Brain Tumor Prediction using CNN

Presented by: Raman Singh

Introduction

Image processing is a field that deals with the manipulation of images for various purposes. The two main purposes for using image processing are:

1. To improve the image: reducing "noise", highlighting details, sharpening, image retouching, creating effects, correcting colors and gray levels, etc..
2. To draw conclusions: measurements, defect detection, motion detection, shape recognition and object classification.

Also, image processing has many uses in research and industry; In medicine, military technologies, autonomous vehicles, cinema, smartphones, etc.

In our project we combine image processing with machine learning, in particular the use of a convolutional neural network.

Machine learning is the domain of businessin developmentalgorithmsintended to allowcomputerlearn from examples, and works in a variety of computational tasks where classical programming is not possibleOr rather complicated.

One of the methods for learning the counter is a neural network ANN (ARTIFICIAL NEURAL NETWORK) which imitates brain activity in the process of identifying objects. the input of theANN we obtained using a convolutional network CNN (CONVOLUTIONARY NEURAL NETWORK).

In our project we will use a convolutional network to differentiate between imagesNormal brain MRI and brain MRI images with a cancerous tumor.

We chose the method CNN for two main reasons:

1. CNN uses the filters we learned in the image processing course
2. The network knows how to teach itself fromdatasetand achieve good results at high speed.

Motivation

The medical industry is developing over the years, imaging technologies such asMRIfMRI CT USare being upgraded and streamlined but there is still a need for human intervention in the form of a technician/doctor who will decipher the imaging image. We believe that the future of medicine lies in the automation of tests and minimal human intervention for several reasons:

1. speed–A computer is capable of decoding an image at a speed that exceeds that of a human.
2. Accuracy test–Computers operate according to algorithms and are not affected by factors such as stress, fatigue, etc.

3. Accessibility and reduction in personnel–The decoder is a computer program that does not require reaching a physical location.

The software does not require salary, conditions, an office and does not take vacations and works 24/7.

In this way, we lower medical costs and increase the well-being of customers.

We wanted to do a project that would deal with machine learningML in combination with image processing, these fields are very innovative and in high demand in the industry, and their combination was a source of interest and a source of learning for us.

The purpose of the project

Classification of images Brain MRI with the help of a machine learning algorithm and a neural network.

Background

Brain tumor

"Brain tumor (in English:Brain tumor) is an abnormal and uncontrolled growth of cells found in the brain: neurons, glial cells, epithelial cells and more. Primary brain tumors are named according to the types of cells from which they arise. The tumor cells may be primary cancer cells or cells originating elsewhere in the human body, which have metastasized to the brain.

Tumors can be benign and are usually confined to a small area. They can be malignant and invasive (ie, spread to neighboring areas)." [9]

Mdata store -DATASET

In order to train a convolutional network, two large pools of images are requiredMRI:

1. Healthy brain image database
2. A brain image database with a tumor.

The identification accuracy of the network is approximately in logarithmic relation to the number of images in the database (Figure 1.1) and therefore the more samples there are in dataset. The system will have a greater variety of cases to learn from and will be able to detect growth better. Sometimes thousands and tens of thousands of photos are required to achieve good results. Another decisive factor is the quality of the images according to the GIGO principle– garbage in garbage out.

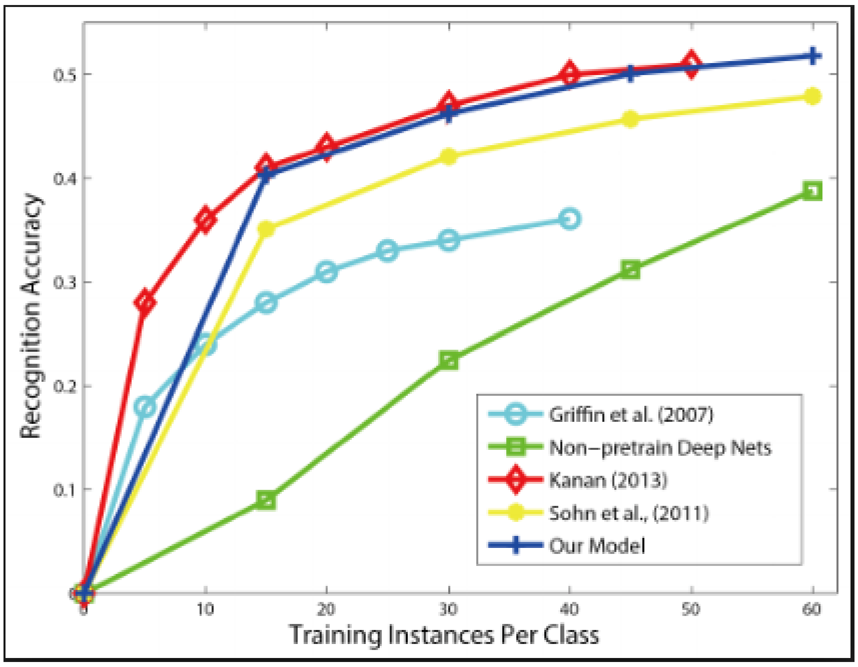


Figure 1.1 - Ratio between the amount of data and the percentage of identification success[4] -

Machine learning

There are many algorithms that are used to solve problems with the help of machine learning for continuous and discrete problems such as:

- Linear Regression
- Multiple Linear Regression
- Logistic Regression
- Random Forest
- KNN
- Kernel SVM
- Decision tree classification
- Gaussian RBF Kernel
- Naïve Bays theorem
- ANN
- CNN

CNN

In our project we chose to use– CNN The principle of network operation can be divided into 4 stages:

Step 1. Convolution

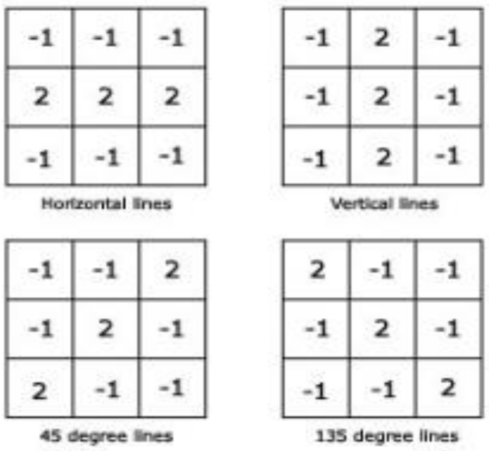
Step 2. Max Pooling

Step 3.Flattening

Step 4.Full connection

Step 1. Convolution

Define several filters mainly of the cutter type (Figure 2.1-2.2) to detect edges and make a two-dimensional convolution of the filter with the image (Figure 2.3–2.4).



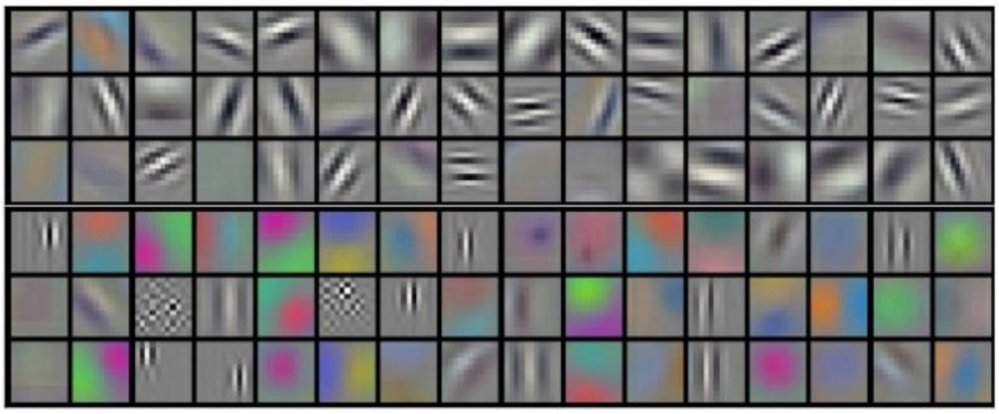
$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$$

Horizontal lines Vertical lines

$$\begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

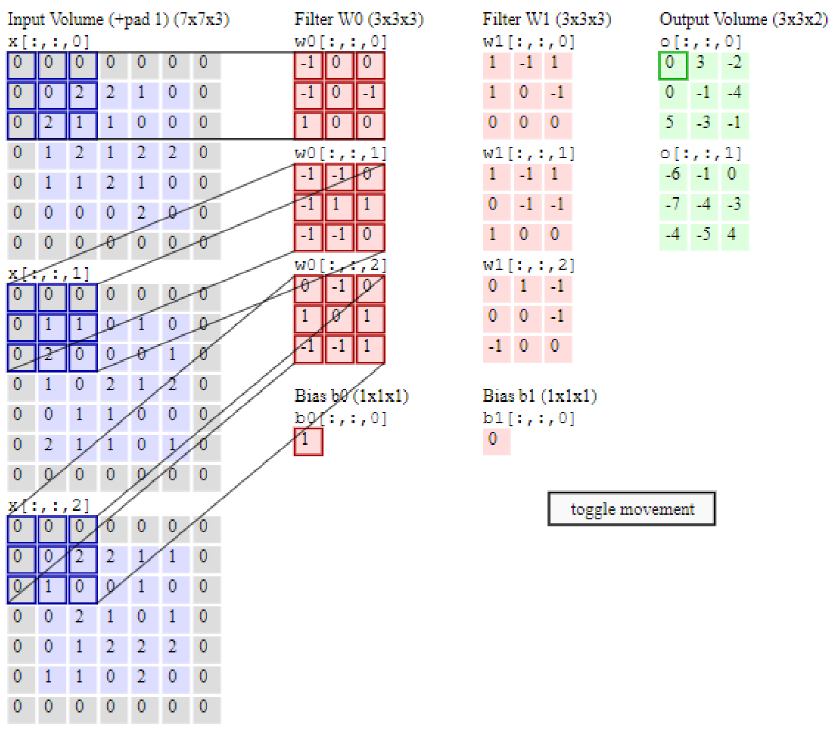
45 degree lines 135 degree lines

Figure 2.1 - Filters[2]



Example filters learned by Krizhevsky et al. Each of the 96 filters shown here is of size [11x11x3], and each one is shared by the

[2]**Figure 2.2**



x[:,:,0]

0	0	0	0	0	0	0
0	0	2	2	1	0	0
0	2	1	1	0	0	0
0	1	2	1	2	2	0
0	1	1	2	1	0	0
0	0	0	0	2	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	1	1	0	1	0	0
0	2	0	0	0	1	0
0	1	0	2	1	2	0
0	0	1	1	0	0	0
0	2	1	1	0	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	0	2	2	1	1	0
0	1	0	0	1	0	0
0	0	2	1	0	1	0
0	0	1	2	2	2	0
0	1	1	0	2	0	0
0	0	0	0	0	0	0

w0[:,:,0]

-1	0	0
-1	0	-1
1	0	0

w0[:,:,1]

-1	-1	0
-1	1	1
-1	-1	0

w0[:,:,2]

0	-1	0
1	0	1
-1	-1	1

b0[:,:,0]

1

w1[:,:,0]

1	-1	1
1	0	-1
0	0	0

w1[:,:,1]

1	-1	1
0	-1	-1
1	0	0

w1[:,:,2]

0	1	-1
0	0	-1
-1	0	0

b1[:,:,0]

0

o[:,:,0]

0	3	-2
0	-1	-4
5	-3	-1

o[:,:,1]

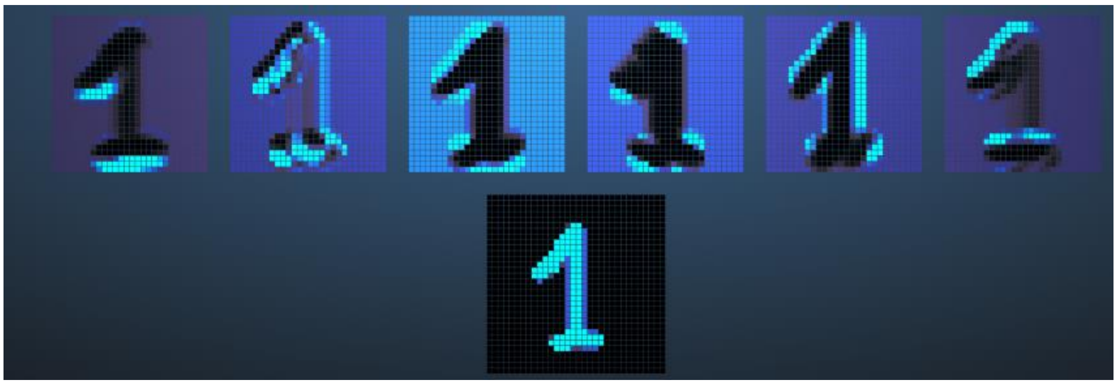
-6	-1	0
-7	-4	-3
-4	-5	4

Figure 2.3

At the end of this step for each existing filterfeature mapWith the unique characteristics of the filter, this can be seen with the following example (Figure 1.5):

The original image of the number 1 below, with 6 filters applied creating 6feature mapsAs you can see in the above pictures. It is possible to distinguish the cutters from the bottom from the top in diagonals, etc.

Figure 2.5[3]



Step 2. Max Pooling (Down Sampling)

There are several types of methodsPOOLING like max, min, mean to reduce the amount of irrelevant information and noise. We will use the MAX POOLING technique in which we will pass a 2x2 filter that will take the maximum value from every 4 pixels in the feature map, thus getting rid of 75% of the irrelevant information and noise, and preserving the main characteristics of the image (Figure 3.2, 3.1)

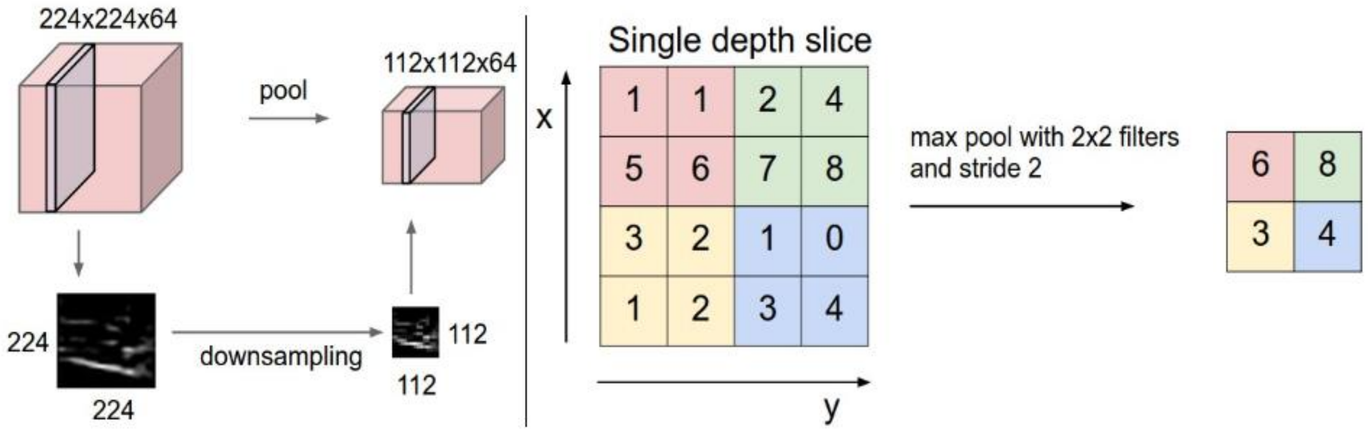


Figure 3.1

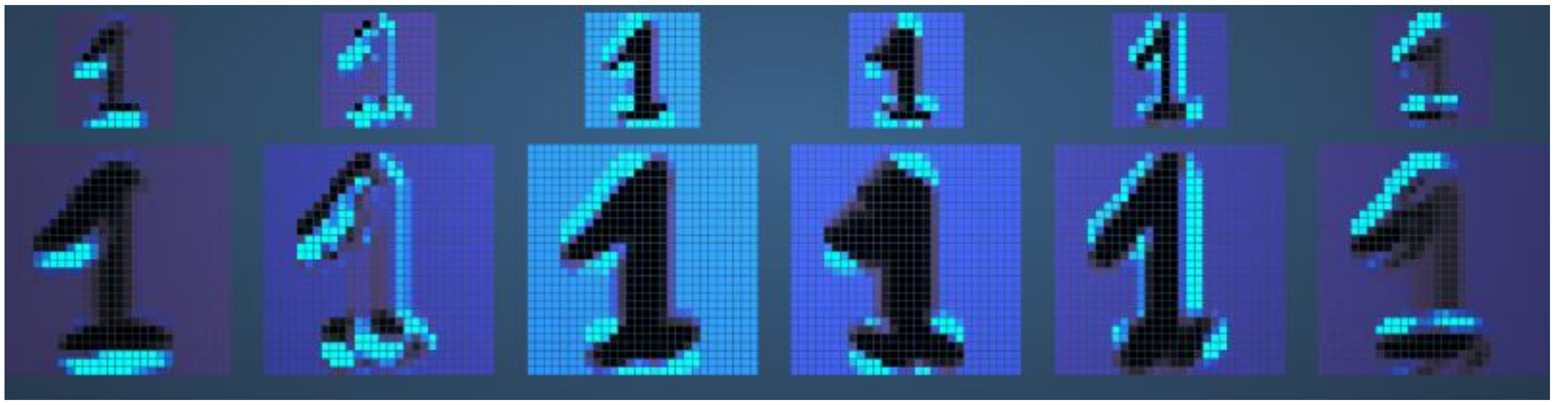


Figure 3.2[3]

Step 3.Flattening

At this stage we would like to flatten the matrices that represent thepooled feature mapsfor the form of a vector to be theinputto the neural network

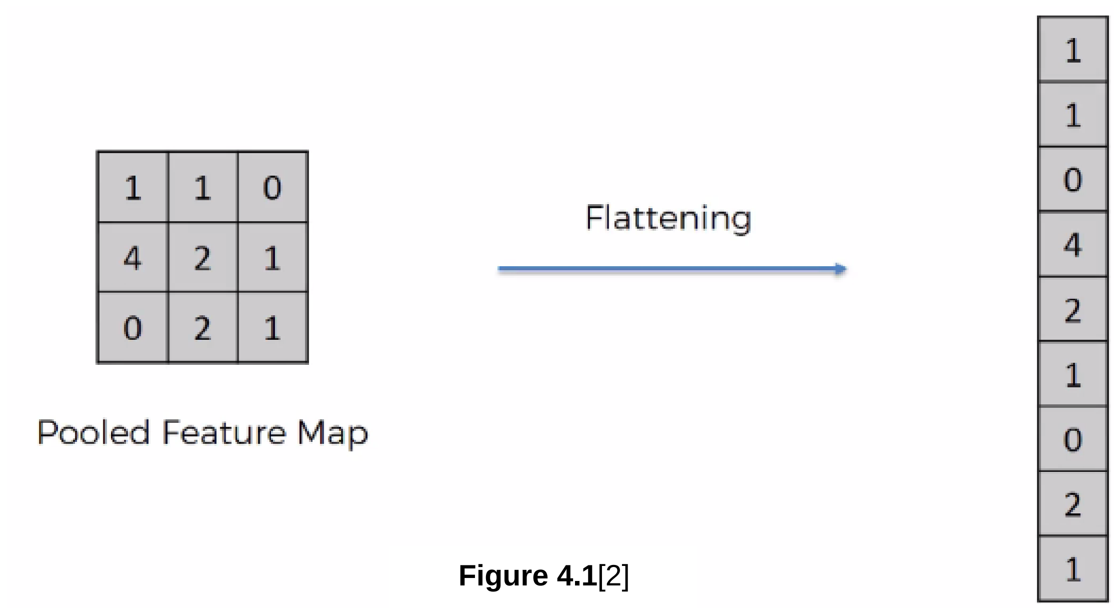


Figure 4.1[2]

Step 4. Full connection

We will use a neural network that imitates brain activity (Figure 5.1) Each neuron is surrounded by synapses (dendrites) which are receptors that transmit electrical signals from other neurons, according to the strength of the signals it receives the neuron decides whether to transmit the signal or not. If the neuron decides to transmit the signal, it will transmit it through the axon (axon) to the synapses of other neurons and thus an information transmission network is formed.

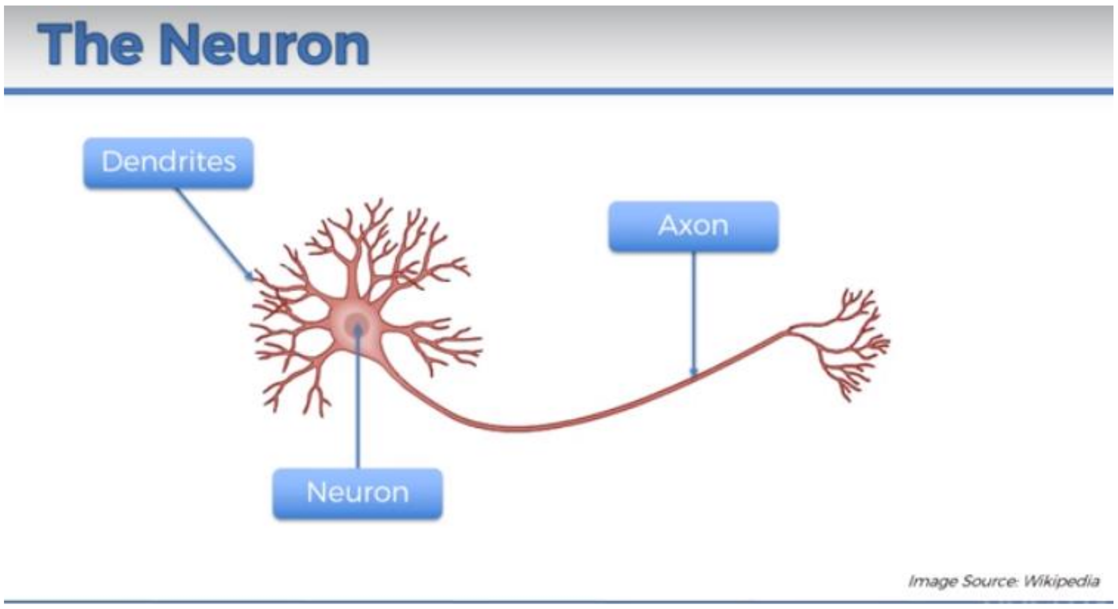


Figure 5.1 - Neuron[5]

We will enter the vectors from the previous section into a neural network (Figure 5.2) where each neuron is connected to all the inputs and performs the operation of multiplying its weights and a scheme of all the values. After that each neuron weighs the values and decides with the help of an activation function (activation function-Sigmoid function/alignment function ReLu) to transmit the information and with what intensity just like a neuron in the brain (Figure 5.3).

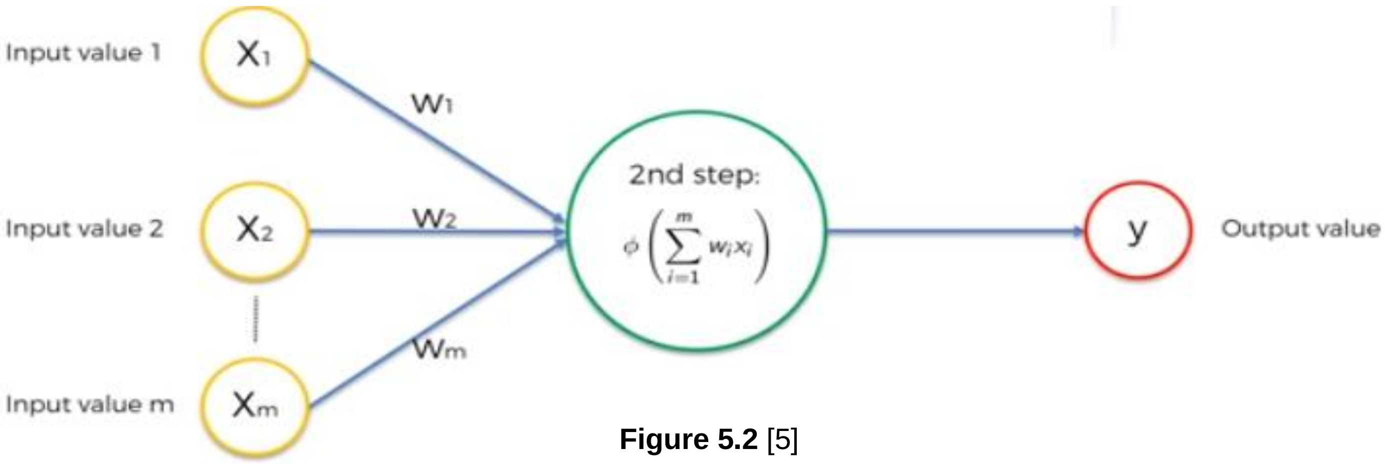


Figure 5.2 [5]

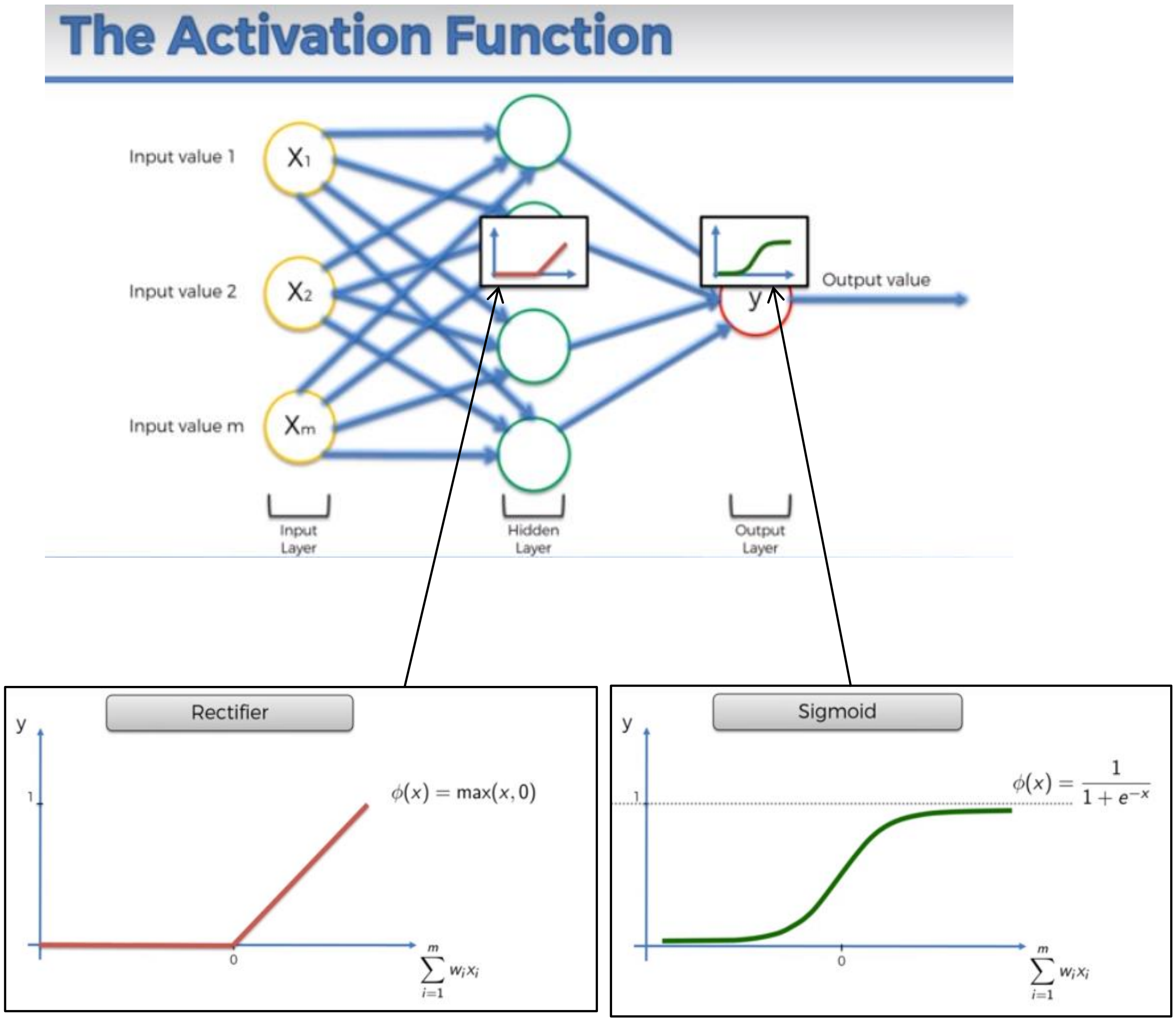
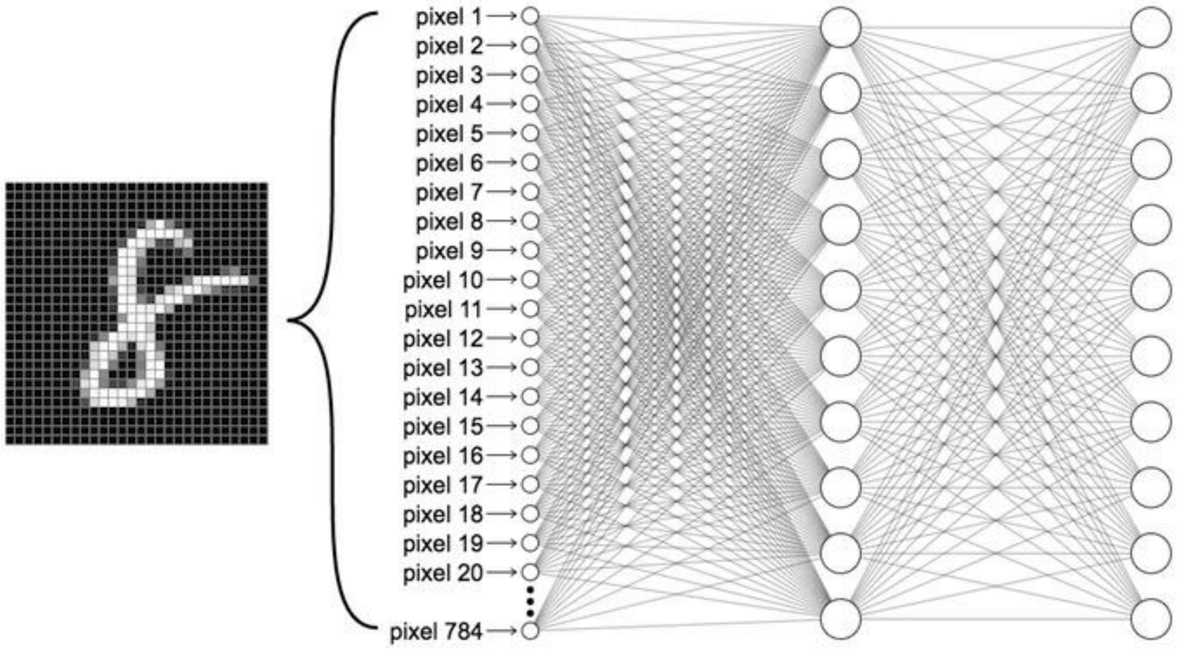


Figure 5.3 [5]

A network with a number of ports (digit identification 0-9) each digit is represented in an image/matrix 28X28 after vector flattening 784 inputs.



[6] **Figure 5.4**

The sigmoid function gives us the probability between 0 and 1 that the output is of a certain type, in our case a brain with a tumor or a healthy brain.

The learning of the network is done by minimizing our error function, an error function of typeMSE (mean square error) mean square error which is defined as follows:

$$J(\theta_n) = \frac{1}{n}\sum(y_i - \hat{y_i})^2$$

The difference between the network's prediction and the real value squared.

The direction of the weights in the synapses allows us to get a prediction as similar as possible to the true value and thus minimize the error function. The minimization algorithm is of the type

Stochastic gradient descent- Jump steps in the opposite direction to the gradient of the error function until you reach a minimum point where the weights of the network are optimal.

an equationGRADIENT DESCENT:

$$\theta_{n+1} = \theta_n - \alpha \frac{\partial}{\partial \theta_n} J(\theta_n)$$

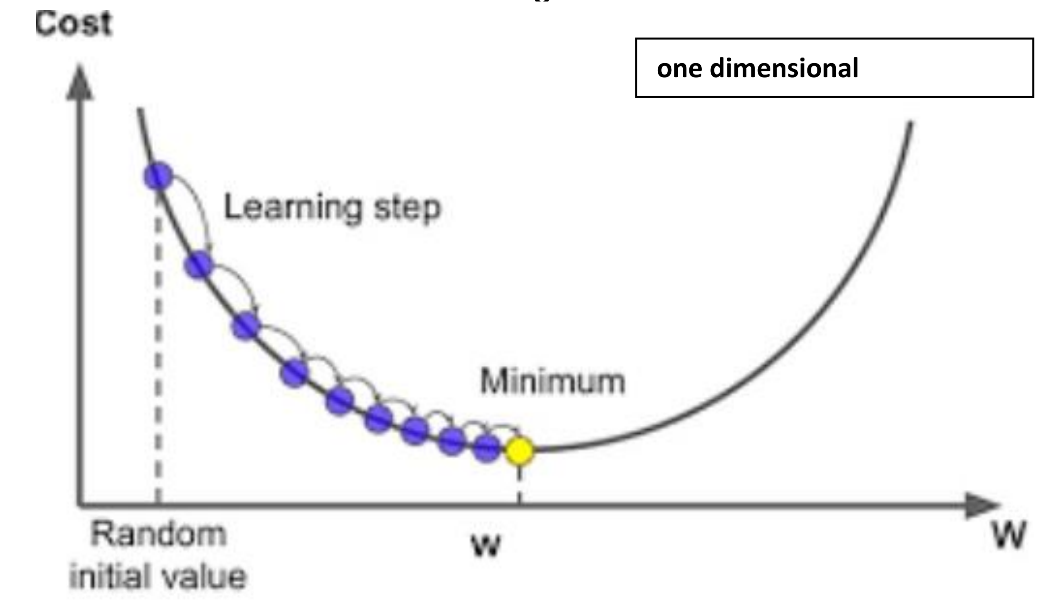
$$\theta \rightarrow parameter\ vector$$

$$J \rightarrow cost\ function$$

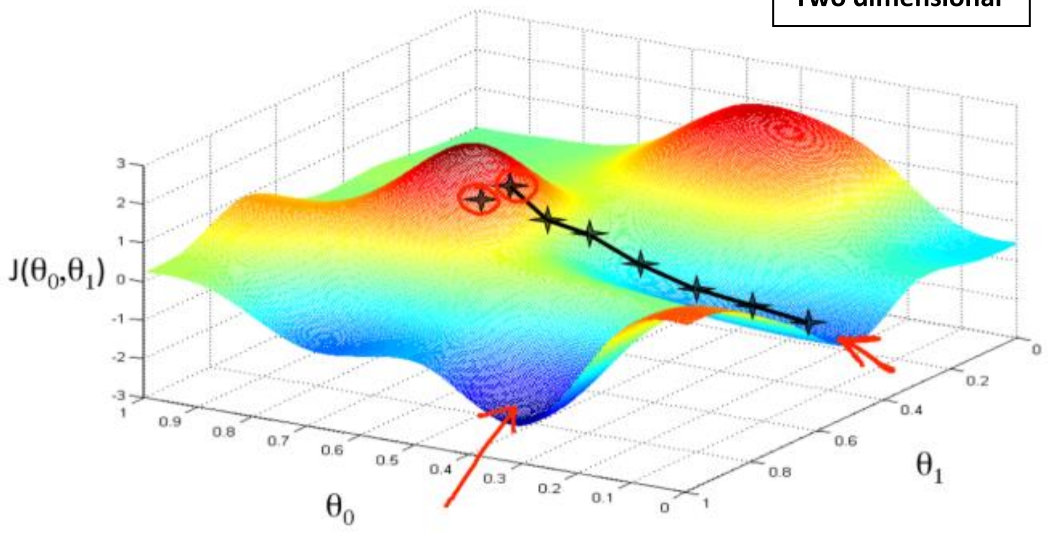
$$\alpha \rightarrow slope\ Parameter$$

Figure 5.5

one dimensional



Two dimensional



Action steps in the project

finding samplesMRI

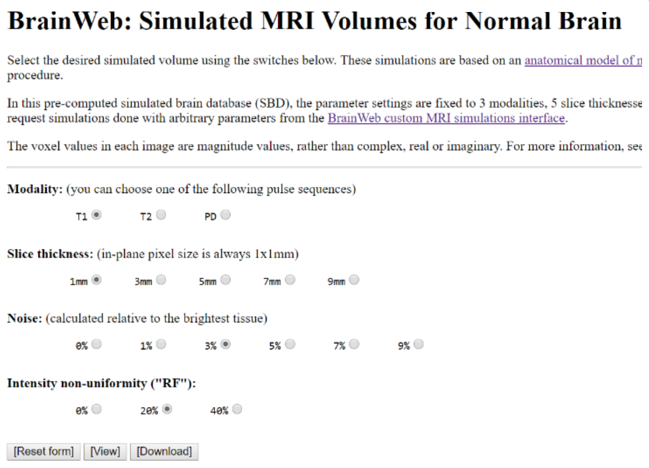
Search OtherStockbig enough ofPhotosMRI We decided to use a simulation that produces imagesMRI of a healthy brain and a patient with fictitious MS [1]. In this way, we could produce a large amount of images by introducing different parameters(Figure 6.1)and taking sections of the created images(Figure 2).

In this way, we producedDATASET including 100 MRI images of a healthy brain and 100 MRI images of a brain with multiple sclerosis (Fig.6.2).The repository shares:

- To 80%training set a database on which the network will be trained (160 images in total).

- To 20% test set an independent database on which the network will be tested (40 images in total).

Figure 6.1



BrainWeb: Simulated MRI Volumes for Normal Brain

Select the desired simulated volume using the switches below. These simulations are based on an anatomical model of n
procedure.

In this pre-computed simulated brain database (SBD), the parameter settings are fixed to 3 modalities, 5 slice thicknesse
request simulations done with arbitrary parameters from the BrainWeb custom MRI simulations interface.

The voxel values in each image are magnitude values, rather than complex, real or imaginary. For more information, see

Modality: (you can choose one of the following pulse sequences)

[✓ T1] T2 PD

Slice thickness: (in-plane pixel size is always 1x1mm)

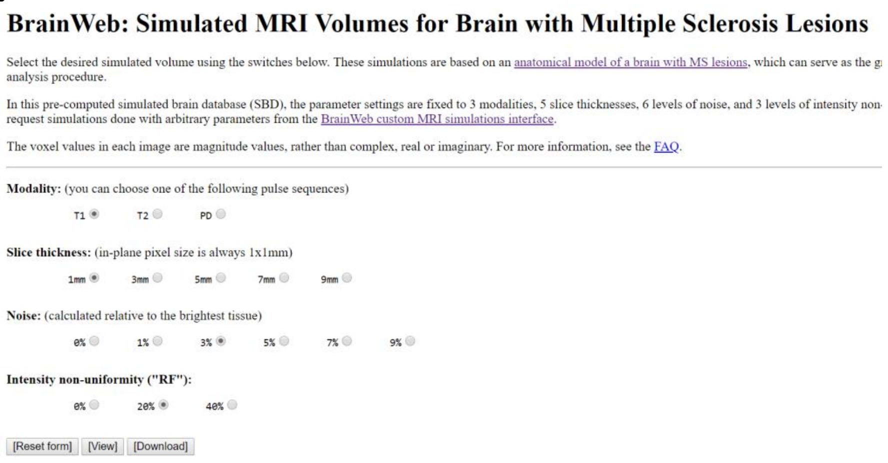
[✓ 1mm] 3mm 5mm 7mm 9mm

Noise: (calculated relative to the brightest tissue)

0% 1% [✓ 3%] 5% 7% 9%

Intensity non-uniformity ("RF"):

0% [✓ 20%] 40%



BrainWeb: Simulated MRI Volumes for Brain with Multiple Sclerosis Lesions

Select the desired simulated volume using the switches below. These simulations are based on an anatomical model of a brain with MS lesions, which can serve as the g
analysis procedure.

In this pre-computed simulated brain database (SBD), the parameter settings are fixed to 3 modalities, 5 slice thicknesses, 6 levels of noise, and 3 levels of intensity non-
request simulations done with arbitrary parameters from the BrainWeb custom MRI simulations interface.

The voxel values in each image are magnitude values, rather than complex, real or imaginary. For more information, see the FAQ.

Modality: (you can choose one of the following pulse sequences)

[✓ T1] T2 PD

Slice thickness: (in-plane pixel size is always 1x1mm)

[✓ 1mm] 3mm 5mm 7mm 9mm

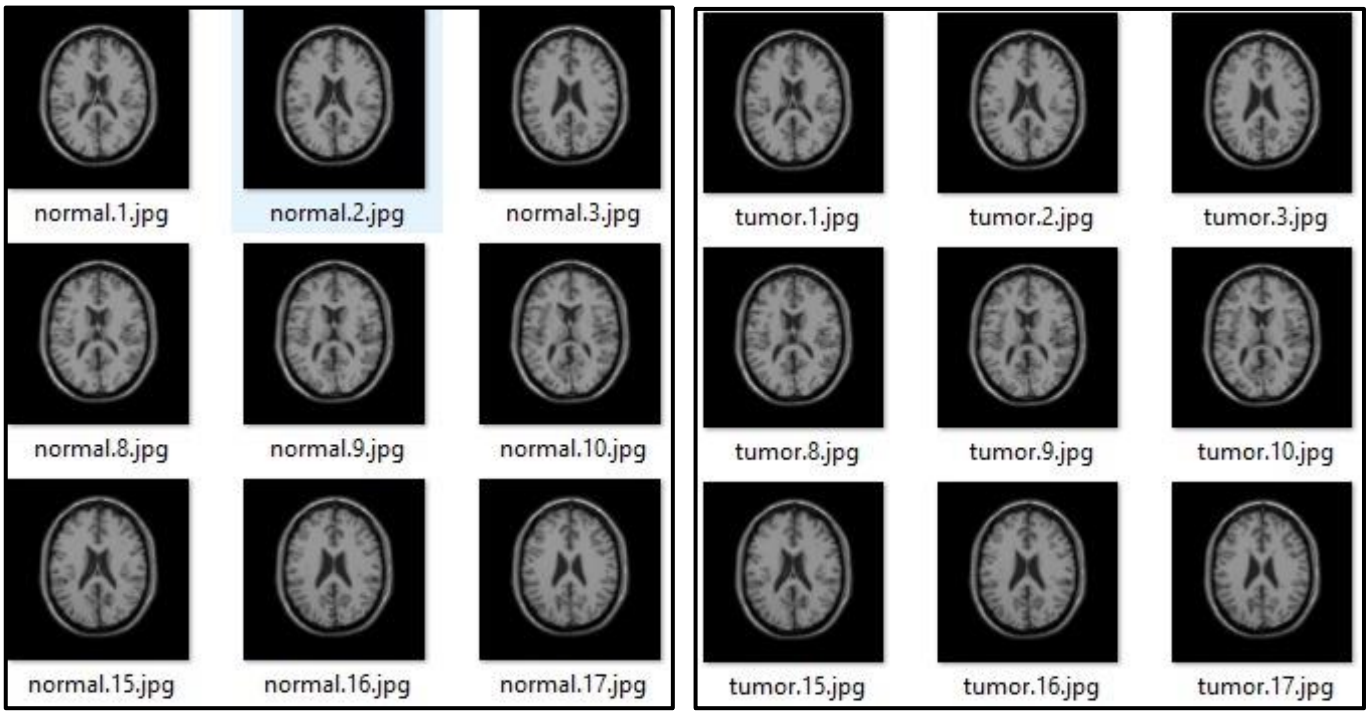
Noise: (calculated relative to the brightest tissue)

0% 1% [✓ 3%] 5% 7% 9%

Intensity non-uniformity ("RF"):

0% [✓ 20%] 40%

Figure 6.2:HEALTHY BRAIN AND MS BRAIN DATASET



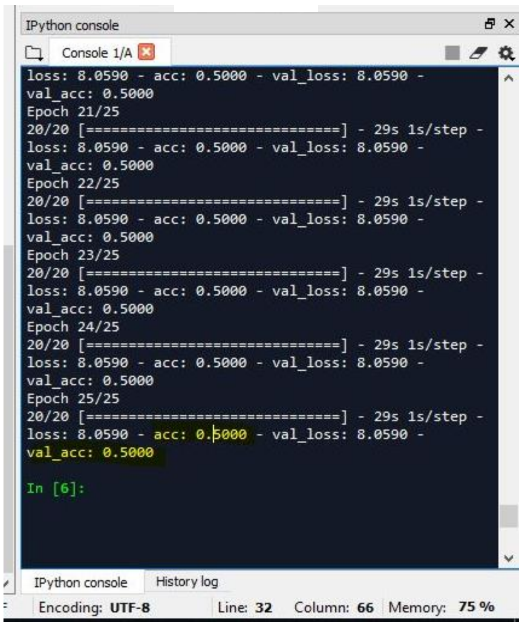
After running theDATASET In the code we wrote for the classification using the CNN method, we saw that the percentage was crazyGodV is 50%, because the network was not able to differentiate between the images (fig6.3). The reasons for this:

1. Since the database is created from a simulation, many of the parameters did not change between the images: image size, resolution, cross-section shape, angle, etc.
2. The tumors in the picturesMS were minimal in relation to the normal images and in relation to the resolutionsandA. The pictures.
3. Since we used the CNN we didn't have the option to manually adjust which filters to use.

Figure

acc - accuracy percentage for identification in TRAINING_SET.

val_acc - accuracy percentage for identification from the TEST_SET.



```
loss: 8.0590 - acc: 0.5000 - val_loss: 8.0590 -
val_acc: 0.5000
Epoch 21/25
20/20 [==============================] - 29s 1s/step -
loss: 8.0590 - acc: 0.5000 - val_loss: 8.0590 -
val_acc: 0.5000
Epoch 22/25
20/20 [==============================] - 29s 1s/step -
loss: 8.0590 - acc: 0.5000 - val_loss: 8.0590 -
val_acc: 0.5000
Epoch 23/25
20/20 [==============================] - 29s 1s/step -
loss: 8.0590 - acc: 0.5000 - val_loss: 8.0590 -
val_acc: 0.5000
Epoch 24/25
20/20 [==============================] - 29s 1s/step -
loss: 8.0590 - acc: 0.5000 - val_loss: 8.0590 -
val_acc: 0.5000
Epoch 25/25
20/20 [==============================] - 29s 1s/step -
loss: 8.0590 - acc: 0.5000 - val_loss: 8.0590 -
val_acc: 0.5000

In [6]:
```

Despite the advantage of a relatively large image database in taking images from a simulation, the results testified that it is better to use a smaller image repository but with images diverse MRI of patients/healthy.

In the article we relied on [7] use less than 100 images to train the network. We built our new database by collecting 150 images from the web, 85 of a healthy brain and 65 of a brain with a tumor (Figure 6.4). In the same way as the previous database, the new database is divided:

- 80%training set a database on which the network will be trained (120 images in total).

- 20% test set an independent database on which the network will be tested (30 images in total).

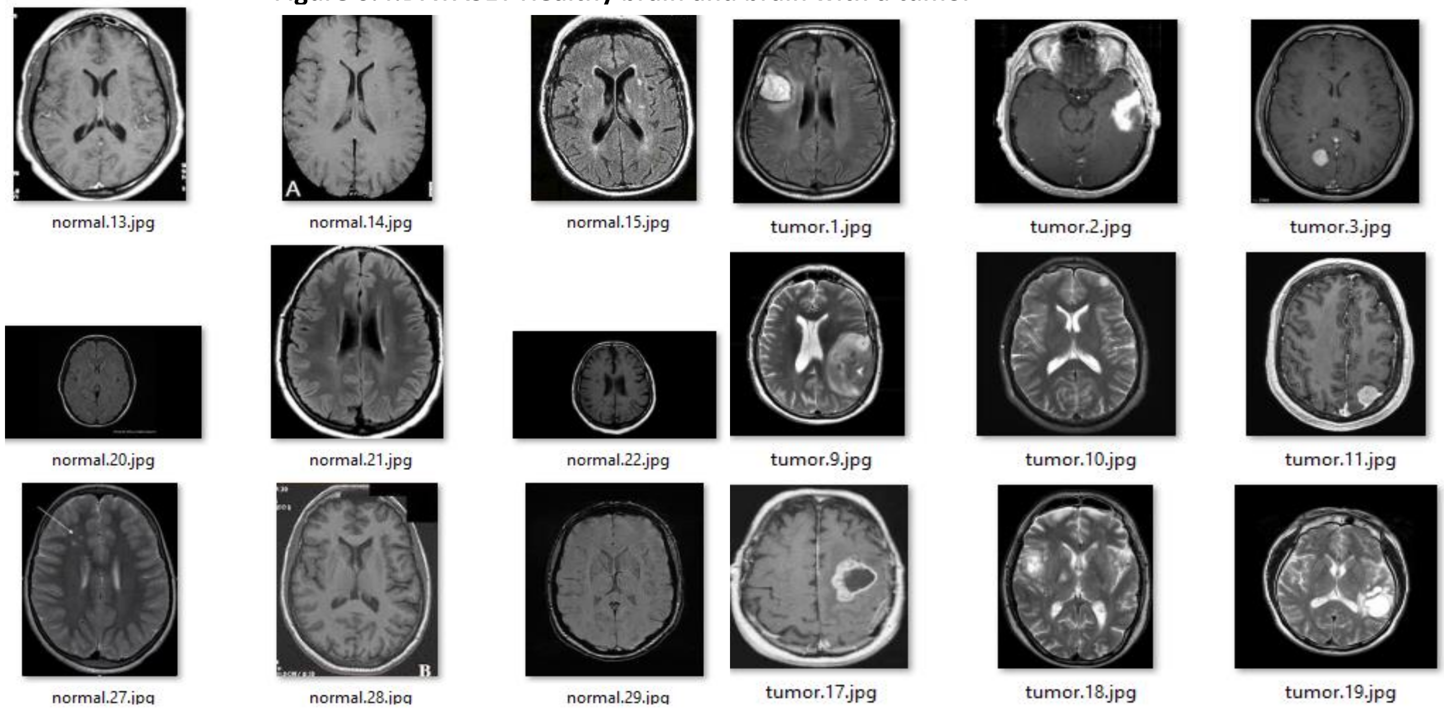


Figure 6.4:DATASET Healthy brain and brain with a tumor

Code explanation

Part One:
Importing the appropriate libraries, in the project we used the library deep learning on behalf of Keras which is based on open libraries from Google (TENSORFLOW) and the Indiana University, Bloomington.

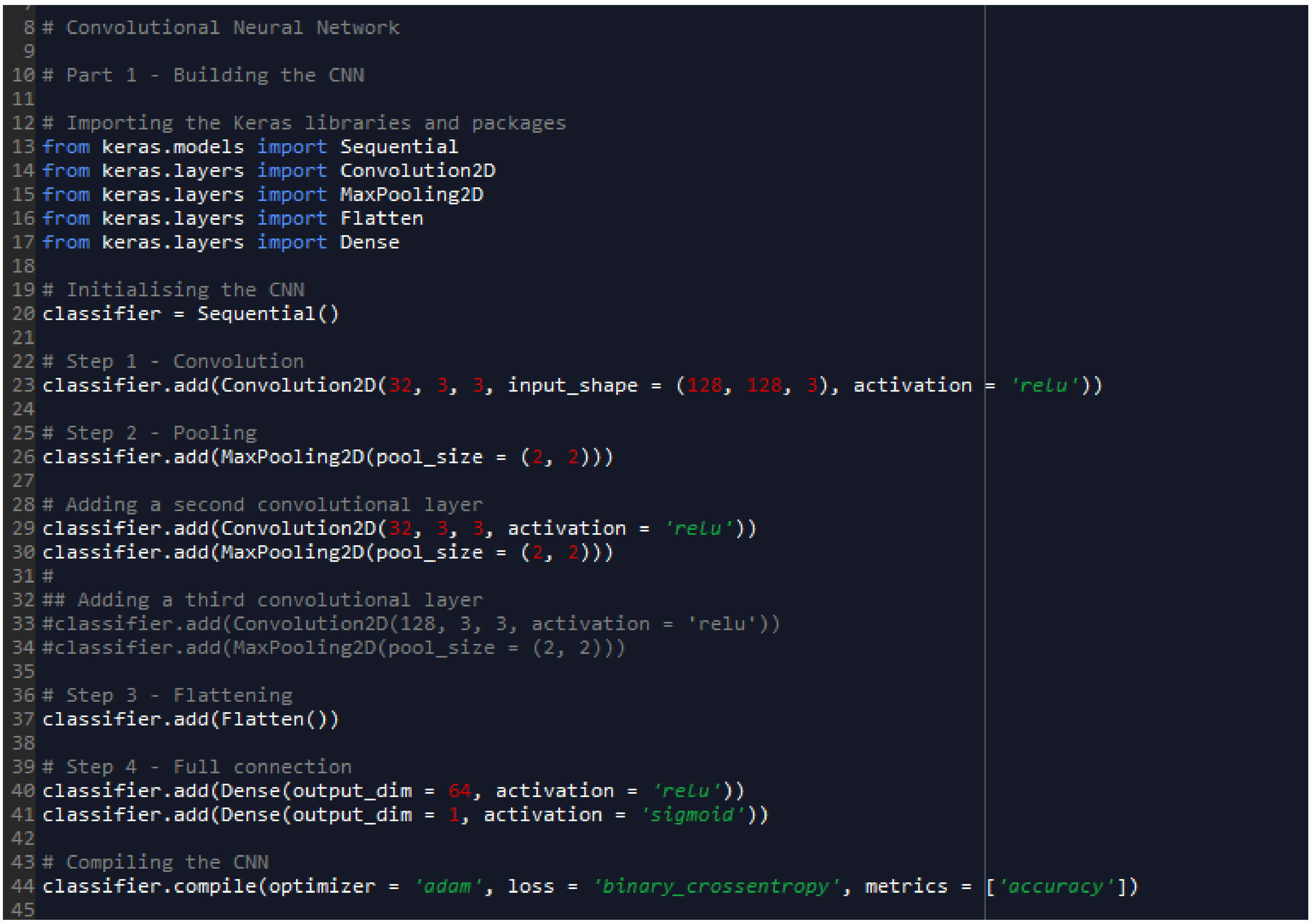
With the help of the libraries, we created an object for our CNN network called model.

Building the convolution network according to steps 1 to 4. In step 1 we defined 32 different filters of size 3X3 and the size of the MRI images to 128X128X3 (RGB) the R-type activation function ReLu.

In step 2 we added a layerMax PoolingSize 2X2

For optimization we added a convolution layer and pooled another one.

Step 3 is the flattening of the results and step 4 is connecting to a neural network, in our case we chose to use 64 neurons in the first layer (it is customary to use powers of 2).



```
# Convolutional Neural Network

# Part 1 - Building the CNN

# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense

# Initialising the CNN
classifier = Sequential()

# Step 1 - Convolution
classifier.add(Convolution2D(32, 3, 3, input_shape = (128, 128, 3), activation = 'relu'))

# Step 2 - Pooling
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Adding a second convolutional layer
classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))
#
## Adding a third convolutional layer
#classifier.add(Convolution2D(128, 3, 3, activation = 'relu'))
#classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Step 3 - Flattening
classifier.add(Flatten())

# Step 4 - Full connection
classifier.add(Dense(output_dim = 64, activation = 'relu'))
classifier.add(Dense(output_dim = 1, activation = 'sigmoid'))

# Compiling the CNN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

Second part

The adjustment of the data in terms of normalization range of values between 0-1 and resolution for our network 128X128 (The images come in a large resolution, on average 250X250)

Using the function horizontal flip which allows us to increase thedatasetours by adding mirror images of the existing images for example:

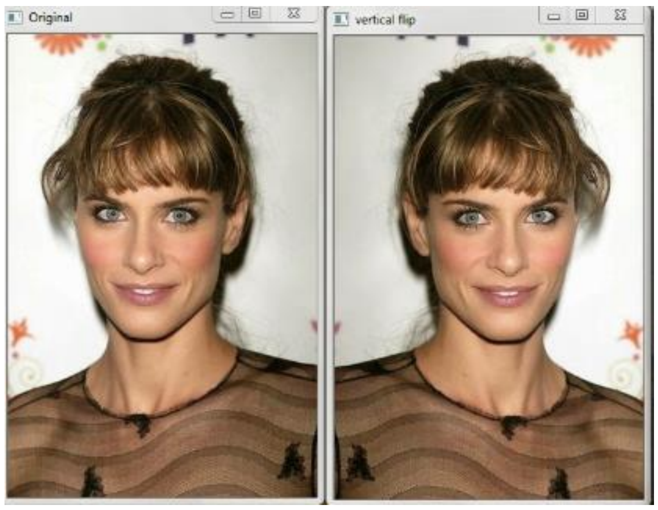
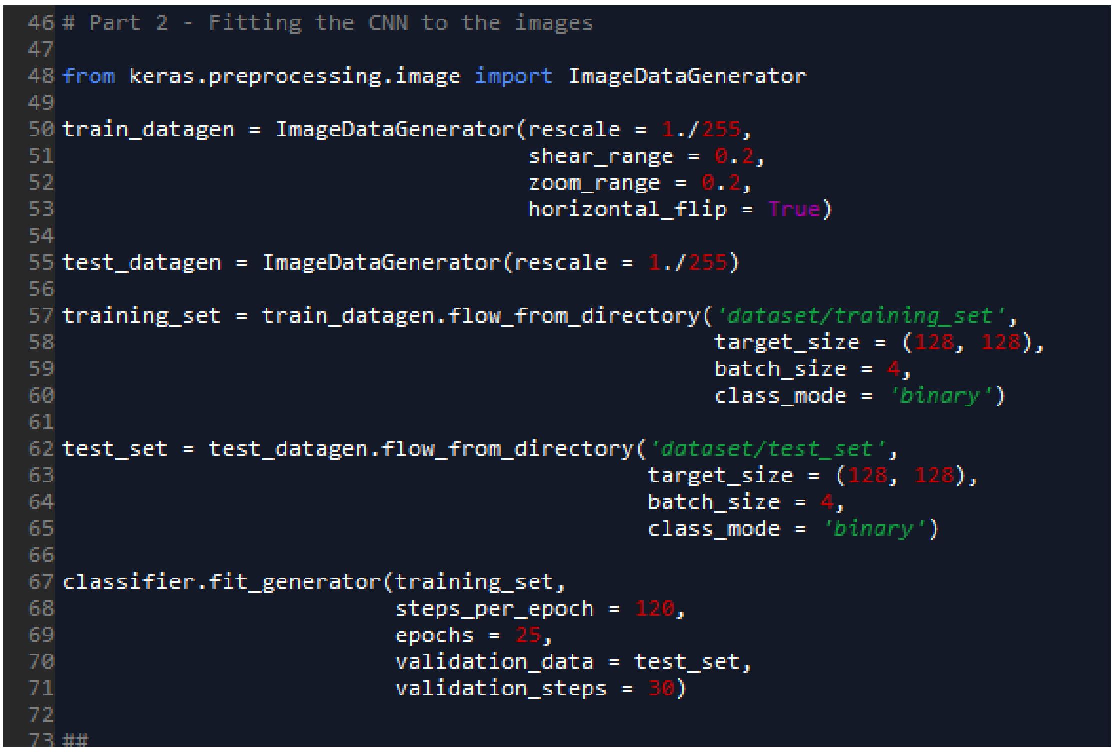
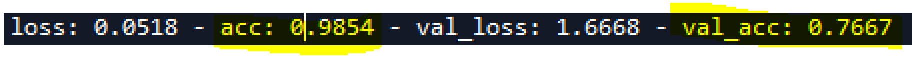


Figure 7.1 [8] - example of horizontal shift



```
# Part 2 - Fitting the CNN to the images

from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)

training_set = train_datagen.flow_from_directory('dataset/training_set',
                                                 target_size = (128, 128),
                                                 batch_size = 4,
                                                 class_mode = 'binary')

test_set = test_datagen.flow_from_directory('dataset/test_set',
                                            target_size = (128, 128),
                                            batch_size = 4,
                                            class_mode = 'binary')

classifier.fit_generator(training_set,
                         steps_per_epoch = 120,
                         epochs = 25,
                         validation_data = test_set,
                         validation_steps = 30)

##
```

Results and conclusions



```
loss: 0.0518 - acc: 0.9854 - val_loss: 1.6668 - val_acc: 0.7667
```

We reached an identification accuracy percentage of 98.5% on the training set.

and an identification accuracy percentage of 76.6% on thetest set

During the runs of the algorithm on the dataset Ours have changed many parameters like

- Image resolutions
- The number of convolutional networks
- the number of neurons
- number of epochs

Conclusions

- We found that using two convolutional layers with 32 filters each, with an image resolution of 128X128 and using 64 neurons, gives the best network for detection on our database.
- We discovered that it is possible to identify with relatively good accuracy images of images ofBrain MRI with tumor
- We discovered thatdataset(the image database) is a critical factor for the success of the model in identifying the tumors

We discovered that if we had more powerful hardware we could run heavier algorithms.

Future Work

- In terms of dataset the accuracy percentages can be improved a lot if there are more samples (thousands) and in a variety of sections and resolutions.
- In terms of hardware, the running times of the algorithm can be improved by overclockingGPU instead of CPU and use dedicated hardware for machine learning that reduces the runtime from days to minutes or hours.
- You can work with different machine learning algorithms and measure the efficiency of each one and find an optimal algorithm
- As was done in the article we relied on, it is possible to build an additional detection network to find out whether the detection is benign or malignant, for this purpose there is an additional network and bring another dataset that contains malignant and benign tumors.

Bibliography

1. (n.d.). Retrieved 08 20, 2018, from BrainWeb: Simulated Brain Database: http://brainweb.bic.mni.mcgill.ca/brainweb/

2. (n.d.). Retrieved 08 18, 2018, from Convolutional Neural Networks (CNNs / ConvNets): http://cs231n.github.io/convolutional-networks/

3. (n.d.). Retrieved 08 18, 2018, from http://scs.ryerson.ca/~aharley/vis/conv/flat.html

4. (n.d.). Retrieved 08 22, 2018, from ResearchGate: https://www.researchgate.net/figure/Mean-per-class-accuracy-of-Caltech-256-dataset-as-a-function-of-number-of-training_fig5_280158161

5. (n.d.). Retrieved 08 25, 2018, from udemy: https://www.udemy.com/machinelearning/learn/v4/overview

6. (n.d.). Retrieved 08 25, 2018, from looking inside neural nets: https://ml4a.github.io/ml4a/looking_inside_neural_nets/

7. (n.d.). Retrieved 08 15, 2018, from An algorithm for detecting brain tumors in MRI images: https://ieeexplore.ieee.org/document/5674887/

8. (n.d.). Retrieved 08 26, 2018, from http://enthusiaststudent.blogspot.com/2015/01/horizontal-and-vertical-flip-using.html

9. (n.d.). Retrieved 08 25, 2018, from https://he.wikipedia.org/wiki/%D7%92%D7%99%D7%93%D7%95%D7%9C_%D7%9E%D7%95%D7%97%D7%99