

Medical Image Classification Using Deep Neural Network

Pallab Paul

Assistant Professor

Department of Computer Science and Engineering, Government Engineering College Kishanganj,
Mahesh Bathna, Kishanganj, Bihar, India

ARTICLE DETAILS

Research Paper

Keywords :

X-radiation,

Magnetic Resonance

Imagging,

Computed Tomography,

Neural Network,

*Convolutional Neural
Network,*

Natural Language

Processing

ABSTRACT

Tumor in the brain is one of the most dangerous sickness or diseases caused by quickly partitioning of uncontrollable and abnormal cells that have formed in the brain. Apart from that a tumor conform in any part of the human body. Some tumors are cancerous (malignant), while others are not(non-cancerous).The healthcare industry has greatly benefited from recent developments in the deep learning process, especially in the field of medical imaging for the diagnosis of various medical disorders. In this work, I present a new architecture that uses Image Processing techniques along with Convolutional Neural Network (CNN) methodology to classify brain MRI or CT scan images into two groups: those that show evidence of a tumour and those that do not. Here, I want to find the highest accuracy by changing layers, number of neurons and epochs. Basically, along with the convolutional layer and maxpooling layer here in this project I used more dense or fully connected layer in a suitable mathematical way to find more accuracy. Because, in our human brain every neuron is connected with other neurons. In this paper, I have used the same mechanism and more dense layers or fully connected layers. It is an imperative aspect of medical science to visualize the internal structures of the brain as well as to diagnose, monitor and treat diseases. Several types of imaging technologies like MRI scan; CT scan and X-ray are used in them edictal field to diagnose the body parts of humans. In this paper I used



Brain images where some have tumor and some images don't have any tumor and here I want to find highest accuracy by changing layers, number of neurons and epochs. Basically, along with the convolutional layer, maxpooling layer and dense layer here in this paper I used more dense layer or fully connected layer in a suitable mathematical way to find more accuracy.

Introduction

The human brain stands as one of the most crucial organs in the human physique. Similarly, brain tumors, composed of billions of cells, also hold paramount importance within the human body.

Tumours are an abnormal collection of tissue caused by unchecked cell growth. They can appear in the brain in two different forms: low-grade, also called benign, and high-grade, or malignant. Benign tumours do not spread quickly to nearby brain areas since they do not possess malignant characteristics. On the other hand, malignant tumours have cancerous characteristics that cause them to spread quickly and uncontrollably to other parts of the body, frequently with deadly consequences.

The major methods for finding the tumours are based on brain MRI or CT scan pictures. Additionally, the primary application of this data is to categorise a picture into two groups: "there is a tumour" and "there is no tumour" in the image. Compared to a CT scan or other ultra scan pictures, an MRI image provides more information on the specified medical imaging.

Medical anomalies include a variety of illnesses and ailments, including TB, diabetic retinopathy, interstitial lung diseases, glaucoma, and tumours. Finding anomalies, determining their borders and locations, and determining their size and severity are all part of the process of analysing medical imaging. Furthermore, these medical conditions differ greatly in their forms, structures, and locations, making it difficult for even highly skilled doctors to make an accurate diagnosis. As a result, there is an increasing need for auxiliary tools among medical practitioners to help with the careful interpretation of medical pictures. The need for this is what motivates the creation of sophisticated picture-understanding systems.

The brain, which controls essential bodily processes and traits, is the most fragile organ in the human body. According to the National Brain Tumour Society survey, around 700,000 Americans received a brain tumour diagnosis; by the end of 2020, that number had increased to 787,000. According to the International Association of Cancer Registries (IARC), brain tumours were the tenth most common kind of tumour among Indians in 2018. Worldwide, brain tumours claim the lives of more than 24,000 people per year [1].

A tumour has a profound and enduring psychological effect on a person's life. The tumour is brought on by tissues that grow in the brain or central spine inappropriately, impairing normal brain function.

Image Processing in Medical Science

Modern medical imaging is unique in that it is a critical process that produces clear and bright images of the body's interior organs, enabling research as well as therapeutic procedures. Furthermore, this imaging technique is quite helpful for seeing how interior tissues work. The quick development of medical imaging may be attributed in large part to advances in image processing techniques, which include image perception, recognition, analysis, and enhancement. These methods are essential for improving the accuracy and amount of tissues found.

The field of medical picture categorization is becoming increasingly important, attracting the attention of researchers and the medical community alike. Significant investments and developments in medical imaging modalities, including ultrasound (USG), magnetic resonance imaging (MRI), and X-rays, provide compelling incentives for researchers to apply new medical image processing algorithms in a variety of domains to produce compelling results.

As well as, Image segmentation is considered the foremost essential medical imaging process because it extracts the region of interest through a semiautomatic or automatic process. It divides an image into areas supported a specified description, like segmenting body organs or tissues within the medical applications for border detection, tumor detection or segmentation and mass detection. Because after we did these tasks manually then it could be wrong and take much time to induce answer but after we did these through a particular computerize algorithms then we did these with far more correctly and it's take much more less time than a personality's.

Although, imaging medical is critical for the visualization of internal organs for the detection of abnormalities in their anatomy or functioning. Medical images capture from medical technologies, such as X-ray, CT scan, MRI, PET and ultrasound scanners. So through these devices body organs are captured which is that the anatomy or functioning of the interior organs and present them as images or videos. The images must be understood for the accurate detection of anomalies or the diagnosis of their functional abnormalities of human body. If there's an abnormality, then its exact location, size and shape must be determined. These tasks are traditionally performed by the trained physicians supported their judgment and knowledge. Modern intelligent healthcare system's aim is to perform these tasks with intelligent medical image understanding with various processes. Medical image classification, segmentation, detection and localization are the important tasks to understand the medical image and understanding is also very important to work on this domain [2].

So, human expertise is commonly essential to translate raw data into the set of useful features with the assistance of feature extraction algorithms.

Image Classification

Image classification is also important part in recent times in digital image processing. In my project I want to do Image Classification task by dividing all images into two groups, one is 'Have Tumor' basis and other is 'Don't have tumor' basis from a set of some images from a folder. And doing this using Neural Network. Further details are given in next pages. Neural Network is the best way of doing Image Classification process [3].

Image Classification is done by Machine Learning process. This process is quite old but at that time we don't have sufficient data to do this task but in modern days we have lots of data to do this task. For example, recent times Facebook generates 4 petabytes of data per day. Using these data now it is easy to learn any models.

Machine learning is the study of computer algorithms that improve over time by using large datasets and acquired knowledge. It is an essential part of AI, or artificial intelligence.

Neural Network

Neural Network is that “the process that mimics the way that the human brain operates”. Through a valid dataset here we learn the Neural Network to get appropriate result and accuracy.

A NN is basically a group of algorithms that trying to get acknowledge underlying relationships using a set of data with a specific type of process that the way of how human brain operates in our body. During this sense, neural networks talk over with systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; that the network generates the most effective possible result with no need to revamp the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity within the development of trading systems.

Basics Of Neural Network: Neural networks, assist within the development of such process as time-series forecasting, algorithmic trading, classification, credit risk modeling for software and constructing proprietary indicators, price derivatives and as well as in the medical field. But here I mainly focused on the medical imaging filed [4].

Similar to the neural networks seen in the human brain, a neural network functions. A "neuron" in a neural network is a mathematical unit that collects and classifies input according to a selected structure. The network has a lot of parallels to statistical methods like regression analysis and curve fitting.

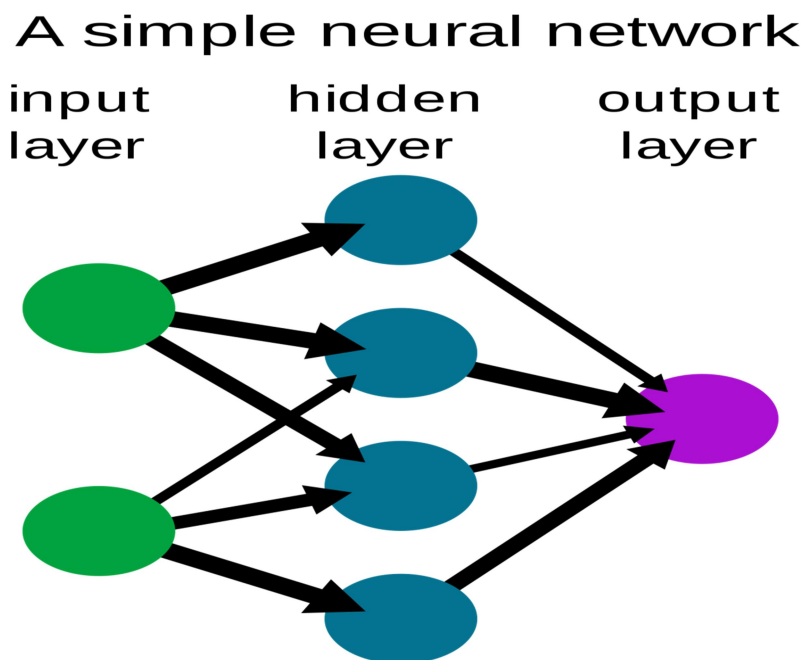


Fig1: A Simple Neural Network (src: Image by Sabrina Jiang © Investopedia 2020)

A NN contains layers of interconnected nodes. Each node is generally a perceptron and is comparable to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function which may be nonlinear.

Every multi-layered perceptron (MLP) has layers that are linked by perceptrons. Identifying patterns in a dataset—which may contain labeled or unlabeled data—is done by the input layer. Input patterns can be mapped using the categorization models in the output layer.

Hidden layers updated the input weightings again and again until the neural network's margin of error is minimal. It's hypothesized that hidden layers extrapolate the salient features within the input file that have the most power regarding the outputs. This describes feature extraction, which accomplishes a utility quite like statistical techniques like principal component analysis.

In NN there are three layers; first one is input layer then second layer is hidden layer and final layer is called output layer. Every layer has an activation function which is discussed later in this report.

The input layer displays the input vector's dimensions.

The intermediary nodes that create edge-containing areas in the input space are represented by the "hidden layer". Its value uses an activation function between each hidden layer to generate output from a collection of weighted input.

The output layer of any neural network shows the results of a particular job.

Neural Network and its types

These kinds of networks are implemented supported the mathematical operations and a group of parameters required to obtain the output. Let's see some of the neural networks:

Feed-forward Neural Network: Neural networks are built on a foundation of feed-forward neural networks. This network only allows information to go in one direction, from input to processing node to output. Because hidden layers are not a part of feed-forward neural networks, they are more transparent than other neural network topologies. This method, which is often referred to as forward propagation, usually makes use of an activation function for categorization. The output layer of a feed-forward neural network receives the total products that are produced by multiplying the input values by their corresponding weights. An example of a single-layer feed-forward neural network is shown below.

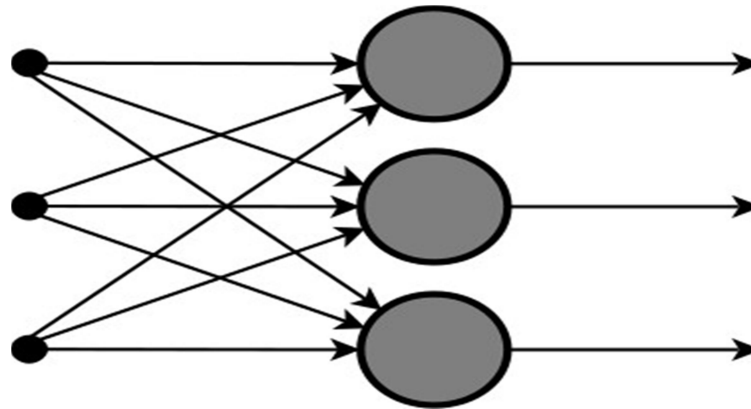


Fig2: A Simple Feed-forward NN

Computer vision and face recognition systems employ feed-forward neural networks. This is due to the challenging nature of the target classes in this kind of NN. Maintaining feed-forward neural networks is also comparatively easy.

Recurrent Neural Network: In modern situations, recurrent neural networks (RNNs) are becoming more complex networks. Data travels over this kind of network in several directions. RNNs store the output data that processing nodes produce and use learning to improve their functions. Weights and features are multiplied in the first layer, which is organized similarly to the feed-forward system. Still, later layers are where the recurrent neural network process starts.

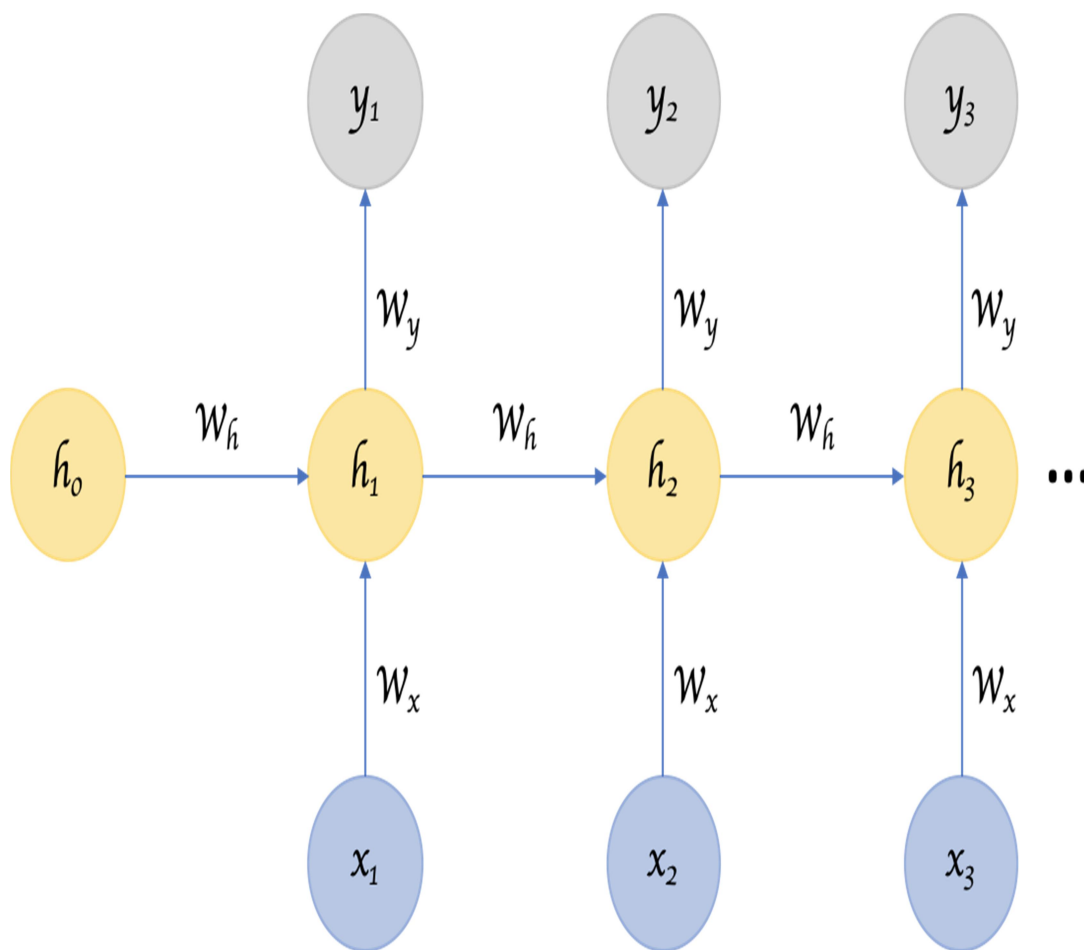


Fig3: Recurrent Neural Network

Convolutional Neural Network: The efficiency of Convolutional Neural Networks (CNNs) in tasks including facial recognition, classification, and medical applications has led to their increasing popularity. Working on the premise that the input represents a picture, they are excellent in encoding qualities into input data.

CNNs and feed-forward neural networks are similar in that their neurons learn by changing their weights and biases.

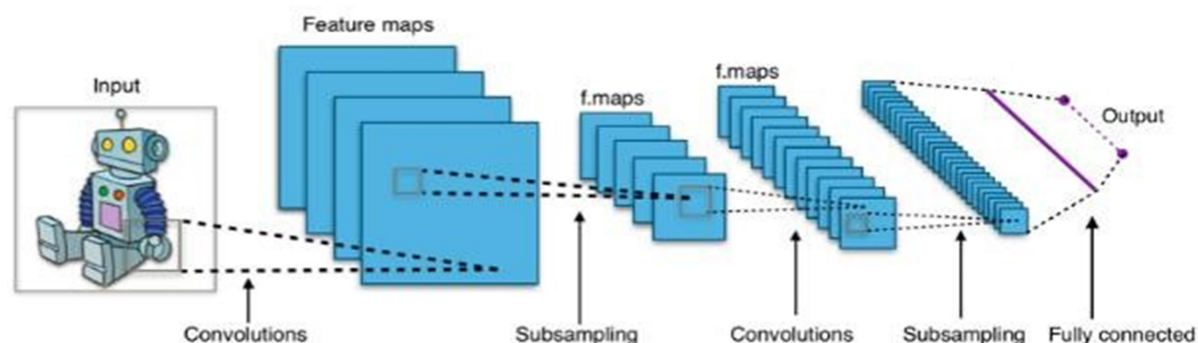


Fig4: Convolutional NN

A variant of the multilayer perceptrons is used by a CNN. Additionally, CNN may include one or more convolutional layers. All of these levels are pooled together or fully integrated. Talk about the in the pages that follow. We'll talk more about CNN later.

Radial Basis Functions Neural Network: Radial Basis Function Network consists of an input vector which is followed by a layer of RBF neurons and an output layer with one node per category. This kind of neural network encodes prototypes in each neuron, and the classification operation is carried out by comparing the input to data points from the training dataset.

When a new input vector needs to be classified, each neuron calculates the Euclidean distance between the input and its prototype. For example, if we have two classes i.e. class X and Class Y, then the new input to be classified is more close to the class X prototypes than the class Y prototypes. Hence, it could be tagged or classified as class X.

This raises the possibility of a blackout. In order to restore electricity as quickly as possible, power restoration systems employ this kind of neural network. [9]



Fig5: Radial Basis Functions Neural Network

Multi-Layer Perceptron: It functions as the first step towards complex neural networks, allowing input data to flow through several layers of artificial neurons inside the framework. A completely connected neural network is one in which every neuron in one layer is linked to every other neuron in the layer below it. Any architecture consists of one or more hidden layers, input and output layers, and at least three levels overall. Bidirectional propagation, including both forward and backward propagation, is a feature of this kind of network.

To put it simply, weights for neural networks are values that can be learned by machines. Depending on how training inputs and anticipated outputs differ, they self-adjust. As output layer activation functions, nonlinear activation functions are employed first, and then either softmax or sigmoid.

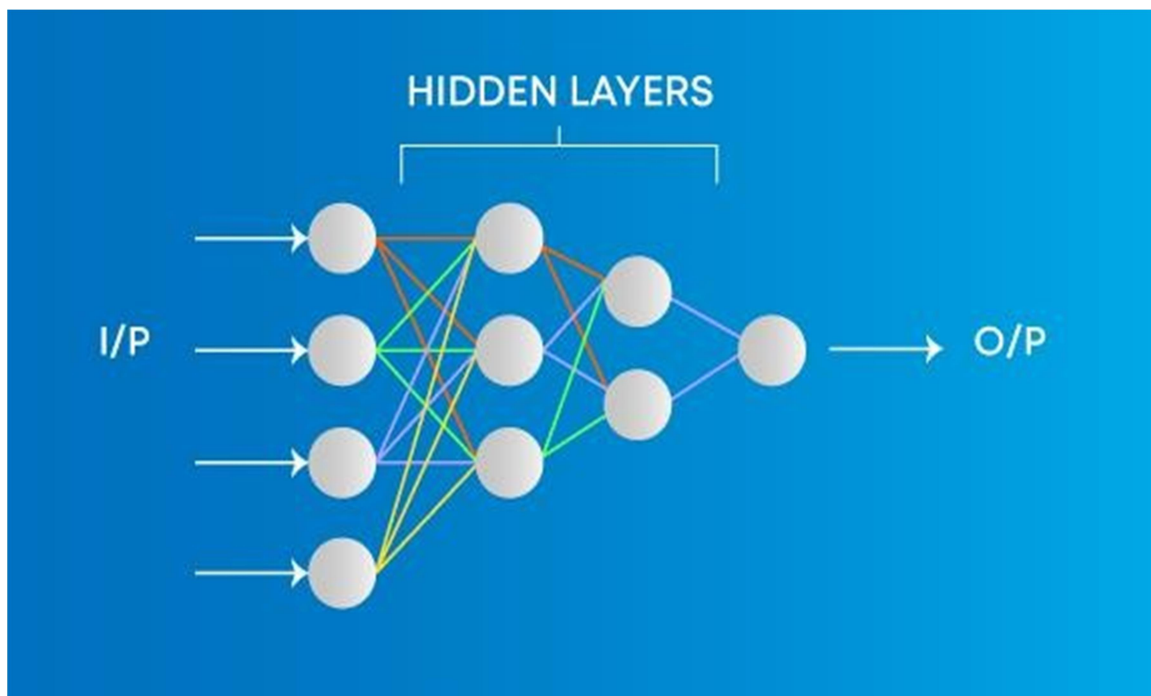


Fig6: An Example of MLP

Convolutional Neural Network

Continuous fast growth is observed in computer vision, with deep learning being a major contributor. Convolutional neural networks (CNNs) stand out in this context since they are widely used in computer vision technologies. CNNs are used for image categorization and facial recognition. Like other simple neural network architectures, CNNs include learnable parameters like weights and biases [5].

Convolutional neural networks (CNNs) are a subset of deep neural networks that are mostly applied to the analysis of visual vision in the context of deep learning. They function using the shared-weight design of convolution kernels, which shift across input characteristics and provide translation-equivariant responses. These artificial neural networks are sometimes referred to as shift-invariant or space-invariant artificial neural networks (SIANNs). It's interesting to note that most CNNs exhibit equivariance to translation rather than invariance, despite their name. CNNs are used in many different

fields, including image and video recognition. These fields include natural language processing, medical image analysis, financial time series analysis, image classification, and image segmentation.

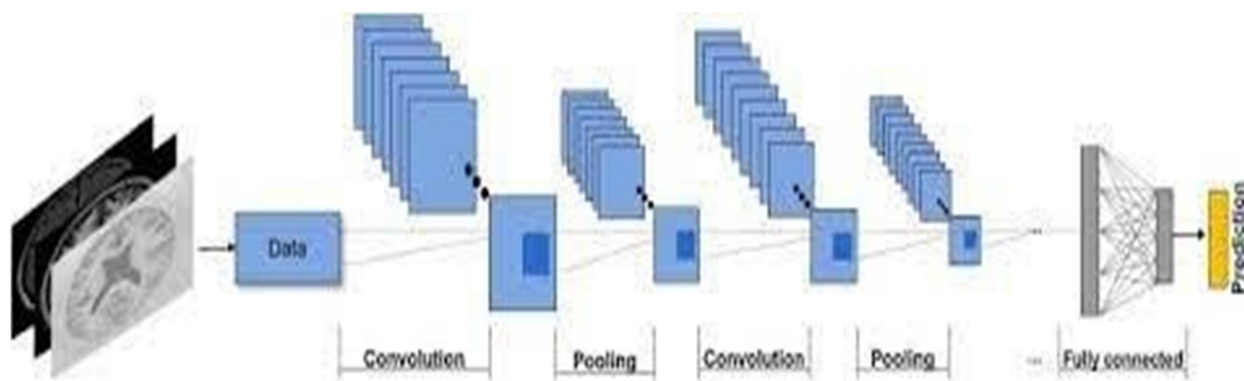


Fig7: CNN to classify Brain Image on the basis of Tumor

The convolutional layer applies a convolutional operation to the input and generates an output for the next layer before forwarding the result to the subsequent levels. The network can have far less parameters and a considerably greater depth due to this convolutional process.

Comparing Convolutional Neural Networks (CNNs) to other image classification techniques, CNNs require comparatively less pre-processing. This suggests that, in contrast to traditional algorithms, where these filters are manually constructed, the network automatically learns to optimise filters (or kernels) through automated learning. One major benefit of CNNs is their ability to extract features independently, without the need for human involvement or prior knowledge [6].

Why We Use CNN?

Assume I am working with the MNIST collection, where each grayscale picture has dimensions of 28 x 28 x 1. In this configuration, there is a manageable total of $28 \times 28 = 784$ neurons in the input layer. But if the image size was increased to 1000 x 1000, the input layer would require an astounding 10^6 neurons,

which would make it computationally inefficient. Convolutional Neural Networks, or CNNs, come into play here. In essence, CNNs take elements out of pictures and resize them to smaller sizes while maintaining all of the important details. Take into consideration, for example, an initial picture size of $224 \times 224 \times 3$ (which denotes a colour image). The input layer would require $224 \times 224 \times 3 = 100,352$ neurons if convolutional procedures were not used. Nevertheless, the input tensor dimension is decreased to $1 \times 1 \times 1000$ after applying a convolutional layer, requiring just 1000 neurons in the CNN's first layer [7].

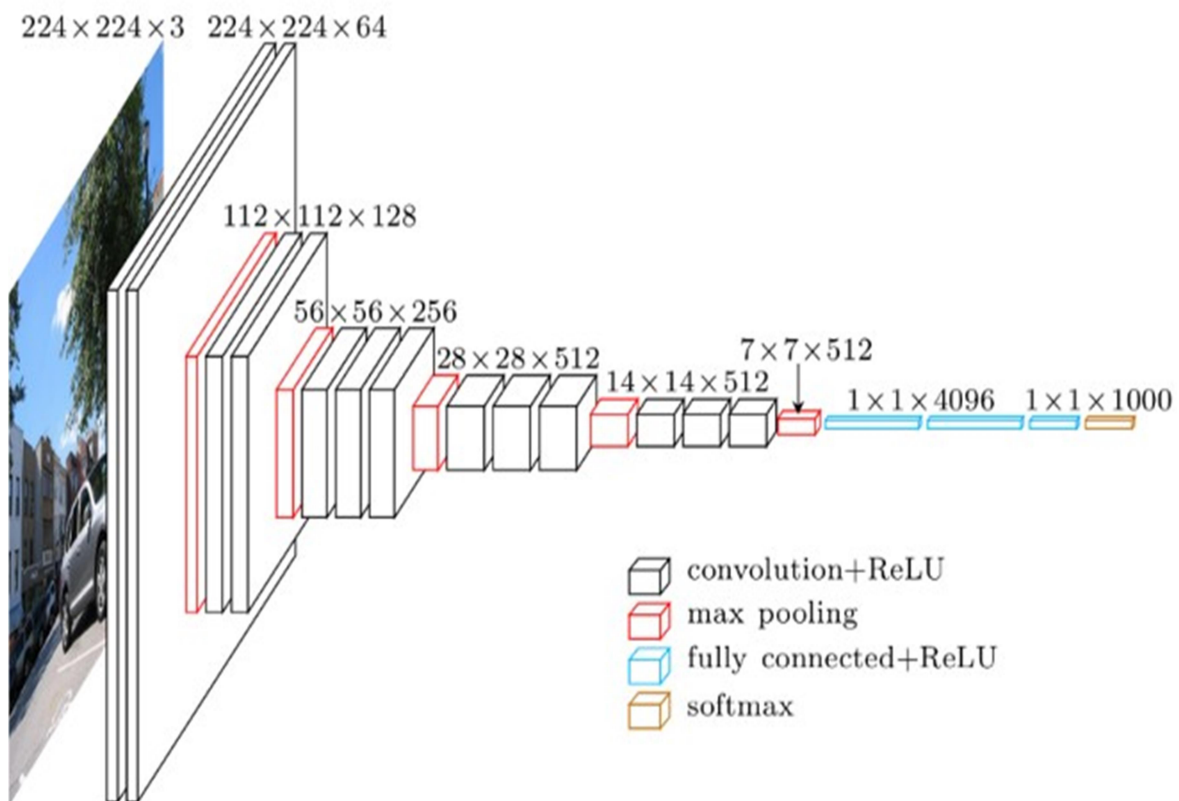


Fig8: Down sampling

(src: <https://towardsdatascience.com/convolutional-neural-network-cb0883dd6529>)

CNN Layers

Input layer: The input layer of a CNN has to be able to hold the picture data in the right way. Three-dimensional matrices, or two-dimensional matrices for grayscale pictures, are commonly used to describe image data. We must resize the image to fit into a single column before feeding it into the network. For example, we must convert a picture with dimensions of $30 * 30 = 900$ into a column vector with dimensions of $900 * 1 = 900$. In the event that 'n' training samples are present, the input's dimension will be $(900, n)$.

Convo layer: A crucial part of extracting features from an image is the convolutional layer, often known as the feature extractor layer. The convolution process is first carried out by the convolutional layer, which calculates the dot product between the filter and the receptive fields, which are local areas of the input picture the same size as the filter. The output volume is a single integer that is produced by this operation. The filter is then moved by a predetermined stride across the next receptive field of the input picture, repeating the process to produce the output for the layer after that. Until the entire image has been scanned, this process is repeated. The output that is produced acts as the input for the layer that follows.

Max Pooling layer: After convolution, the input image's spatial volume is decreased using the pooling layer. In between two convolution layers, it is utilized. We do not want the computationally costly result that occurs from putting FC straight after the Convo layer without first applying the pooling layer. Therefore, the pooling layer is the sole method for reducing the input image's spatial volume. We have used max pooling in a single depth slice with a stride of two in the example above. As you can see, the input's four-by-four dimensions have been shrunk to two by two.

There's no parameter in pooling layer but it has two hyperparameters — Filter (F) and Stride (S).

In general, if we've input dimension $A_1 \times B_1 \times C_1$, then

$$A_2 = (A_1 - F) / S + 1 \quad B_2 = (B_1 - F) / S + 1 \quad C_2 = C_1$$

where A_2 , B_2 and C_2 are the width, height and depth of output.

Fully connected (FC) layer or Dense Layer: Fully connected layer involves weights, biases, and neurons. All the neurons between every layer is connected with one another. Using training, this method is used to categories photos into several categories.

Softmax/logistic or Sigmoid layer or Activation Function: The Softmax or Logistic layer is the term commonly used to describe the last layer of a CNN. Located at the end of every dense and convolutional layer, it uses various activation functions according to the classification job. A logistic activation function is used for binary classification, while a softmax activation function is used for multiple-class classification.

Output layer: The output layer has values that are dependent on the number of outputs required. I now comprehend CNN rather well. Now let's put a CNN into Keras.

Perceptron

In its most basic form, a perceptron is an algorithm for binary classifier supervised learning. Binary classifiers determine if an input, which is often expressed as a set of vectors, is a member of the chosen class.

Perceptrons are, in essence, single-layer neural networks. They are composed of four primary components: an activation function, weights and bias, and input data.

The first step of the method is to multiply each input value by the appropriate weight. To get the weighted total, the product of these multiplications is then added together. The activation function used to this weighted sum defines the perceptron's output. In order to guarantee that the output is inside the intended range, such as (0,1) or (-1,1), the activation function is essential. Notably, the input's weight

indicates the strength of the node, and the input's bias value allows the activation function curve to be pushed either upward or downward [8].

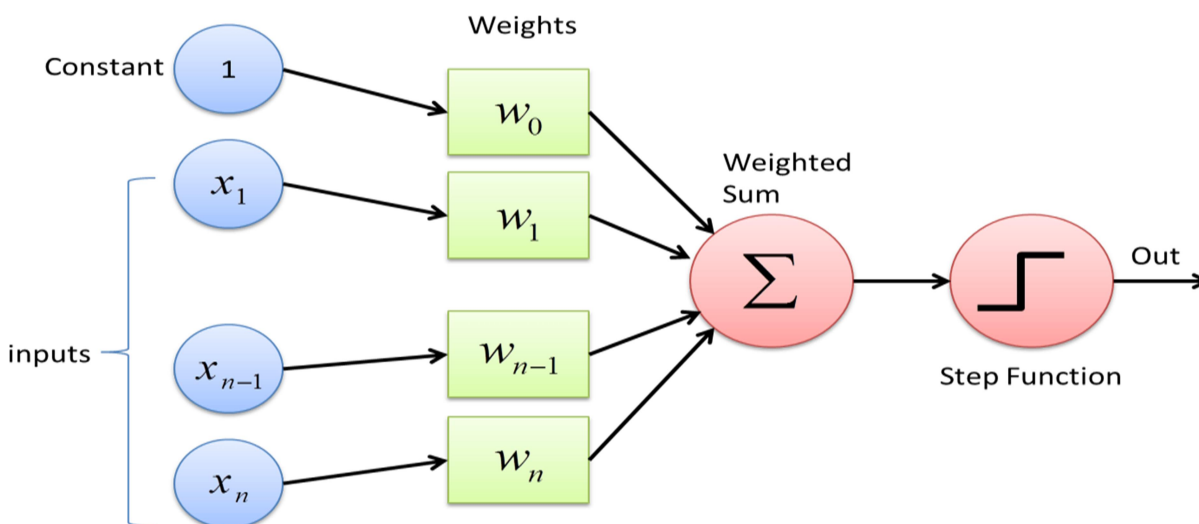


Fig9: A Simple Example of Perceptron

System Required

Processor: i3 based 4th generation processor, 1.7 GHz RAM: 4 GB

Hard Disk: 240 GB (SSD)

Language: Here, I used Python as a programming language.

Some packages that I have used:

Opencv, Numpy, Tensorflow, Keras, Os.

Here I used 'Spyder IDE' to compile my program.

In my work I have total 3000 images. There are 1500 images in 'Don't have tumor' folder and 1500 images in 'Have tumor' folder. (Dataset: Some are random and some from Kaggle.com)

Literature Survey: Convolutional Neural Network and Some Existing CNN Architecture

In the field of artificial intelligence, a convolutional neural network, or CNN, functions similarly to a feed-forward neural network. CNN uses multidimensional arrays to represent input data and is mostly used for image recognition applications. It performs best when given a lot of labelled data. Every part of the input picture, referred to as the receptive field, is analysed in detail by CNN. Each neuron is given a weight, which establishes its relative significance in the receptive field. This weighting helps determine the importance of neurons about each other. Convolution, pooling, and fully linked layers are the three primary types of layers that make up CNN's architecture, as seen in Fig. 1 [9].

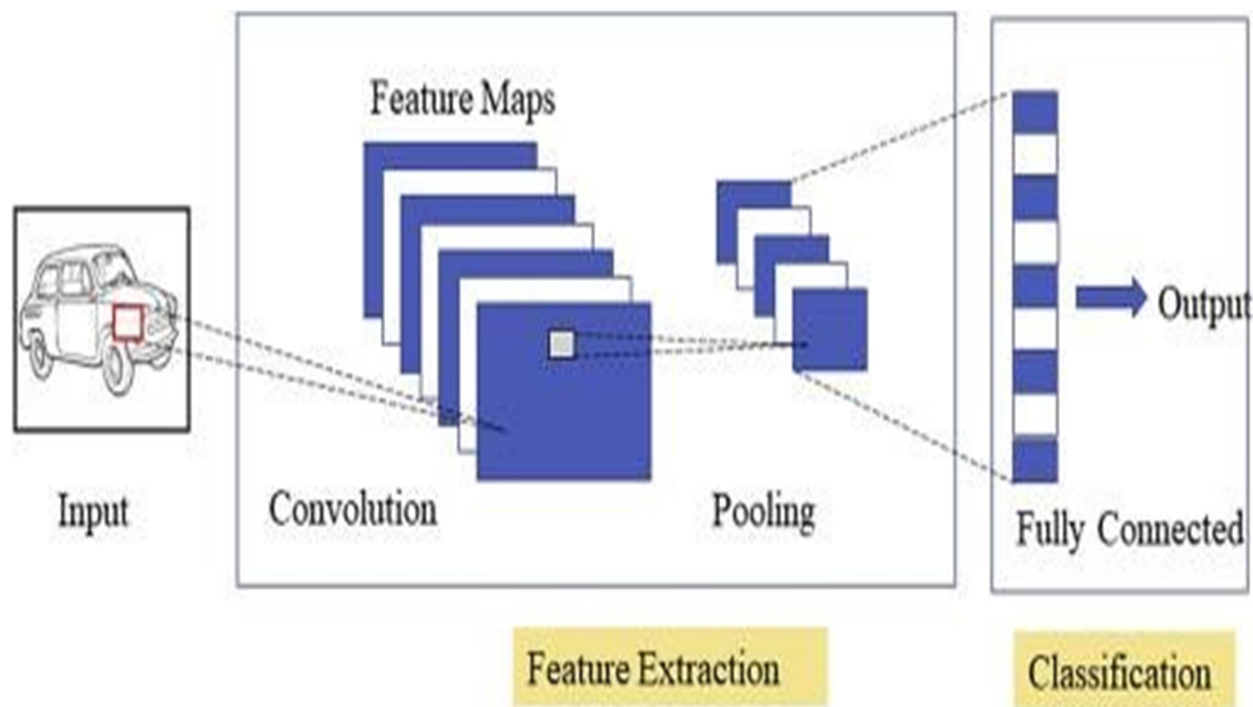


Fig10: Structure of Convolutional Neural Network

(src: "<https://www.sciencedirect.com/topics/engineering/convolutional-neural-network>")

LeNet & It's Architecture

Convolutional Neural Networks are the foundation of deep-learning based computer vision and computer based image classification problems. LeNet was the first convolutional neural network. And it is used to classify two dimensional images, two dimensional images means gray scale images.

LeNet was created during a period when CPUs were extremely sluggish and there was no GPU to aid in training. For this reason high resolution images can't be used to train in LeNet architecture. So, LeNet was trained on gray scale images and the shape of all the images are $32 \times 32 \times 1$. That means the height and width is 32 respectively and the number of channels is 1. Number of channels is equal to 1 that indicates gray scale images. The goal of LeNet is to recognize handwritten digits on bank checks.

LeNet Layer Structure:

The LeNet layer structure is very simple. There are only 7 layers in this structure which is shown in Fig2. Input to the LeNet network is 32*32*1. Image size is the input to the network. Two convolutional layers, two average pooling layers, two fully linked convolutional layers, and one output layer make up this network.

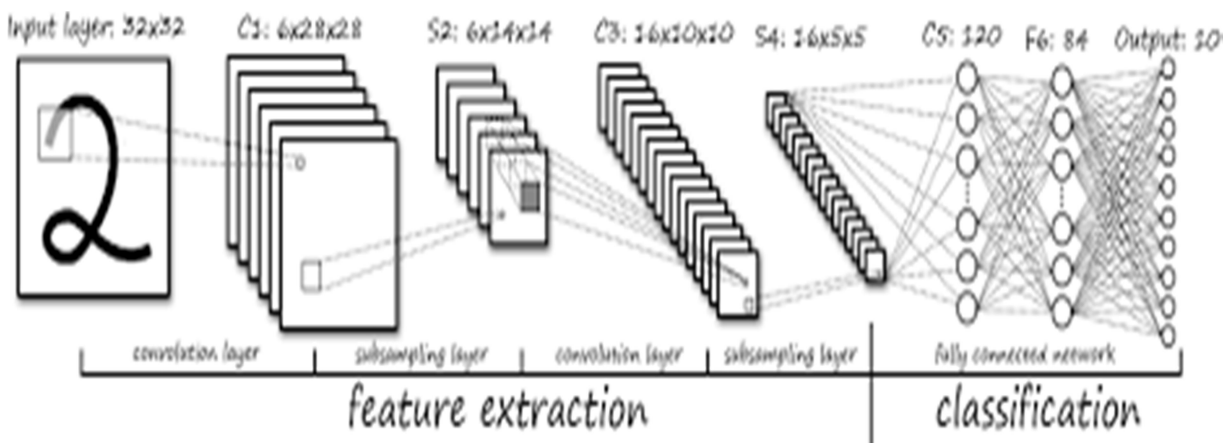


Fig11: LeNet Architecture

There is a formula to reduce the image’s size which is:

$$\frac{n + 2p - f + 1}{s} * \frac{n + 2p - f + 1}{s} \dots\dots\dots\text{equ}(1)$$

In the first convolutional later, there are 6 filters or feature maps having size of 5*5 and a stride of 1. That 32*32*1 size image is used as an input to the first convolutional layer then after this convolutional operation the output of this layer is reduced to 28*28*6 using the equ(1). The image dimensions change

from $32*32*1$ to $28*28*6$. Number of channels always depends on the number of filters used here in that layer. Number of channels are always same as the number of filters. In essence, the ReLU activation function is responsible for identifying the output of a neural network as "yes" or "no". The ability of the ReLU activation function to specifically activate neurons distinguishes it from other activation functions. ReLU does not activate every neuron at once, in contrast to some other activation functions. Consequently, weights and biases for specific neurons do not change during the backpropagation process. As a result of this condition, "dead neurons" may form that eventually stop functioning.

The average pooling layer is the following layer. The utilisation of the average pooling approach might lead to the smoothing out of the picture, making it difficult to identify sharp features. Stated otherwise, the objective of this pooling layer is to minimise the image's size while improving it. In this case, the pooling layer employs a side window filter of $2*2$ with a stride of 2. By using stride 2, the image will automatically get straight and the image size will get reduced. The input to the average pool layer is $28*28*6$. And then by applying this pooling operation, the pooling layer output is $14*14*6$ by the equ(1).

Another convolutional layer makes up the third layer. The input size in this layer is $14*14*6$. The 16 filters that make up the convolutional layer have sizes of 55, strides of 1, and paddings of 0. Equation (1) is used, and all parameters—feature maps, stride, and padding—are used to calculate the output size, which comes out to be $10*10*16$ with 16 channels. The neural

network's output is once more determined using the ReLU activation function, which categorises it as "yes" or "no".

The average pooling layer is the following layer, and its input size is determined by the output of the layer before it. Additionally, this layer's goal is to minimise the image's size. Where the filter size equals $2*2$ and the stride equals 2. Thus, the output will be decreased from $10*10*6$ to $5*5*16$ by applying Equation (1).

The first 4 layers are convolutional layers to do convolutional operations. In these layers the image size is reduced from $32*32*1$ to $5*5*16$ and features increase from 1 to 16. These layers are used to collect features from an image.

As we go to the fifth layer, we come across a completely linked layer, sometimes referred to as a dense layer that consists of 120 feature mappings, or neurons, that are responsible for classification and recognition tasks. The result of the previous layer is multiplied to create a 1-D vector picture to be ready for this assignment. 400 pixels (5516), the result of the fourth layer, provide the input for the completely linked layer. All 120 neurons in this completely linked layer are 1*1 in size. Before sending the 400 pixels to the next completely linked layer, each neuron processes them.

The sixth layer is a fully connected layer with 84 feature maps or neurons. All the functions are the same as the previous layer.

At last, there is an output layer y that is fully linked and has 10 potential values that correspond to the numbers 0 through 9. In this case, the softmax activation function was utilised to choose the firing neuron.

Since neural network models forecast the probability distribution, the softmax function is employed as the activation function in the output layer. Since more than two class labels need class membership, the activation function for multiple-class classification problems is softmax.

AlexNet & It's Architecture

Eleven layers make up the architecture: three max-pooling layers, three fully-connected layers, and five convolutional layers. But it's not only the layer arrangement that distinguishes AlexNet; instead, it's the creative ways in which convolutional neural networks are applied that include the following characteristics:

ReLU Nonlinearity: Rectified Linear Units (ReLU) are used by AlexNet in place of the traditional tanh function. ReLU has an advantage in training, as seen by the fact that a CNN using ReLU completed tasks on the CIFAR-10 dataset six times quicker than a CNN using tanh, with an error rate of just 35%.

Multiple GPUs: In contrast to the considerably bigger capacities accessible now, GPUs of the past had restricted memory capabilities, usually about 3 gigabytes. For 1.2 million photos in the training set, this constraint was even worse. Larger models could therefore be trained, although doing so also required longer training sessions.

Overlapping Pooling: CNNs often "pool" the non-overlapping outputs of nearby classes of neurons. Nevertheless, the scientists found that models with an overlapping pooling approach typically find it more difficult to overfit, as well as observed an error discount of up to 0.5% when they included the overlapping method.

Data Augmentation: The authors employed the label-preserving transformation technique to increase the diversity of their data. They specifically carried out horizontal reflections and picture translations, which expanded the training dataset. Additionally, I altered the RGB channel intensities by doing Principle Component Analysis (PCA) on the picture RGB pixel values, which decreased the top-1 error rate by more than 1%.

Dropout: This method involves "switching off" neurons with a preset probability. This gives the model more resilient features that can be applied to other random neurons since each iteration utilises a new sample of the model's parameters. Dropout, however, also lengthens the training period required for the model to converge.

The best-performing model in the 2010 ImageNet competition had a top-1 error rate of 47.1% and a top-5 error rate of 28.2%. But with a top-1 error rate of 37.5% and a top-5 error rate of 17.0%, the AlexNet design greatly outperformed this. Notably, off-center objects may be recognised by AlexNet thanks to its architecture, and each image's top five projected classes are usually plausible.

AlexNet Layer Structure:

AlexNet architecture has 11 layers which are shown in Fig3. The input size to the AlexNet network is 227*227*3.

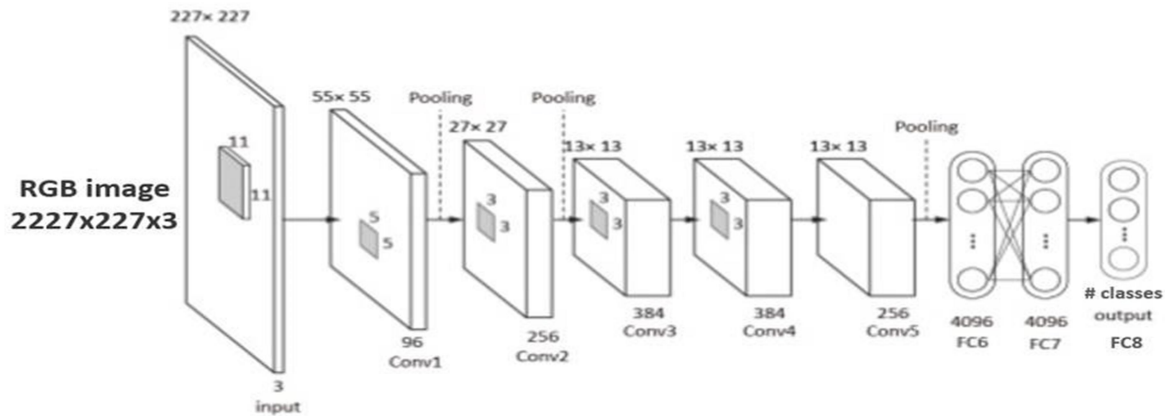


Fig12: AlexNet Architecture

There are 96 filters in the first convolutional layer, each with an 11*11 size and a 4 stride. The first convolutional layer uses the picture with dimensions of 227*227*3 as its input. Equation (1) is then utilised to decrease the layer's output to 55*55*96. The number of filters used in this layer determines the number of channels in every case. Here, the output of a neural network, such as yes or no, is determined using the Relu activation function.

The max pooling layer is the following layer. The process of pooling data that determines the maximum value in each patch of each feature map is known as max-pooling pooling. As a consequence feature maps that are down-sampled or pooled and show the feature that is most prevalent in the patch—rather than the features average presence in the case of average pooling. To put it another way, the purpose of this pooling layer is to minimise the size of the picture, improve it, and use a side window filter of 3*3 with a stride of 2. By using stride 2, the image will automatically get straight and the image size will get reduced. The average pool layer receives an input of 55*55*96. The output of the pooling layer is thus 27*27*96 by the equ (1) after executing this pooling procedure.

The convolutional layer is once more the third layer. The input dimensions in this layer are 27*27*96. There are 256 filters in this convolutional layer, each measuring 5 by 5, with a stride of 1 and padding of 2. This image has an output size of 27*27*256 when the same equ(1) is applied. These channels number 256. ReLU activation is another function utilised here to ascertain a neural network's yes-or-no output.

The max pooling layer is the following layer, and its input size is determined by the output of the layer before it. And this layer's duty is the same as the second layer's. where the filter size equals 3×3 and the stride equals 2. Thus, the output will be decreased from $10 \times 10 \times 6$ to $13 \times 13 \times 2566$ by using Equation (1).

Once more, the convolutional layers make up the fifth, sixth, and seventh layers. The input size for the fifth convolutional layer is $13 \times 13 \times 384$. There are 384 filters, or neurons, with a size of 3×3 , a stride of 1, and padding of 1 in this convolutional layer. By applying the same equ(1) on this image, the output size is $13 \times 13 \times 384$. In the sixth layer the input size is $13 \times 13 \times 384$. Here also the number of neurons is the same as the previous layer, that's why the output is also the same. For the next convolutional layer the input size is $13 \times 13 \times 384$. In this layer the number of neurons is 256 and by applying the equ(1), the output is $13 \times 13 \times 256$. Here also relu activation is a function used to determine the output of a neural network like yes or no.

The max pooling layer is the following layer, and its input size is determined by the output of the layer before it. Additionally, this layer's job is to make the image smaller. where the filter size equals 3×3 and the stride equals 2. Thus, the output will be decreased from $6 \times 6 \times 256$ to $13 \times 13 \times 256$ by using Equation (1).

The following layer, which has 4096 feature mappings or neurons to complete the classification job, is a fully linked convolutional or dense layer. For this assignment, the multiplication operation is used to transform the output of the preceding layer into a 1-D vector image. Then the output of the fourth layer is $6 \times 6 \times 256$ which means 9216 pixels. Now, these 9216 pixels will be the input of the next layer which is a fully connected convolutional layer. In this fully connected layer there are 4096 neurons and each of size 1×1 . In this layer every neuron will process these pixels and then it will pass it to a next fully connected layer.

Having 4096 neurons, the eleventh layer is completely linked. Every function is the same as what was in the layer before. This layer's output is included in the subsequent final layer.

The output layer y , which is fully linked and has two or more potential values, is the last layer. Here, the sigmoid activation function determines which neurons fire and which do not.

A mathematical function with a distinctive "S"-shaped curve, or sigmoid curve, is called a sigmoid function. The sigmoid function's output falling between 0 and 1 is essentially the primary justification for its employment. As such, it is particularly useful for models whose output is a probability prediction. Since the output value's probability only occurs in the interval between 0 and 1, the sigmoid is the best option.

Working Procedure of a New Architecture

Machine learning techniques are becoming increasingly important in object recognition techniques. I can gather additional datasets, pick up more potent models, and employ better overfitting prevention strategies to boost their performance. Up until recently, there were not many datasets with labelled photographs. For this experiment, I used 3,000 images that I acquired from Kaggle.com. There is a brain tumour in fifteen hundred of the photographs I have, and the remaining fifteen hundred are just pictures of the brain without a tumour. As I noted above, datasets of large may be used to answer pretty successfully simple recognition tasks, especially when label-preserving changes are added to the dataset.

When we talk about the working procedure then we start from the libraries, which are needed to be import from the python libraries. Then we need to create our dataset as an image and load all the images from folders. Then resize the images as we need. Then label the images as '0' and '1'. Then we need to do partitions as training and testing dataset. Then validate the dataset as we need in the output layer. From 'tensorflow' library now we need to create models as we need. Then compile the model. After compilation we need to test our dataset using testing dataset. After these we need to check our results and accuracy then finally load a random image and need to check the result. These are the working procedure using CNN.

I employed 14 levels in this construction, as seen in Fig. 4 below. The network's input is the size of the image, and it is $255*255*3$. This network consists of eight fully connected convolutional layers, three max-pooling layers to shrink the picture's size, and three convolutional layers to extract features from the image. The last of these eight completely linked layers is the output layer, from which two possible outputs may be obtained: either the brain scans show a tumour or not [10].

Mathematical Calculation:

There is a formula to reduce the image’s size which is:

$$\frac{n + 2p - f}{s} + 1 * \frac{n + 2p - f}{s} + 1$$

.....equ(1)

Here, I use the same mathematical calculations or equations for finding the output of a particular layer, which is shown in the above equ(1). Where...

n = input size p = padding

f = number of filters s = strides

what is the output of any layer is always depends on this calculation.

Working Procedure

When we talk about the working procedure then we start from the libraries, which is need to be import from the python libraries. Then we need to create our dataset as an image and load all the images from folders. Then resize the images as we need. Then label the images as ‘0’ and ‘1’. Then we need to do partitions as training and testing dataset. Then validate the dataset as we need in the output layer. From ‘tensorflow’ library now we need to create models as we need. Then compile the model. After compilation we need to test our dataset using testing dataset. After these we need to check our results and accuracy then finally load a random image and need to check the result. These are the working procedure using CNN [11].

partitioning i





Fig13: Working Procedure of This Project

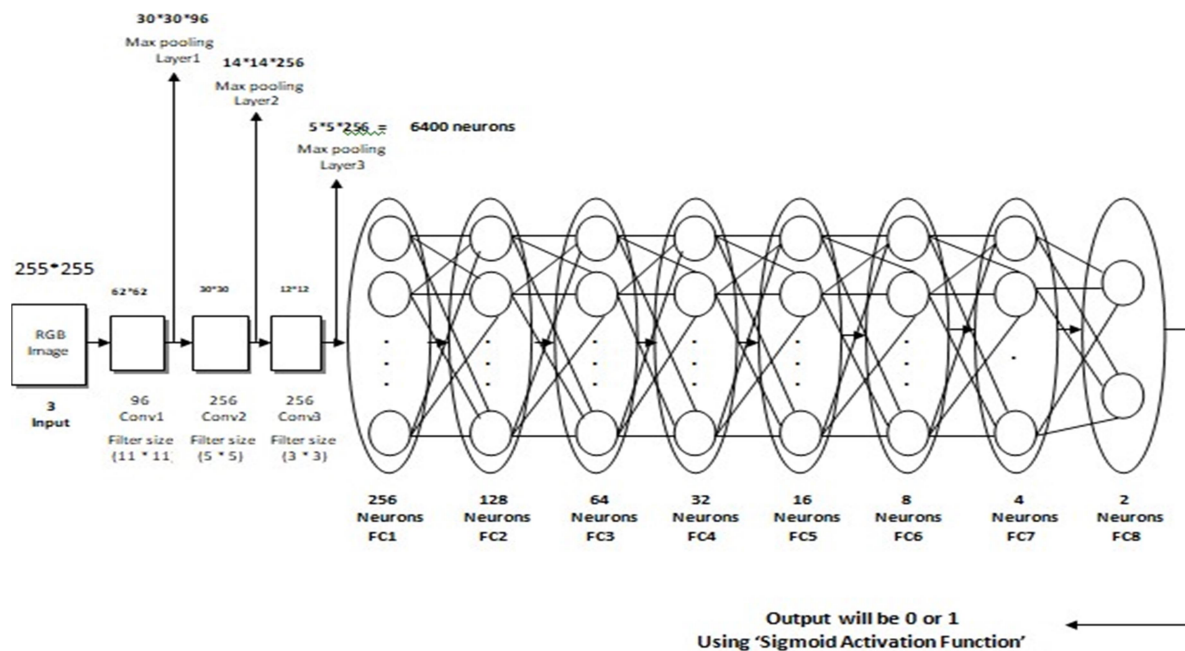


Fig5: Architecture of this network

Fig14: Structure of new Architecture

Architecture of this network:

This architecture has 14 layers which are shown in Fig14. The input size to this network is 255*255*3.

First Layer:

There are 96 filters in the first convolutional later, each with an 11*11 size and a 4 stride. The first convolutional layer uses the picture with dimensions of 255*255*3 as its input. The output of this layer

is then reduced to $55 \times 55 \times 96$ using equation (1), as seen below, after the convolutional process. The number of filters used in this layer determines the number of channels in every case. The output of a neural network, such as yes or no, is determined here using the Relu activation function.

Conv1: Convolutional Layer: Input size, $n = 255$

Filter size, $f = 11$ Padding, $p = 0$ Strides, $s = 4$ From equ(1)...

$$\frac{255 + 2 * 0 - 11}{4} + 1 * \frac{255 + 2 * 0 - 11}{4} + 1$$

= $62 * 62 * 96$ (where 96 is the number of channels or filters) so, this output $62 * 62 * 96$ is the input of next maxpooling layer.

Second Layer:

The next layer is the maxpooling pooling layer. The process of pooling data that determines the maximum value in each patch of each feature map is known as max-pooling pooling. As a consequence feature maps that are down-sampled or pooled and show the feature that is most prevalent in the patch—rather than the feature's average presence in the case of average pooling. In other words, the task of this pooling layer is to minimize the size of the image, enhance the image and the pooling layer using here a filter of side window of $4 * 4$ and stride is equal to 2. By using stride 2, the image will automatically get straight and the image size will get reduced. The input to the maxpooling layer is $62 * 62 * 96$. And then by applying this pooling operation, the output of the pooling layer is $30 * 30 * 96$ by the equ(1)...

Maxpooling Layer1:

Input size, $n = 62$ Filter size, $f = 4$ Padding, $p = 0$ Strides, $s = 2$

From equ(1)...

$$\frac{62 + 2 * 0 - 4}{2} + 1 * \frac{62 + 2 * 0 - 4}{2} + 1$$

= $30 * 30 * 96$ (where 96 is the number of channels or filters) So, this output $30 * 30 * 96$ is the input of next maxpooling layer.

Third Layer:

The third is again the convolutional layer. In this type of layer the input size is $30 * 30 * 96$. In this convolutional layer there are 256 filters having size of $6 * 6$, a stride of 1 and padding is 2. By applying the same equ(1) on this image, the output size is $28 * 28 * 256$. There are 256 such channels. Here also relu activation is a function used here...

Conv2: Convolutional Layer:

Input size, $n = 30$ Filter size, $f = 5$ Padding, $p = 2$ Strides, $s = 1$

From equ(1)...

$$\frac{30 + 2 * 2 - 5}{1} + 1 * \frac{30 + 2 * 2 - 5}{1} + 1$$

= $30 * 30 * 256$ (where 256 is the number of channels or filters) so, this output $30 * 30 * 256$ is the input of next maxpooling layer.

Fourth Layer:

After the second layer then the next layer is the maxpooling pooling based layer. The task of this pooling based layer is to minimize or reduce the size of image, enhance the image and the pooling layer



using here a filter of side window of $4 * 4$ and stride is equal to 2 and padding is 0. By using stride 2, the image will automatically get straight and the image size will get reduced. The input to the maxpooling layer is $30 * 30 * 256$. And then by applying this pooling operation, the output of the pooling layer is $14 * 14 * 256$ by the equ(1)...

Maxpooling Layer2:

Input size, $n = 30$ Filter size, $f = 4$ Padding, $p = 0$ Strides, $s = 2$

From equ(1)...

$$\frac{30 + 2 * 0 - 4}{2} + 1 * \frac{30 + 2 * 0 - 4}{2} + 1$$

= $14 * 14 * 256$ (where 256 is the number of channels or filters) so, this output $14 * 14 * 256$ is the input of next maxpooling layer.

Fifth Layer:

The input size for this layer is $14 * 14 * 256$. There are 256 filters with a size of $3 * 3$, a stride of 1, and padding of 2 in this convolutional layer. This image may be processed to an output size of $12 * 12 * 256$ by using the same equ(1). These feature maps, or channels, number 256. ReLU activation is another function that's employed here to assess if a neural network produces a yes or no answer...

Conv3: Convolutional Layer:

Input size, $n = 14$ Filter size, $f = 3$ Padding, $p = 0$ Strides, $s = 1$

From equ(1)...

$$\frac{14 + 2 * 0 - 3}{1} + 1 * \frac{14 + 2 * 0 - 3}{1} + 1$$

= $12 * 12 * 256$ (where 256 is the number of channels or filters) so, this output $12 * 12 * 256$ is the input of next maxpooling layer.



Sixth Layer:

After the second layer then the next layer is the maxpooling pooling based layer. The task of this pooling layer is to reduce the size of image, enhance the image and the pooling layer using here a filter of side window of $4 * 4$ and stride is equal to 2 and padding is 0. By using stride 2, the image will automatically get straight. The input to the maxpooling layer is $12 * 12 * 256$. And then by applying this pooling operation, the output of the pooling layer is $5 * 5 * 256$ by the equ(1)...

Maxpooling Layer3:

Input size, $n = 12$ Filter size, $f = 4$ Padding, $p = 0$ Strides, $s = 2$

From equ(1)...

$$\frac{12 + 2 * 0 - 4}{2} + 1 * \frac{12 + 2 * 0 - 4}{2} + 1$$

= $5 * 5 * 256$ (where 256 is the number of channels or filters)

so, this output $5 * 5 * 256$ is the input of next layer and the next layer all the layers are fully connected layer.

Seventh Layer:

Now the next layer is a fully connected convolutional or dense layer with 256 feature maps or neurons to do the classification task. For this task the output neurons of the previous layer is converted into a 1-D vector image by doing the multiplication operation. Then the output of the sixth layer is $6*6*256$ which means 6400 pixels. Now, these 6400 pixels will be the input of the next layer which is a fully connected convolutional layer. In this fully connected layer there are 256 neurons and each of size $1*1$. In this layer every neuron will process these pixels and then it will pass it to a next fully based connected layer.

Eighth Layer:

There are 128 neurons or feature maps in this densely linked or completely connected layer. This layer's input is the output of the layer before it. Every function is the same as what was in the layer before.

Ninth Layer:

This completely linked layer consists of 64 feature mappings or neurons. This layer's input is the output of the layer before it.

Tenth Layer:

This completely linked layer consists of 32 feature mappings or neurons. This layer's input is the output of the layer before it.

Eleventh Layer:

This completely linked layer consists of 16 feature mappings or neurons. This layer's input is the output of the layer before it.

Twelfth Layer:

This completely linked layer consists of 08 feature mappings or neurons. This layer's input is the output of the layer before it.

Thirteenth Layer:

This completely linked layer consists of 04 feature mappings or neurons. This layer's input is the output of the layer before it.

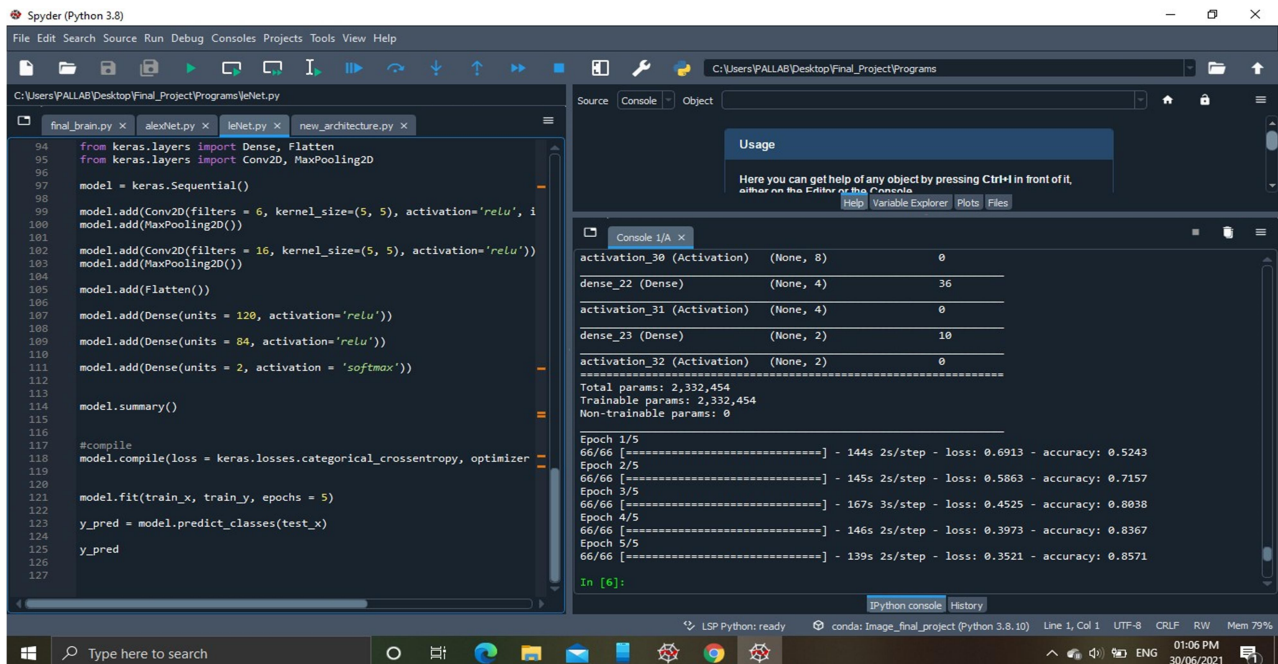
Fourteenth Layer or Final Layer:

This completely linked layer consists of two feature maps or neurons. This layer's input is the output of the layer before it. The output layer y is fully linked and has two potential values: "Have tumour" and "Don't have tumour." The numbers "Don't have a tumour" and "Have a tumour" are 0 and 1, respectively. Here, the sigmoid activation function determines which neurons fire and which do not.

A mathematical function with a typical "S"-shaped or sigmoid-type curve is called a sigmoid function. The sigmoid activation function is mostly used because its value falls between 0 and 1. As such, it is particularly useful for models whose output is a probability value that has to be predicted. Since everything with probability only exists in the interval between 0 and 1, the sigmoid is the best option.

Dataset and Data Validation

There are total of 3000 images which I have downloaded from kaggle.com. In that 3000 brain tumor images there are total of 2 folders [15]. In the first folder there are total of number 1500 images and all those images there is no tumor and in another folder there are total number of 1500 images and all those images there is a tumor. Label the images accordingly as assign '1' to images with tumors and '0' to images without tumors. Normalize pixel values to bring them within a similar range, typically between 0 and 1. This is the validation part and also one of the most important parts. Split the dataset into training and testing datasets. Common splits include 70% for training and 30% for testing. Here I ensured that the validation set contains a balanced representation of both classes. After training, “use measures like accuracy, precision, recall, and F1-score to assess the model's performance on the validation set”. In the



The screenshot shows the Spyder Python IDE interface. The left pane displays a Python script for a neural network architecture. The right pane shows the console output, including a summary of the model's parameters and training progress over 5 epochs.

```

94 from keras.layers import Dense, Flatten
95 from keras.layers import Conv2D, MaxPooling2D
96
97 model = keras.Sequential()
98
99 model.add(Conv2D(filters = 6, kernel_size=(5, 5), activation='relu', i
100 model.add(MaxPooling2D())
101
102 model.add(Conv2D(filters = 16, kernel_size=(5, 5), activation='relu'))
103 model.add(MaxPooling2D())
104
105 model.add(Flatten())
106
107 model.add(Dense(units = 120, activation='relu'))
108
109 model.add(Dense(units = 84, activation='relu'))
110
111 model.add(Dense(units = 2, activation = 'softmax'))
112
113
114 model.summary()
115
116
117 #compile
118 model.compile(loss = keras.losses.categorical_crossentropy, optimizer
119
120
121 model.fit(train_x, train_y, epochs = 5)
122
123 y_pred = model.predict_classes(test_x)
124
125 y_pred
126
127

```

```

Usage
Here you can get help of any object by pressing Ctrl+I in front of it,
either on the Editor or the Console.
Help Variable Explorer Plots Files

Console [1/A x]
activation_30 (Activation) (None, 8) 0
dense_22 (Dense) (None, 4) 36
activation_31 (Activation) (None, 4) 0
dense_23 (Dense) (None, 2) 10
activation_32 (Activation) (None, 2) 0
-----
Total params: 2,332,454
Trainable params: 2,332,454
Non-trainable params: 0
-----
Epoch 1/5
66/66 [=====] - 144s 2s/step - loss: 0.6913 - accuracy: 0.5243
Epoch 2/5
66/66 [=====] - 145s 2s/step - loss: 0.5863 - accuracy: 0.7157
Epoch 3/5
66/66 [=====] - 167s 3s/step - loss: 0.4525 - accuracy: 0.8038
Epoch 4/5
66/66 [=====] - 146s 2s/step - loss: 0.3973 - accuracy: 0.8367
Epoch 5/5
66/66 [=====] - 139s 2s/step - loss: 0.3521 - accuracy: 0.8571
In [6]:

```

below figure I have seen 85.71% accuracy which is based on the training dataset.

Fig15: Result of a New Architecture

Result Analysis and Output

Layers	Epochs	Parameters	Test Accuracy
5 Convolutional Layers, 3 Maxpooling Layers, 3 Dense Layers (AlexNet)	5	2, 47, 35, 106	49.90%
2 Convolutional Layers, 2 Maxpooling Layers, 3 Dense Alyers (LeNet)	5	61, 326	91.52%
3 Convolutional Layers, 3 Maxpooling Layers, 8 Dense Layers (New Architecture)	20	23, 32, 454	97.11%

Table1: Results of My Work with Test Accuracy

I utilised 16 neurons for the first convolutional layer and 32 neurons for the second convolutional layer in these models. Subsequently, the first and second density layers contain 8 and 2 neurons, respectively. Then there are two more layers one is MaxPooling layer and other is Flatten layer wich is shown in the above table (Table1). In the hidden layers or Convolutional layer and hidden Dense layers I use 'Relu' as activation function and in the final layer or output layer I use 'Sigmoid' as an activation function. Then fit the model and Compile. From the above table I have seen that I got 97.11% accuracy based on training dataset.



Output

LeNet Architecture:

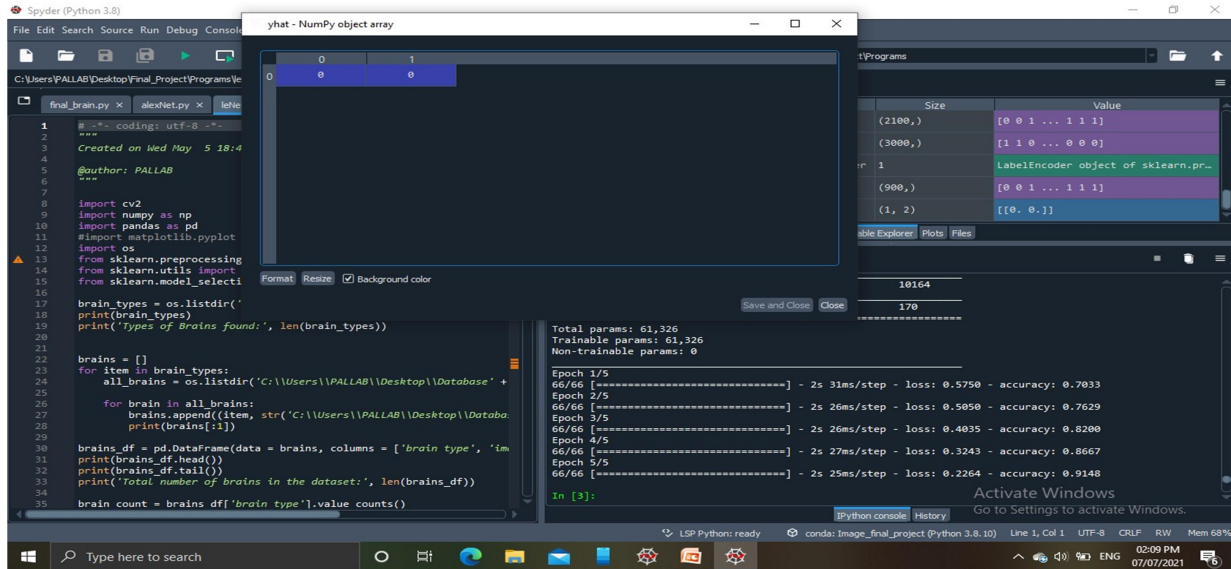


Fig16: Result of LeNet Architecture

AlexNet Architecture:

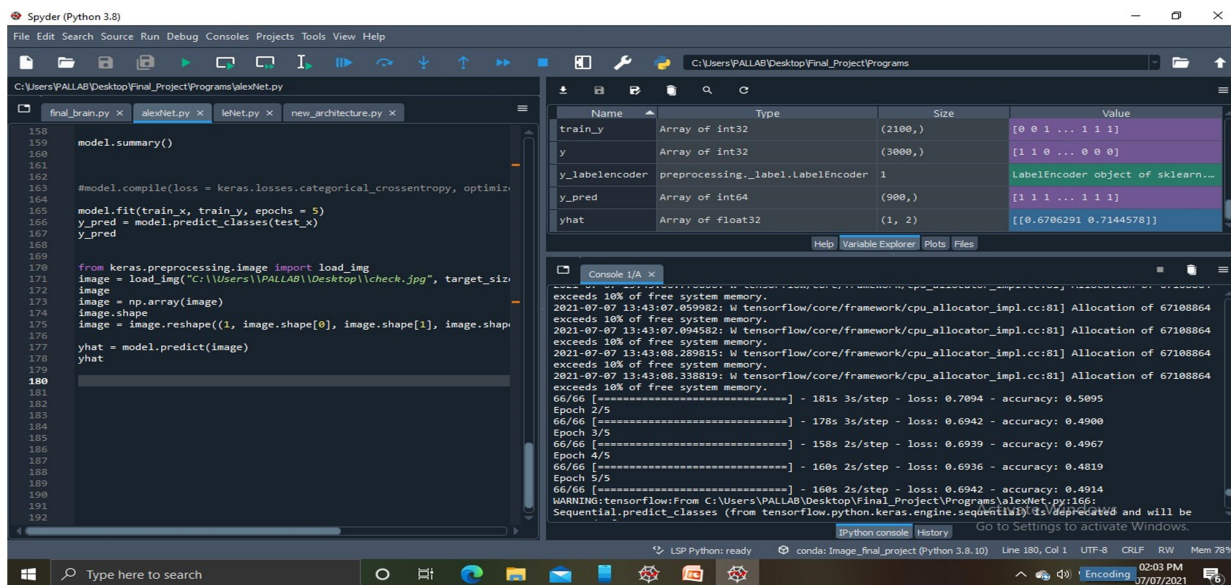


Fig17: Result of AlexNet Architecture



Result of my Work:

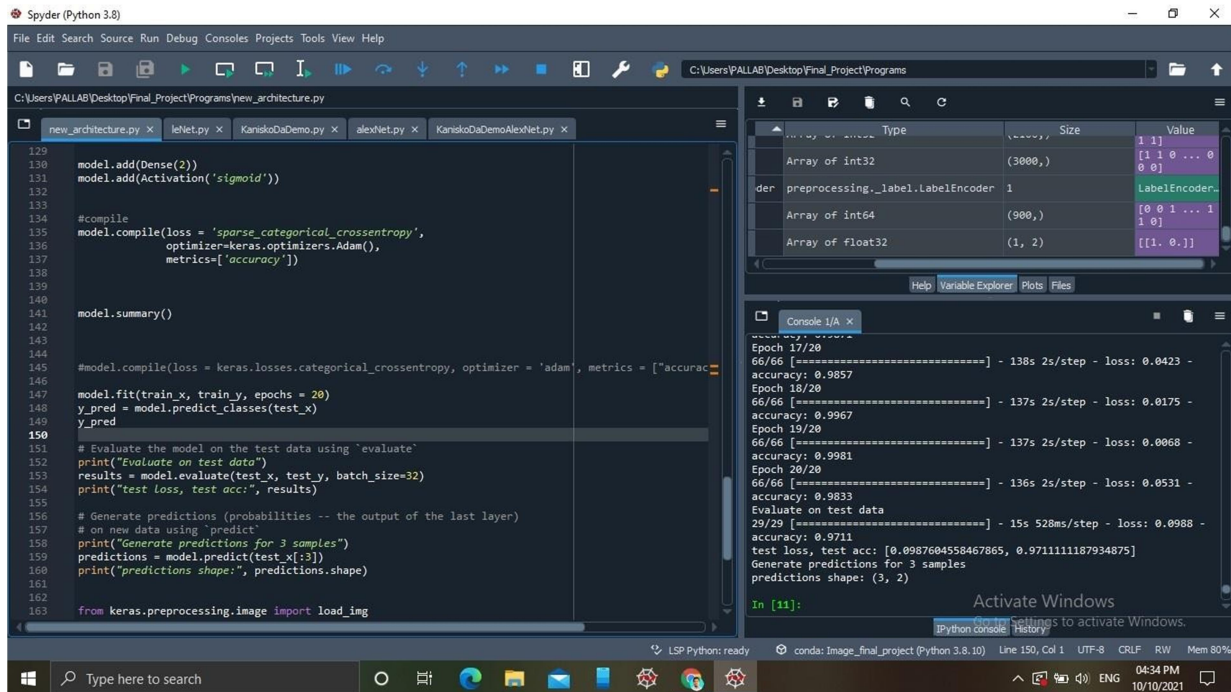


Fig18: Result of a New Architecture

End Notes

Here is the architecture I have seen in this report:

It has 14 layers with learnable parameters.

It consists of three convolutional layers that combine max-pooling layers.

Then it has 8 fully connected based layers.

The activation based function used in all the layers is 'Relu activation functions' except output layer.

I got 97.11% accuracy based on the new architecture.

The output layer uses the "Sigmoid activation function" as its activation function.

Some other examples are VGG-16 architecture, ResNet – 50 etc. All these are maintain similar kind of working procedure [12].

Future Scope

In future I like to work more on this new architecture to get more accuracy and along with this I want to try this same on other different medical images to classify or to detect there is any problem in that case or not. After this then I will go for segmentation also. We can use this same architecture in the other image classification processes like Skin Cancer Classification, Breast Cancer process and some others. Another one important work we can do based on this architecture is that if we can receive any regional cancer data as image then we can use this architecture and we can check that there is a tumor in that image or there is no tumor in that image. Moreover, based on that regional image cancer data we can differentiate the type of tumors like Benign Tumor, Premalignant Tumor and Malignant Tumor [13].

Spiking Neural Network is another very important topic which is used recently very effectively. A particular kind of artificial neural network that mimics the actions of real neurons is called a spike neural network (SNN). SNNs communicate by discrete, asynchronous "spikes" or bursts of activity, in contrast to standard artificial neural networks, which predominantly employ continuous-valued signals. This resembles the action potential-based neuronal communication found in the brain. We can use SNN model in the future to reduce the complexity, to reduce the energy and some other usefulness.

Conclusion

Deep Convolutional Neural Networks, recently introduced by deep learning techniques, show promise when used for medical picture processing. The entire range of medical image analysis, including computer-aided diagnosis, detection, segmentation, and classification, is covered by the appliance domain.

Image segmentation helps determine the relations between objects, in addition because the context of objects in an image [14].

Image Classification is also a critical process in computer vision. In recent times due to data availability this process get much more gain popularity day by day, which is good for our health industry. Here, in this report I gave an overview of the main training techniques for medical image classification, architecture of a new network, their advantages, and downsides. Here, in this report by doing classification task I got accuracy up to 97.11%.

References

1. NBTS, National Brain Tumor Society: Quick brain tumor facts, 2020. Available from: <https://braintumor.org/brain-tumor-information/brain-tumor-facts/>.
2. Cancer. Net, Brain Tumor: Statistics, 2020. Available from: <https://www.cancer.net/cancertypes/brain-tumor/statistics>.
3. <https://www.intechopen.com/books/medical-imaging-principles-and-applications/research-in-medical-imaging-using-image-processing-techniques>.
4. https://users.cs.cf.ac.uk/Dave.Marshall/Vision_lecture/node35.html.
5. https://users.cs.cf.ac.uk/Dave.Marshall/Vision_lecture/node34.html#:~:text=The%20basic%20idea%20of%20region,region%20satisfy%20some%20similarity%20constraint.
6. <https://ijcsmc.com/docs/papers/May2014/V3I5201499a84.pdf>
7. <https://www.investopedia.com/terms/n/neuralnetwork.asp#:~:text=A%20neural%20network%20is%20a,organic%20or%20artificial%20in%20nature>.
8. <https://www.digitalvidya.com/blog/types-of-neural-networks>
9. <https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529>
10. https://www.tensorflow.org/tutorials/images/data_augmentation.
11. <https://www.geeksforgeeks.org/activation-functions-neural-networks/>
12. <https://builtin.com/data-science/disadvantages-neural-networks>
13. Digital Image Processing by Rafael C. Gonzalez and Richard E. Woods.
14. Wang, W., Liang, D., Chen, Q., Iwamoto, Y., Han, X. H., Zhang, Q., ... & Chen, Y. W. (2020). Medical image classification using deep learning. Deep learning in healthcare: paradigms and applications, 33-51.
15. Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621.