 2D structure features into a predefined one-dimensional feature vector. Essentially, the main task of the Fully Connected (FC) layer is to mine the incoming features to extract information about the input image. The process usually involves flattening the feature maps coming from convolutional layers, to achieve a one-dimension feature vector representation, and, then, inputting it to the FC layer. The output of this layer could either be the predict class labels or an intermediate layer (consecutive FC layers can be stacked together).

Loss function:

CNNs are well-known for their ability to extract discriminative features using learned weights in each layer. The learning process is reinforced by employing appropriate loss functions. Loss functions are designed to encourage intra-class similarity and inter-class separability.

Optimization:

Deep CNNs are learnt by searching appropriate values for model parameters to optimize the loss function. Gradient descent methods, in which the parameter update is performed using a back-propagation algorithm are widely used for minimization. This parameter update is performed by computing the loss function for a single, small subset, or the whole training set.

Model Building:

The proposed work accomplishes brain tumor classification using three models implemented in three different frameworks. The first model was implemented by constructing the CNN layer by layer using Pytorch. In this model, the input images were given as train and test sets. The dataset 'MRI_data' is split into train and validation. Within each of the data split (i.e. train and validation) the images are segregated into 'yes' and 'no'. These subsets of data are used for both the CNN architectures in Pytorch. However, for the CNN implementation using Keras, the dataset is split into 'yes' and 'no' subsets.

The CNN network has four layers divided into input and convolution sections. The input layer processes the input image in order to produce the convolved designed image patches. There are multiple layers in convolution section which helps to sequentially identify the features using convolution operations.

The convolution section processes this designed image from previous layer onto output feature maps. In order to reduce the computation produced by the large image patches we use down sampling process called as pooling. Further, the fully connected layer groups all the feature maps. The classification section estimates the predictions to classify every image. Figure 4 shows the CNN model of the developed system.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 16)	3904
max_pooling2d (MaxPooling2D)	(None, 16, 16, 16)	0
batch_normalization (BatchNo	(None, 16, 16, 16)	64
dropout (Dropout)	(None, 16, 16, 16)	0
conv2d_1 (Conv2D)	(None, 16, 16, 16)	20752
max_pooling2d_1 (MaxPooling2	(None, 8, 8, 16)	0
batch_normalization_1 (Batch	(None, 8, 8, 16)	64
dropout_1 (Dropout)	(None, 8, 8, 16)	0
conv2d_2 (Conv2D)	(None, 8, 8, 36)	46692
max_pooling2d_2 (MaxPooling2	(None, 4, 4, 36)	0
batch_normalization_2 (Batch	(None, 4, 4, 36)	144
dropout_2 (Dropout)	(None, 4, 4, 36)	0
flatten (Flatten)	(None, 576)	0
dense (Dense)	(None, 512)	295424
dropout_3 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 1)	513

Total params: 367,557
Trainable params: 367,421
Non-trainable params: 136

Figure 4: CNN model of the developed system

VGG16 is a convolutional neural network model that is very deep CNN for large scale image recognition. VGG network consists of VGG Blocks. One VGG block consists of sequence of convolutional layers followed by max pooling layer for spatial down sampling. The network is partitioned into two parts first contains convolutional and pooling layers and second consists of fully connected layers. The VGG16 architecture consists of twelve convolutional layers, some of which are followed by maximum pooling layers and then four fully-connected layers and finally a 1000-way SoftMax classifier. We have loaded a pretrained version of the network in which transfer learning was implemented to take advantage of the pretrained weights to predict new classes. By using a pretrained network to do transfer learning, we are simply adding a few dense layers at the end of the pretrained network and learning what combination of these already learnt features help in recognizing the objects in our new dataset. The network we used that is trained on more than a million images from the ImageNet database. We used vgg-16 network due to its reusable convolutional blocks. The use of blocks leads to very compact representations of the network. Sometimes it gives a very good accuracy for high dimensional images as it is 16 layers deep.

Keras is a high-level neural network API written in Python. It is more restrictive compared to Tensorflow. Thus, it possesses a high-level framework than TensorFlow. Figure 5 shows the CNN structure developed using Keras.

III. RESULTS

The performance of the CNN depends on the hyperparameter used while building the architecture, that is the number of layers, number of feature maps, type of activation function, batch normalization etc. All these factors govern the training of our model. Accuracy and Validation Loss are calculated for each of the three models.

The evaluation measurements compare the output of segmentation algorithms with ground truth on a pixel-wise basis. Three models were built and their accuracies were compared. Models were continuously trained for accuracy improvement. They were enhanced by addition or changing of some convolutional layers, pooling layers and performing batch normalization. They were tested against the data for several epochs and hence validated for accuracy measurement.

The accuracy, batch size and number of epochs for each of the models is shown in table 1.

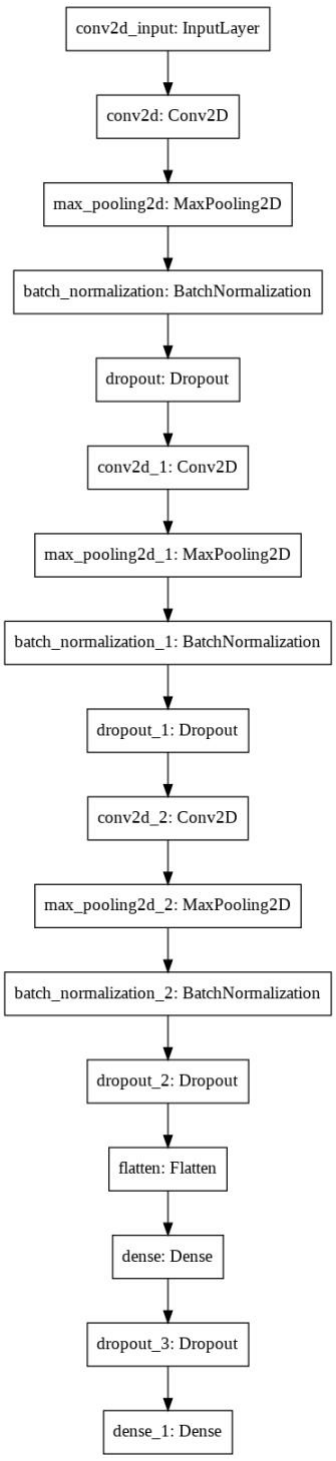


Figure 5: CNN structure using Keras

Model	Accuracy	Batch size	Epochs
CNN	82	Train=4, Test=1	150
VGG-16	58.82	Train=16, Test=4	150
Keras	98.08	Train=1, Test=1	150

Table 1: Details of accuracy, batch size and number of epochs for three models

A subset of predicted images from CNN using Keras is shown in figure 6. It is observed that the classes have been accurately predicted matching with their ground truth.

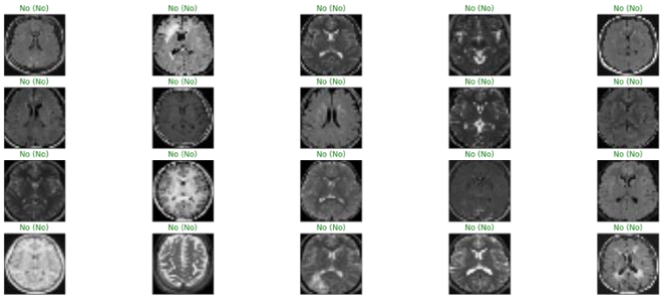
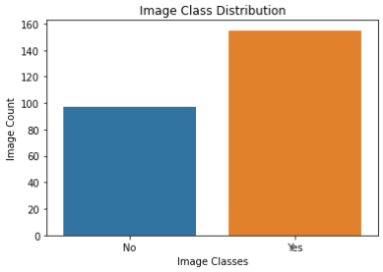


Figure 6: Predictions of the test images by the CNN model using Keras

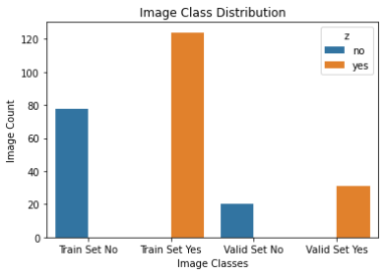
MR image analysis can train deep CNNs to obtain accurate segmentation algorithms. Although working with each model has its computational advantages and drawbacks, obtaining good generalization requires an architecture with optimized layers, considering the class imbalance and selecting the best hyper-parameters and advanced training procedures.

IV. DISCUSSION

The image class distribution for CNN in Keras and Pytorch is shown in figure 7. As described above, the dataset is split uniquely for CNN implementations in Pytorch and Keras to meet the data loader requirements in Pytorch.



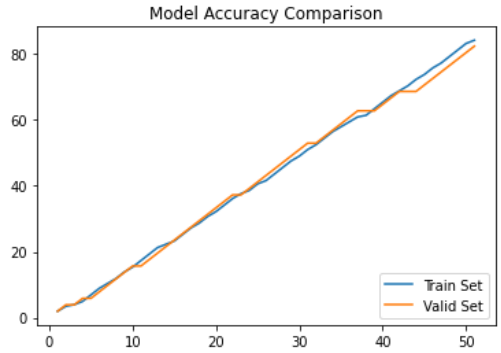
(a)



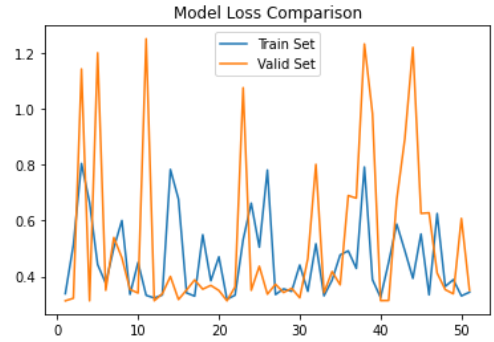
(b)

Figure 7: Image class distribution for (a)CNN in Keras, (b) CNN and VGG 16 in Pytorch.

Figure 8 shows the plot of model accuracy and loss for CNN model for 150 epochs. 'Epoch' is defined as a full sweep through the entire data collection. The train and validation accuracy are in good agreement. The validation accuracy is found to increase with increase in number of epochs figure 8(a). As the epochs increase, it is seen from figure 8(b) that loss has lot of fluctuations and drops up to a minimum quite a few times. The accuracy obtained from this model was 82%.



(a)



(b)

Figure 8: CNN model (a) accuracy and (b) loss

Confusion matrix:

It is also known as error matrix which is a table layout to describe the performance of a classification model on a set of test data for which the true values are known. It has four different value combinations of predicted and actual values. It allows the visualization of the performance of an algorithm. When we get the data after preprocessing, we feed it into a model and get the output in probabilities. The effectiveness of our model is determined by confusion matrix. The total number of correct and incorrect classification predictions are captured along with their respective counts for each class. It shows how the classification model is confused while making predictions. It provides details of the misclassifications made as well as the types of misclassifications. It is used for measuring accuracy, recall and precision.

From the figure 9, we can deduce information as:
TP - Image predicted as having tumor and it is true
TN - Image predicted negative for tumor and is true.
FP - Image predicted as having tumor but is false
FN - Image predicted negative for tumor and is false.

Accuracy is determined from confusion matrix as:

$$\textbf{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

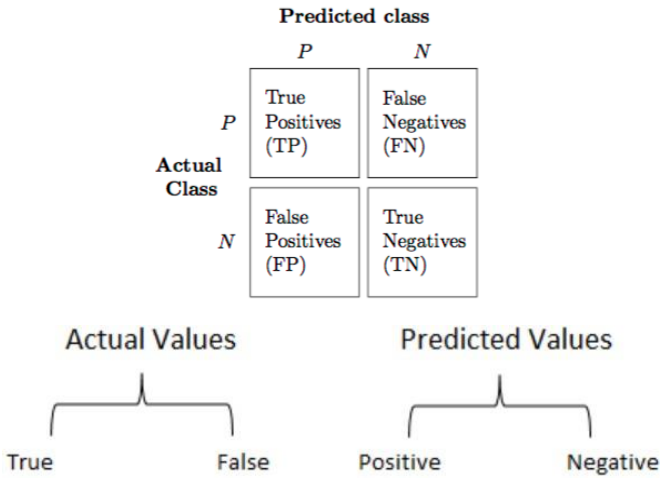


Figure 9: Example of confusion matrix.

The confusion matrix for CNN is shown in figure 10. Using the above formula for accuracy, we can confirm the accuracy to be 82.35% (as quoted in table)

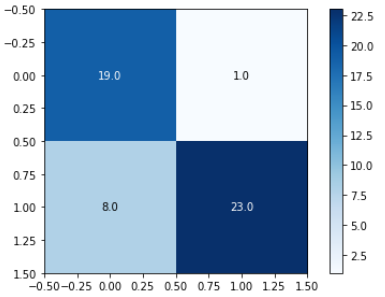


Figure 10: Confusion Matrix and loss for CNN

Figure 11 shows the plot of model accuracy and loss for VGG-16 model for 150 epochs. The train and validation accuracy are in good agreement. The validation accuracy is found to increase with increase in number of epochs figure 11(a). The test accuracy was found to drop at epoch=11 after which the training was stopped. As the epochs increase, it is seen from figure 11(b) that loss remains almost a constant but spikes up at a value of epoch =11, supporting the reduction in accuracy as described above. The accuracy obtained from this model was 58%.

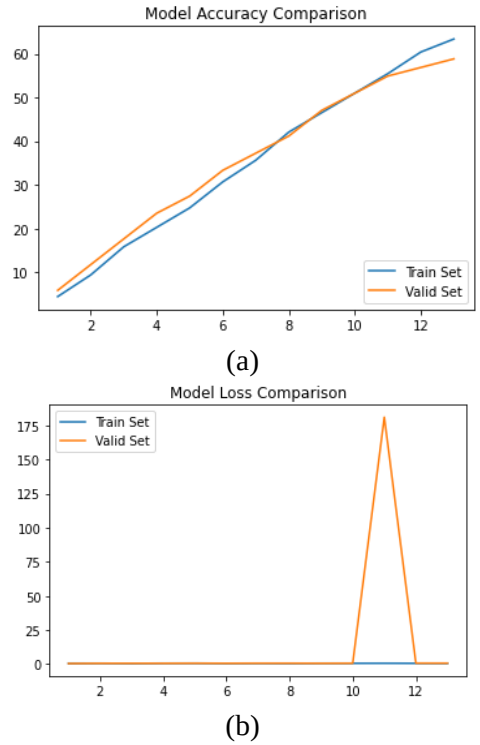


Figure 11: VGG-16 model (a) accuracy and (b) loss

The confusion matrix for VGG-16 is shown in figure 12. The accuracy is found to be around 58.82 and we can confirm this as quoted in table.

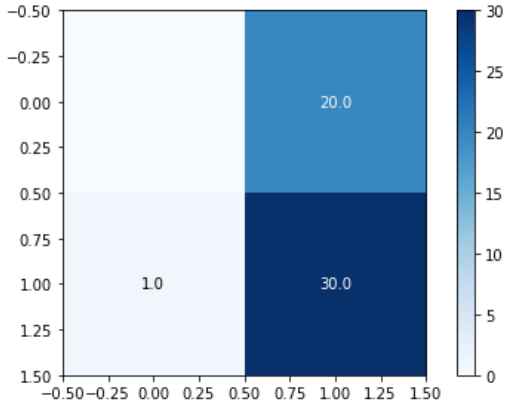
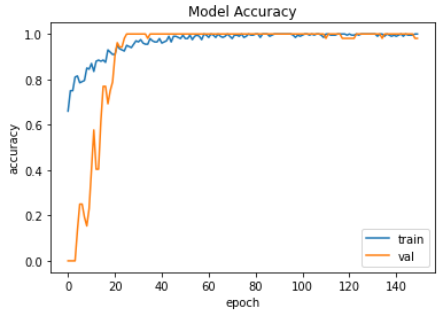
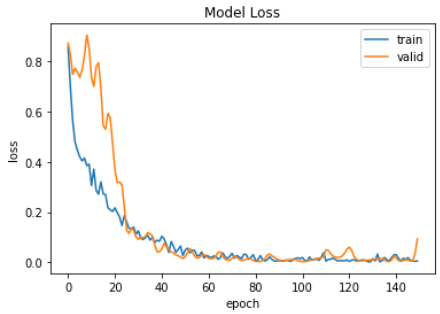


Figure 12: Confusion Matrix and loss for VGG-16

Figure 13 shows the plot of model accuracy and loss for CNN model using Keras for 150 epochs. The train and validation accuracy are in good agreement after about 40 epochs. The validation accuracy is found to gradually increase with increase in number of epochs figure 13(a). After about 60 epochs, there is not much change in accuracy. The train accuracy was more than 60% during the entire training process. As the epochs increase, it is seen from figure 13(b) that loss goes on decreasing, providing evidence for the increase in accuracy as described above. The accuracy obtained from this model was 98.08%.



(a)



(b)

Figure 13: CNN model (a) accuracy and (b) loss (Keras)

The confusion matrix for CNN is shown in figure 14. Using the above formula for accuracy, we can confirm the accuracy to be 98.07% (as quoted in table).

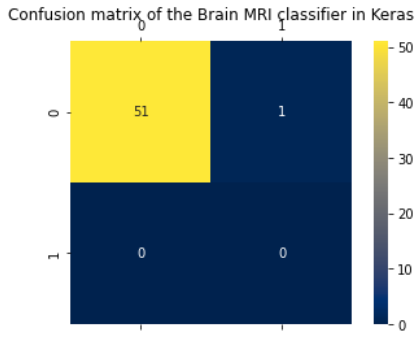


Figure 14: Confusion Matrix and loss for model using Keras

The system being developed will classify MR images as having tumors or not. The results from the system are expected to be consistent and it will provide radiologist a system that will be self-explanatory and simple to operate.

V. CONCLUSION

In this paper, classification of brain tumor is implemented using CNN architecture which is revised by enhancing accuracies by subsequent models building. The final CNN achieves an accuracy of about 98.08% during training and validation phase. CNN networks require less hardware specifications and takes convenient time of processing for large size images. The accuracy was increased by performing batch normalization at the end of each block or after pooling layer in each block. The training speed was increased
by using pretrained models such as VGG-16 and performing transfer learning. Speed was increased by reducing the features in the fully connected layer. Overfitting was controlled by reduction of parameters used in the model. The good results achieved using the CNN architecture implemented in Keras could be employed with CNN's for tumor detection and segmentation in future.

Hence, the CNN classifier will be very significant in the medical field and in saving precious lives.

REFERENCES

[1] Hossain, Tonmoy, Fairuz Shadmani Shishir, Mohsena Ashraf, MD Abdullah Al Nasim, and Faisal Muhammad Shah. "Brain Tumor Detection Using Convolutional Neural Network." In 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), pp. 1-6. IEEE, 2019.

[2] Nandpuru, Hari Babu, S. S. Salankar, and V. R. Bora. "MRI brain cancer classification using support vector machine." In 2014 IEEE Students' Conference on Electrical, Electronics and Computer Science, pp. 1-6. IEEE, 2014.

[3] Ouchtati, Salim, Jean Sequeira, Belmeguenai Aissa, Rafik Djemili, and Mohamed Lashab. "Brain tumors classification from MR images using a neural network and the central moments." In 2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET), pp. 455-460. IEEE, 2018.

[4] https://towardsdatascience.com/convolutional-neural-network-for-image-classification-with-implementation-on-python-using-pytorch-7b88342c9ca9