Name : Akshay Kumawat

Roll no:17MI33002

Brain Tumor detection report

**Abstract:**

The brain tumors are the most common and aggressive disease, leading to a very short life expectancy in their highest grade. Thus, treatment planning is a key stage to improve the quality of life of patients. Generally, various image techniques such as Computed Tomography (CT), Magnetic Resonance Imaging (MRI) and ultrasound image are used to evaluate the tumor in a brain, lung, liver, breast, prostate…etc. Especially in this work MRI images are used to diagnose tumors in the brain. However the huge amount of data generated by MRI scan thwarts manual classification of tumor vs non-tumor in a particular time. But it has some limitations (i.e) accurate quantitative measurements are provided for a limited number of images. Hence trusted and automatic classification schemes are essential to prevent the death rate of humans. The automatic brain tumor classification is a very challenging task in large spatial and structural variability of the surrounding region of brain tumor. In this work, automatic brain tumor detection is proposed by using Convolutional Neural Networks (CNN) classification. The deeper architecture design is performed by using small kernels. The weight of the neuron is given as small. Experimental results show that the CNN archives rate of 87% accuracy with low complexity and compared with the all other state of arts methods.

**Introduction:**

Brain tumor is one of the vital organs in the human body, which consists of billions of cells. The abnormal group of cells is formed from the uncontrolled division of cells, which is also called a tumor. Brain tumours are divided into two types such low grade (grade1 and grade2) and high grade (grade3 and grade4) tumor. Low grade brain tumor is called benign. Similarly, the high grade tumor is also called malignant. Benign tumor is not a cancerous tumor. Hence it doesn’t spread other parts of the brains. However the malignant tumor is a cancerous tumor. So it spreads rapidly with indefinite boundaries to other regions of the body easily. It leads to immediate death.

Brain MRI image is mainly used to detect the tumor and tumor progress modeling process. This information is mainly used for tumor detection and treatment processes.MRI image gives more information about given medical image than the CT or ultrasound image. MRI image provides detailed information about brain structure and anomaly detection in brain tissue.Actually, Scholars offered unlike automated methods for brain tumors finding and type cataloging using brain MRI images from the time when it became possible to scan and freight medical images to the computer. Conversely, Neural Networks (NN) and Support Vector Machine (SVM) are the usually used methods for their good enactment over the most recent few years.11 However freshly, Deep Learning (DL) models fixed a stirring trend in machine learning as the subterranean architecture can efficiently represent complex relationships without needing a large number of nodes like in the superficial architectures e.g. K-Nearest Neighbor (KNN)and Support Vector Machine (SVM).Consequently, they grew quickly to become the state of the art in unlike health informatics areas, for example/medical image analysis, medical informatics and bioinformatics.

In [**deep learning**](https://en.wikipedia.org/wiki/Deep_learning), a convolutional neural network (CNN, or ConvNet) is a class of[**deep neural networks**](https://en.wikipedia.org/wiki/Deep_neural_network), most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and[**translation invariance**](https://en.wikipedia.org/wiki/Translation_invariance) characteristics.They have applications in [**image and video recognition**](https://en.wikipedia.org/wiki/Computer_vision)**,**[**recommender systems**](https://en.wikipedia.org/wiki/Recommender_system)**,**[**image classification**](https://en.wikipedia.org/wiki/Image_classification)**,**[**medical image analysis**](https://en.wikipedia.org/wiki/Medical_image_computing)**,**[**natural language processing**](https://en.wikipedia.org/wiki/Natural_language_processing), and financial [**time series**](https://en.wikipedia.org/wiki/Time_series)**.**

CNNs are[**regularized**](https://en.wikipedia.org/wiki/Regularization_(mathematics)) versions of [**multilayer perceptrons**](https://en.wikipedia.org/wiki/Multilayer_perceptron). Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to [**overfitting**](https://en.wikipedia.org/wiki/Overfitting)data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of contentedness and complexity, CNNs are on the lower extreme.

CNNs use relatively little pre-processing compared to other[**image classification algorithms**](https://en.wikipedia.org/wiki/Image_classification). This means that the network learns the [**filters**](https://en.wikipedia.org/wiki/Filter_(signal_processing))that in traditional algorithms were[**hand-engineered**](https://en.wikipedia.org/wiki/Feature_engineering). This independence from prior knowledge and human effort in feature design is a major advantage.

A convolutional neural network consists of an input and an output layer, as well as multiple [**hidden layers**](https://en.wikipedia.org/wiki/Multilayer_perceptron#Layers). The hidden layers of a CNN typically consist of a series of convolutional layers that *convolve* with a multiplication or other[**dot product**](https://en.wikipedia.org/wiki/Dot_product). The activation function is commonly a [**ReLU layer**](https://en.wikipedia.org/wiki/Rectifier_(neural_networks)), and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final [**convolution**](https://en.wikipedia.org/wiki/Convolution)**.**

When programming a CNN, the input is a[**tensor**](https://en.wikipedia.org/wiki/Tensor)with shape (number of images) x (image height) x (image width) x ([image depth](https://en.wikipedia.org/wiki/Image_depth)). Then after passing through a convolutional layer, the image becomes abstracted to a feature map, with shape (number of images) x (feature map height) x (feature map width) x (feature map channels). A convolutional layer within a neural network should have the following attributes:

* Convolutional kernels defined by a width and height (hyper-parameters).
* The number of input channels and output channels (hyper-parameter).
* The depth of the Convolution filter (the input channels) must be equal to the number channels (depth) of the input feature map.

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus.Each convolutional neuron processes data only for its[**receptive field**](https://en.wikipedia.org/wiki/Receptive_field). Although[**fully connected feedforward neural networks**](https://en.wikipedia.org/wiki/Multilayer_perceptron) can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for *each* neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters.For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learn-able parameters. By using regularized weights over fewer parameters, the vanishing gradient and exploding gradient problems seen during[**backpropagation**](https://en.wikipedia.org/wiki/Backpropagation) in traditional neural networks are avoided.

### **Pooling**

Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer. In addition, pooling may compute a max or an average. *Max pooling* uses the maximum value from each of a cluster of neurons at the prior layer. *Average pooling* uses the average value from each of a cluster of neurons at the prior layer.

### **Fully connected**

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional [**multi-layer perceptron**](https://en.wikipedia.org/wiki/Multi-layer_perceptron)neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

### **Receptive field**

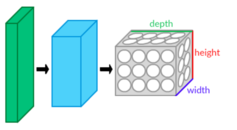
In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from *every* element of the previous layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer. Typically the subarea is of a square shape (e.g., size 5 by 5). The input area of a neuron is called its *receptive field*. So, in a fully connected layer, the receptive field is the entire previous layer. In a convolutional layer, the receptive area is smaller than the entire previous layer. The subarea of the original input image in the receptive field is increasingly growing as getting deeper in the network architecture. This is due to applying over and over again a convolution which takes into account the value of a specific pixel, but also some surrounding pixels.

### **Weights**

Each neuron in a neural network computes an output value by applying a specific function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers). Learning, in a neural network, progresses by making iterative adjustments to these biases and weights.

The vector of weights and the bias are called *filters* and represent particular[**features**](https://en.wikipedia.org/wiki/Feature_(machine_learning)) of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons can share the same filter. This reduces [**memory footprint**](https://en.wikipedia.org/wiki/Memory_footprint) because a single bias and a single vector of weights are used across all receptive fields sharing that filter, as opposed to each receptive field having its own bias and vector weighting.

In the past, traditional [multilayer perceptron](https://en.wikipedia.org/wiki/Multilayer_perceptron) (MLP) models have been used for image recognition.However, due to the full connectivity between nodes, they suffered from the [curse of dimensionality](https://en.wikipedia.org/wiki/Curse_of_dimensionality), and did not scale well with higher resolution images. A 1000×1000-pixel image with[**RGB color**](https://en.wikipedia.org/wiki/RGB_color_model) channels has 3 million weights, which is too high to feasibly process efficiently at scale with full connectivity.



CNN layers arranged in 3 dimensions

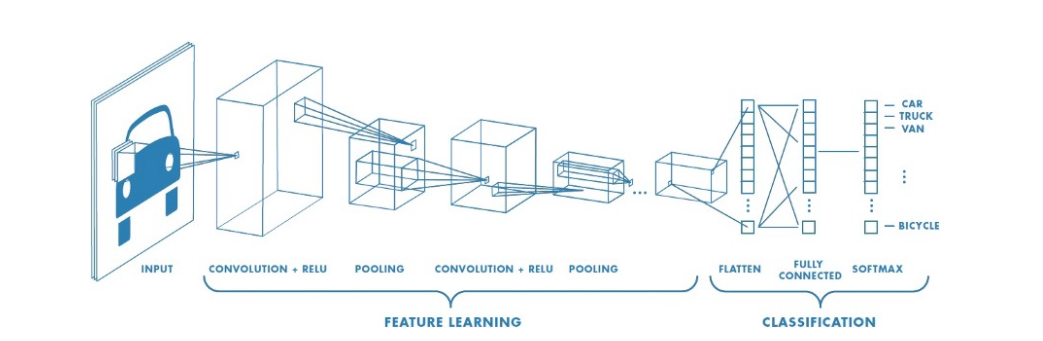
For example, i**n**[**CIFAR-10**](https://en.wikipedia.org/wiki/CIFAR-10), images are only of size 32×32×3 (32 wide, 32 high, 3 color channels), so a single fully connected neuron in a first hidden layer of a regular neural network would have 32\*32\*3 = 3,072 weights. A 200×200 image, however, would lead to neurons that have 200\*200\*3 = 120,000 weights.

Also, such network architecture does not take into account the spatial structure of data, treating input pixels which are far apart in the same way as pixels that are close together. This ignores[**locality of reference**](https://en.wikipedia.org/wiki/Locality_of_reference) in image data, both computationally and semantically. Thus, full connectivity of neurons is wasteful for purposes such as image recognition that are dominated by[**spatially local**](https://en.wikipedia.org/wiki/Spatial_locality) input patterns.

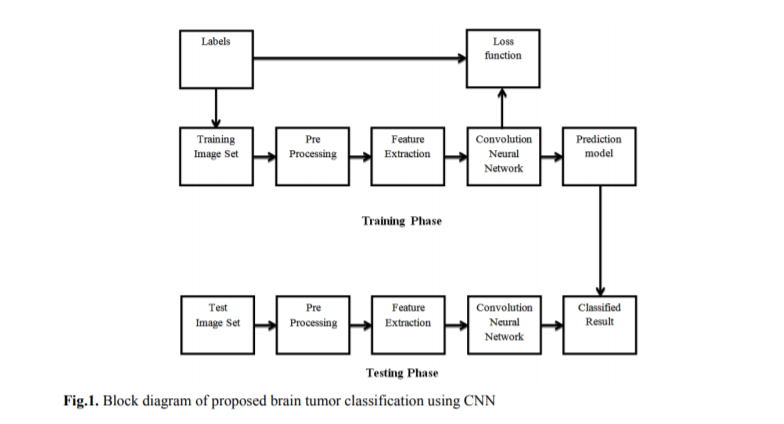
Convolutional neural networks are biologically inspired variants of multilayer perceptrons that are designed to emulate the behavior of a [**visual cortex**](https://en.wikipedia.org/wiki/Visual_cortex)**.** These models mitigate the challenges posed by the MLP architecture by exploiting the strong spatially local correlation present in natural images. As opposed to MLPs, CNNs have the following distinguishing features:

* 3D volumes of neurons. The layers of a CNN have neurons arranged in[**3 dimensions**](https://en.wikipedia.org/wiki/Three-dimensional_space): width, height and depth.] where each neuron inside a convolutional layer is connected to only a small region of the layer before it, called a receptive field. Distinct types of layers, both locally and completely connected, are stacked to form a CNN architecture.
* Local connectivity: following the concept of receptive fields, CNNs exploit spatial locality by enforcing a local connectivity pattern between neurons of adjacent layers. The architecture thus ensures that the learned "[**filters**](https://en.wikipedia.org/wiki/Filter_(signal_processing))" produce the strongest response to a spatially local input pattern. Stacking many such layers leads to [**non-linear filters**](https://en.wikipedia.org/wiki/Nonlinear_filter)that become increasingly global (i.e. responsive to a larger region of pixel space) so that the network first creates representations of small parts of the input, then from them assembles representations of larger areas.
* Shared weights: In CNNs, each filter is replicated across the entire visual field. These replicated units share the same parameterization (weight vector and bias) and form a feature map. This means that all the neurons in a given convolutional layer respond to the same feature within their specific response field. Replicating units in this way allows for the resulting feature map to be [**equivariant**](https://en.wikipedia.org/wiki/Equivariant_map) under changes in the locations of input features in the visual field, i.e. they grant translational equivariance.
* Pooling: In a CNN's pooling layers, feature maps are divided into rectangular sub-regions, and the features in each rectangle are independently down-sampled to a single value, commonly by taking their average or maximum value. In addition to reducing the sizes of feature maps, the pooling operation grants a degree of[**translational invariance**](https://en.wikipedia.org/wiki/Translational_symmetry) to the features contained therein, allowing the CNN to be more robust to variations in their positions.

Together, these properties allow CNNs to achieve better generalization on [**vision problems**](https://en.wikipedia.org/wiki/Computer_vision). Weight sharing dramatically reduces the number of[**free parameters**](https://en.wikipedia.org/wiki/Free_parameter) learned, thus lowering the memory requirements for running the network and allowing the training of larger, more powerful networks.



**BLOCK DIAGRAM OF CNN MODEL**



**Contribution:**

Building a detection model using a convolutional neural network in Tensorflow & Keras.

Used a brain MRI images data founded on Kaggle.

**About the data:**

The dataset contains 2 folders: yes and no which contains 253 Brain MRI Images. The folder yes contains 155 Brain MRI Images that are tumorous and the folder no contains 98 Brain MRI Images that are non-tumorous.

## **Data Augmentation:**

Why did I use data augmentation?

Since this is a small dataset, There weren't enough examples to train the neural network. Also, data augmentation was useful in tackling the data imbalance issue in the data.

Further explanations are found in the Data Augmentation notebook.

Before data augmentation, the dataset consisted of:

155 positive and 98 negative examples, resulting in 253 example images.

After data augmentation, now the dataset consists of:

1085 positive and 980 examples, resulting in 2065 example images.

Note: these 2065 examples contain also the 253 original images. They are found in a folder named 'augmented data'.

## **Data Preprocessing:**

For every image, the following preprocessing steps were applied:

1. Crop the part of the image that contains only the brain (which is the most important part of the image).
2. Resize the image to have a shape of (240, 240, 3)=(image\_width, image\_height, number of channels): because images in the dataset come in different sizes. So, all images should have the same shape to feed it as an input to the neural network.
3. Apply normalization: to scale pixel values to the range 0-1.

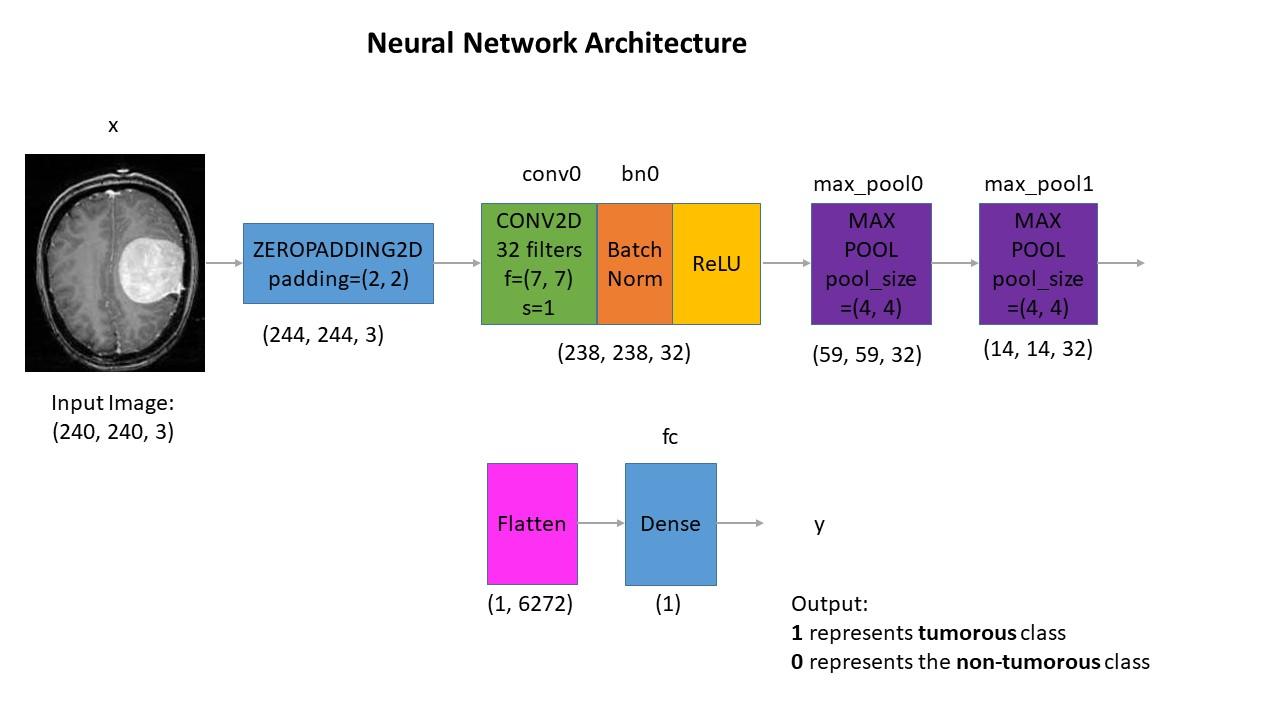
## **Data Split:**

The data was split in the following way:

1. 70% of the data for training.
2. 15% of the data for validation.
3. 15% of the data for testing.

# Neural Network Architecture

This is the architecture that I've built:



**Understanding the architecture:**

Each input x (image) has a shape of (240, 240, 3) and is fed into the neural network. And, it goes through the following layers:

1. A Zero Padding layer with a pool size of (2, 2).
2. A convolutional layer with 32 filters, with a filter size of (7, 7) and a stride equal to 1.
3. A batch normalization layer to normalize pixel values to speed up computation.
4. A ReLU activation layer.
5. A Max Pooling layer with f=4 and s=4.
6. A Max Pooling layer with f=4 and s=4, same as before.
7. A flatten layer in order to flatten the 3-dimensional matrix into a one-dimensional vector.
8. A Dense (output unit) fully connected layer with one neuron with a sigmoid activation (since this is a binary classification task).

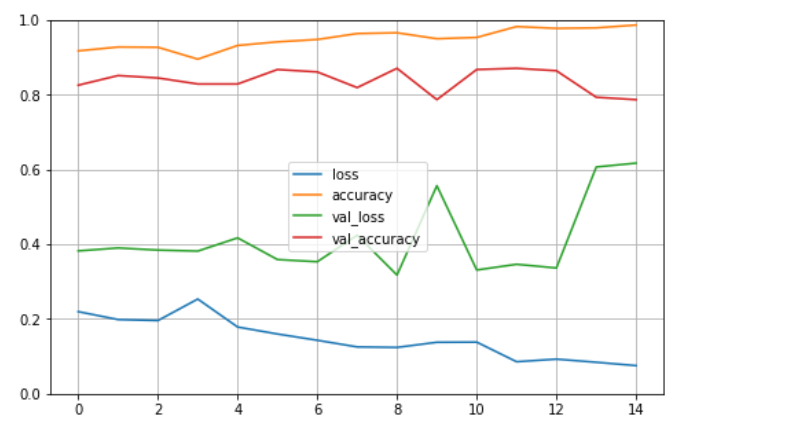
**Why this architecture?**

Firstly, I applied transfer learning using a **ResNet50** and **vgg-16**, but these models were too complex for the data size and were **overfitting**. Of course, you may get good results applying transfer learning with these models using data augmentation. But, this will take a lot of time to train the model . So, I had to take into consideration computational complexity and memory limitations.

So why not try a simpler architecture and train it from scratch and it worked.

# **Training the model:**

The model was trained for **25 epochs** and these are the loss & accuracy plots:



The best validation accuracy was achieved on the 22nd iteration.

# **Results**

Now, the best model (the one with the best validation accuracy) detects brain tumor with:

**87.1**% accuracy on the test set.

**0.87** f1 score on the test set.

These results are very good considering that the data is balanced.

**Reference:**

1. [biomedpharmajournal](https://biomedpharmajournal.org/vol11no3/brain-tumor-classification-using-convolutional-neural-networks/#:~:text=Especially%2C%20in%20this%20work%20MRI,diagnose%20tumor%20in%20the%20brain.&%20text=In%20this%20work%2C%20automatic%20brain,neuron%20is%20given%20as%20small.)
2. [Convolutional neural networks for brain tumour segmentation](https://insightsimaging.springeropen.com/articles/10.1186/s13244-020-00869-4)
3. [Wikipedia CNN](https://en.wikipedia.org/wiki/Convolutional_neural_network)