Cerebri Tumor Deprehensio

Brain Tumor Detection with Convolutional Neural Networks

ABSTRACT

This report is prepared to give information about the Detection of Brain Tumor from Brain X-Ray Images. Developed as a final project for Digital Image Processing Course. It is expected to develop a neural network model that is capable of performing medical diagnoses, in this instance, the diagnosis of Brain Tumor.

Within the scope of this project, a rapidly advanced convolutional neural network is developed that diagnoses Brain Tumor as accurately as ever.

This report gives detailed information about all the modules in the project. First, presenting an overview of the problem definition, project description, and aim of the project. Then, materials and methods are described that are at the core of the project.

THE PRESENTATION

https://youtu.be/XOV22TDWCHw

1. Introduction

This document is arranged to give technical and practical knowledge about the project. In this part, specific and definitive information is given about the problem definition, the aim of the project, and the project description which contains the subject and details of the project. On the other hand, the main topic of the problem, Brain Tumor, is illuminated in detail.

1.1. Project Description

This project contains an artificial neural network system and a convolutional neural network system, that are combined together to detect Brain Tumor from Brain X-Ray Images.

1.2. Why Deep Learning is Widely Used in Medical Diagnoses

The branch of science used by machines to make decisions and predict by imitating the work of the human brain is called Artificial Intelligence. It has significant success when used in problems that are difficult to solve even for manpower. Deep Learning, a sub-branch of artificial intelligence, provides solutions to many difficult problems such as image and voice recognition by learning from examples in the given data. Especially, the Convolutional Neural Networks perform the best results in image classification tasks.

Deep Learning is being used in many fields for so long and has solid potential to be used in the early diagnosis of diseases. Since Brain Tumor is common and deadly, it has catastrophic effects on health. Thus, advanced technological studies are carried out in medicine to reduce this danger and increase patient lifetime. Since the early diagnosis of the disease is an important factor in the treatment, these technological developments used in medicine are mostly *focused on early diagnosis*.

The use of deep learning technologies in the diagnosis of Brain Tumor from X-Ray Images has extremely radical advantages in terms of speed and accuracy because the success and speed of deep learning applications trained with the right data set is higher than humans in the early diagnosis of the disease.

1.3. Problem Definition

Brain Tumor is a mass or growth of abnormal cells in a person's brain. It has different types with different danger levels, and some types of these tumors are benign. In Medicine, it is known that early diagnosis reduces deaths. Because most of the deaths from this disease are due to late treatments and misdiagnosis. The main symptoms of this disease are headaches that are gradual and severe, nausea, vomiting, vision problems, feeling tired, personality changes, and hearing problems. Since these symptoms can be seen in many diseases, it is difficult to diagnose a patient with Brain Tumor as short as possible.

Misdiagnosis and mistreatment always result in one way, which is death. Using Artificial Intelligence applications in the diagnosis of Brain Tumor is more reliable, accurate, and faster than human

predictions because humans are more prone to mistakes than algorithms. Additionally, much slower to make decisions.

1.4. The Aim of the Project

The aim of this project is to train a neural network with thousands of Brain X-Ray images of people having Brain Tumor and train a computer to learn what Brain Tumor is and how to detect it.

The various benefits of high accuracy, speed, and efficiency of Artificial Intelligence can be utilized in the diagnosis of Brain Tumor. Consider the scenario, where it takes more than 20 years to train a human doctor, it takes a couple of hours to train an Artificial Intelligence model. Therefore, training and using a deep learning model in the diagnosis of the disease saves a great deal of time. Additionally, in recent studies, it is found out that a well-trained Artificial Intelligence model can diagnose disorders accurately in a shorter time and at a higher rate than doctors. This shows that the accuracy of the machines is more reliable.

2. Brain Tumor in Detail

Brain Tumor Overview

A brain tumor is a mass or growth of abnormal cells in your brain.

Many different types of brain tumors exist. Some brain tumors are noncancerous (benign), and some brain tumors are cancerous (malignant). Brain tumors can begin in your brain (primary brain tumors), or cancer can begin in other parts of your body and spread to your brain as secondary (metastatic) brain tumors.

How quickly a brain tumor grows can vary greatly. The growth rate, as well as the location of a brain tumor, determines how it will affect the function of your nervous system.

Brain tumor treatment options depend on the type of brain tumor you have, as well as its size and location.

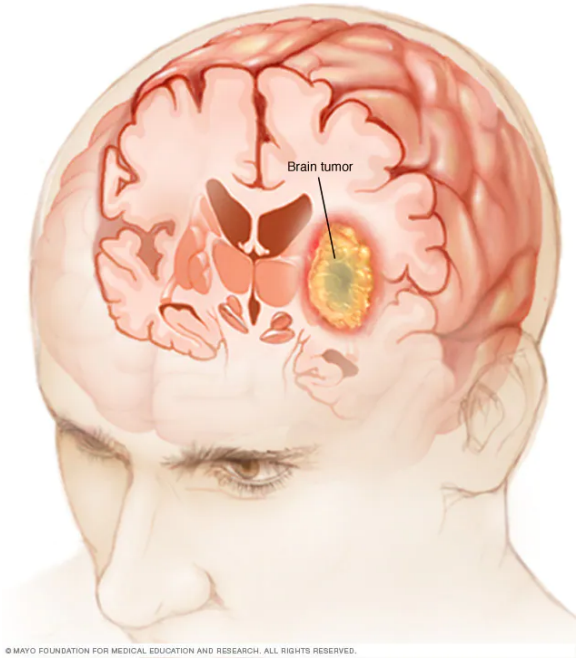


Figure 1 illustrates Brain Tumor on Brain.

Symptoms

The signs and symptoms of a brain tumor vary greatly and depend on the brain tumor's size, location, and rate of growth.

General signs and symptoms caused by brain tumors may include:

- New onset or change in the pattern of headaches
- Headaches that gradually become more frequent and more severe
- Unexplained nausea or vomiting
- Vision problems, such as blurred vision, double vision or loss of peripheral vision
- Gradual loss of sensation or movement in an arm or a leg
- Difficulty with balance
- Speech difficulties
- Feeling very tired
- Confusion in everyday matters
- Difficulty making decisions
- Inability to follow simple commands
- Personality or behavior changes
- Seizures, especially in someone who doesn't have a history of seizures
- Hearing problems

Brain Tumors That Begin in the Brain

Primary brain tumors originate in the brain itself or in tissues close to it, such as in the brain-covering membranes (meninges), cranial nerves, pituitary gland, or pineal gland.

Primary brain tumors begin when normal cells develop changes (mutations) in their DNA. A cell's DNA contains the instructions that tell a cell what to do. The mutations tell the cells to grow and divide rapidly and to continue living when healthy cells would die. The result is a mass of abnormal cells, which forms a tumor.

How Common Brain Tumors Are and Whether They are Dangerous

In the United States, brain and nervous system tumors affect about 30 adults out of 100,000. Brain tumors are dangerous because they can put pressure on healthy parts of the brain or spread into those areas. Some brain tumors can also be cancerous or become cancerous. They can cause problems if they block the flow of fluid around the brain, which can lead to an increase in pressure inside the skull. Some types of tumors can spread through the spinal fluid to distant areas of the brain or the spine.

3. Materials

This section focuses on platforms, libraries, software development environment and programming language which are used in this project.

3.1. Development Platform

Artificial Intelligence researchers have a couple of platform options while developing an AI program. These platform options are development on a local machine and development on cloud platforms.

In this project, Kaggle is chosen to be the platform of development, and there is a logic lying behind this decision. The main reason why Kaggle is the development platform of this project is the limitations of local machines, be they memory capacity or storage capacity.

Another option would be development on Cloud platforms. These platforms vary from Amazon Web Services to Google Cloud and Kaggle is among cloud Jupyter notebook providers. Even though Amazon Web Services' and Google Cloud's services are faster than Kaggle's Notebook services, Kaggle has a unique advantage that other cloud notebook providers do not have, which is the fact that the data is already on Kaggle, and Kaggle allows us to interact and import data in seconds. No need to download data or copy data from Kaggle to another provider. This feature of Kaggle has been fascinating and labor-saving in development. That is another reason why Kaggle has become the development platform.

3.2. Code Structure

The code structure of CTD (basically a Convolutional Neural Network project) is as follows:

Loading the Dataset: Training a neural network is matrix operations, and in order to perform those operations on data, the data must be imported and loaded on memory.

Preprocessing the Data: Data, in general, are not clean, and not in the form that is trainable. That is why some cautions and preprocessing must be done on data to clean it and make it trainable.

Data Augmentation: The neural network must not memorize the dataset. Wherever the neural network is fed with an image for prediction, the fed image is generally similar to the actual train data but the only difference is small variations. That is why data must be augmented, new data must be created for training.

Building the CNN Model: The architecture of the CNN is described, computed and built in this step.

Optimizations on Learning Rate: Configurations for optimizing the dynamic learning rate in this neural network

Training the Neural Network: In this step, the actual training process takes place. It makes more than 20 minutes on a low-powered machine. The result will a trained neural networks with computed weights and biases.

Saving the Model: The trained neural network might be used in the future with a Web Application, which will be served via HTTP. That is why the model must be saved on the storage for later use.

Evaluation: Neural networks can be used in critical areas such as Healthcare. That is why evaluation of the neural network is significant. There are a couple of factors for evaluation and testing.

These are the basic steps of a general structure of Cerebri Tumor Deprehensio.

3.3. Programming Languages

In the domain of Artificial Intelligence, generally, two main languages shine, which are Python and R programming languages respectively. Even though technically, Artificial Intelligence is just mathematical algorithms and those are irrelevant to the implementation, in other words, programming languages.

However, there is a reason why R or Python is massively popular in AI. As known, Artificial Intelligence, specifically Neural Networks, can be abstracted as gigantic matrix operations. One must consider billions of matrix operations. That is the main reason that AI Frameworks such as Tensorflow, Keras, and PyTorch use C and C++ internally since C and C++ are known for their unprecedented performance. But, these fancy overperforming languages have a non-negligible problem, that is they are extremely technical and have a huge roadblock people, especially for those who are non-technical, such as doctors or researchers with non-CS degrees. That is the same reason why Keras and Tensorflow are written in low-level languages such as C and C++ and **give users an**

interface with Python. Users interact with AI Frameworks with Python, which has skyrocketed Python's popularity.

In this project, a couple of programming languages are considered for development. Those are JavaScript, Java, C++, and Python, and at the end of the day, Python has been chosen as the development language. The main reason that Python is chosen over C++ is that Python development takes approximately ten times shorter than development with C++ and C. In the process of choosing the right development language, Javascript was also an option. Javascript is known for its development abilities, and its popularity in Front-End and Back-End development. That is why Google has published TensorflowJS, which is Javascript wrapping of Tensorflow platform. Python is chosen over Javascript, simply because Python has way more resources on the internet.

In conclusion, Python has been the best language for the development, in terms of its popularity, availability and ease.

3.4. Packages and Libraries

In Artificial Intelligence projects, there's been a huge need for third-party packages and libraries. Actually, the whole neural network development depends on it. That is why for years, the AI community published a variety of packages.
In Cerebri Tumor Deprehensio, a couple of libraries and packages is needed as well.

The main areas where a third-party code is needed are the following:

- Training the Neural Network
- Loading the Dataset in RAM
- Matrix Manipulations

In this project, the packages below are vastly used, and the reason of usage is explained as well.

Tensorflow
Tensorflow is not actually a library, rather it is an Artificial Intelligence platform that contains other frameworks and libraries for AI. Tensorflow allows developers and researchers high-power parallel computations and a simple development framework.

Keras

Keras is known for its elegant programming interface for Artificial Neural Networks. In reality, Keras behaves as an interface for Tensorflow internal training functionalities. Additionally, Keras provides a variety of tools, ranging from Learning Rate Annealers, Optimizers to Activation Functions.

In addition to the benefits listed above, Keras has a functionality to train not only Artifical Neural Networks but also Convolutional Neural Networks and Recurrent Neural Networks. Since this project is about Convolutional Neural Networks, Keras is a good choice.

The competition of Keras was PyTorch. Keras is chosen over PyTorch. The main reason is that PyTorch is unnecessarily complex for training a neural network whereas Keras is both simple and elegant, opposite to PyTorch.

MatPlotLib (Mat Plot Library)

Mat Plot Library is all about visualizations. In Artifical Intelligence Development, visualization of different kinds are vastly needed by researchers. Examples for visualizations are Line Plots, Histograms, Scatter Plots, 3D Plots, Image Plots, Contour Plots, Scatter Plots, Polar Plots and Line Plots.

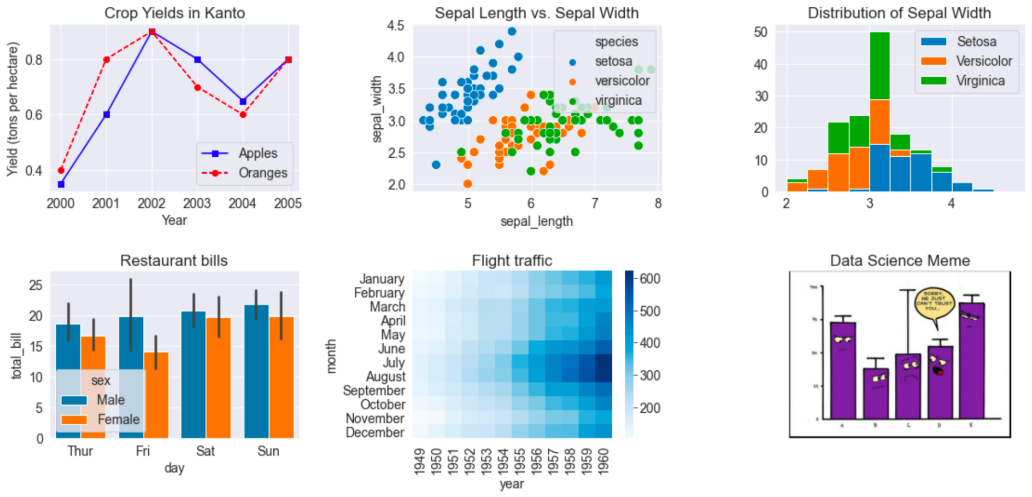


Figure 2 illustrates different kinds of visualizations that can be done with Mat Plot Library.

Mat Plot Library plays an important role as well. It is vastly used in **Drawing Accuracy Plot Over Epochs** and **Drawing Loss Plot Over Epochs**.

4. Methods

In this part of the report, the technical methods and procedures which are used in the project will be discussed in detail.

4.1. Theory of Artificial Neural Networks

4.1.1. What is an Artificial Neural Network?

It is a computing system unit to perform tasks like classification and prediction which consists of interconnected nodes called neurons and is modeled with inspiration from the human brain. ANN is one of the main algorithms like CNN, and RNN that forms Deep Learning. A basic Artificial Neural Network structure is named 2-layered structure that includes an input layer, hidden layer, and output layer. It is called 2-layered because the input layer is ignored when calculating the number of the layers.

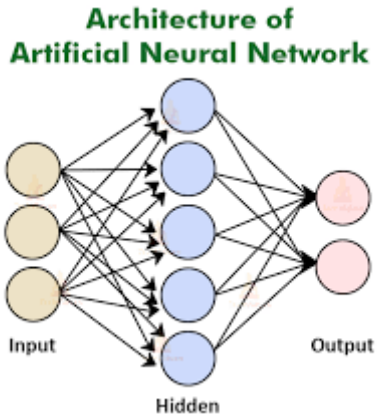


Figure 3. The basic 2-layered ANN structure

4.1.2. Computation Graph of ANN

An Artificial Neural Network has some parameters to process computations. The connections between two consecutive neurons have weights. During the learning process, weights are updated to reach less error rate and correct results. In the beginning, weights can be intuitively chosen at random. On the other hand, bias is a constant variable that helps to shift the activation function for getting a successful learning outcome. The activation function is a mathematical equation used to increase the nonlinearity of the output of a neuron and to calculate the result of it. The common types of activation functions used in neural networks are Sigmoid, Tanh, Step, Softmax, and ReLU. While sigmoid function is generally used for binary classification, softmax is used for multi-class classification.

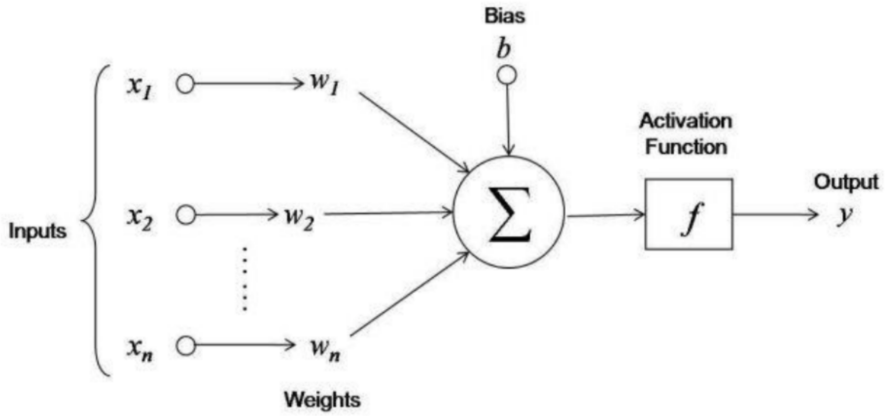


Figure 4. Visualization of ANN as Mathematical Expression

4.1.3. How Does ANN Work?

Initializing Values of Weights and Bias

Initially, the weights can be selected randomly and can be close to zero such as 0.01 but should not be zero. Otherwise, gradient descent will be constant after multiple iterations. Therefore, it will behave like linear but non-linear behavior should be used to make the neural network like in a human brain structure. In addition, bias can be any constant like 0 or 1.

Forward Propagation

It is the process of computation from the input layer to the output layer. Input data is given to the hidden layer for calculations. The weights entering each hidden node are multiplied by the input value which is denoted by x. After that, bias is added and the last equation is calculated in the activation function as an output of the layer. Second, the weights entering to output node are multiplied by the value from the sum of previous multiplications entering that node and the same procedure is applied. This multiplication operation is called the “dot product” because the operations are made with vectors.

Calculation of Loss Function and Cost Function

The loss (error) function calculates how much loss the established deep learning model. It is a measure of error between the prediction result and the actual result. The output of the model established as a result of the training set and the results reserved for the test are compared through this function.

The cost function includes the summation of loss functions that occur after the training process of each input.

Backward Propagation and Updating Parameters

After the learning process of a neural network, generally cost function is higher. The weights and bias values should be updated to decrease the cost incurred due to wrong predictions. The process beginning with the cost function and ending with the updating weights and bias is named backward propagation.

The Gradient Descent Method is applied to find the appropriate values of weights and bias according to cost. Cost cannot be zero but proper adjustment of weights reduces the error rate and makes the model certain.

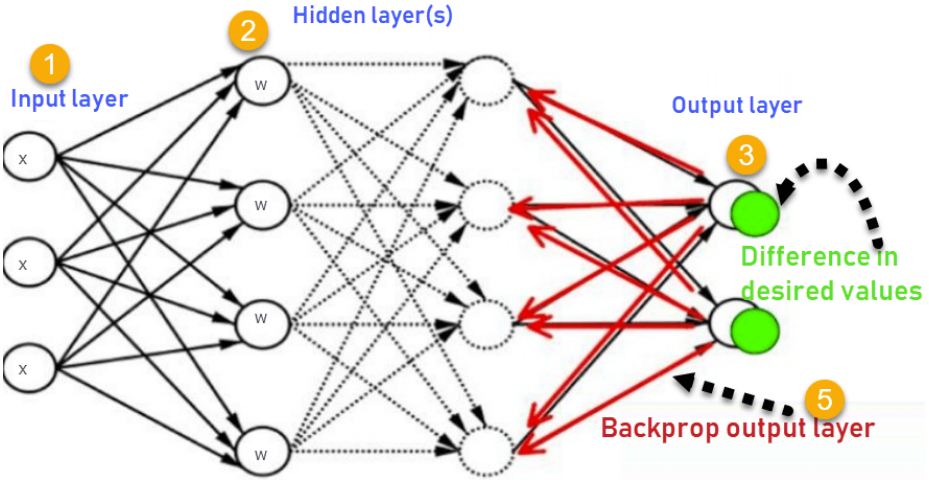


Figure 5. Forward and Backward Propagation Illustrated with Arrows

Optimization Algorithm with Gradient Descent Method

The aim of the Gradient descent Algorithm is finding the local minimum value of the differentiable function iteratively and repeatedly to be able to find optimized values of the weights and bias.

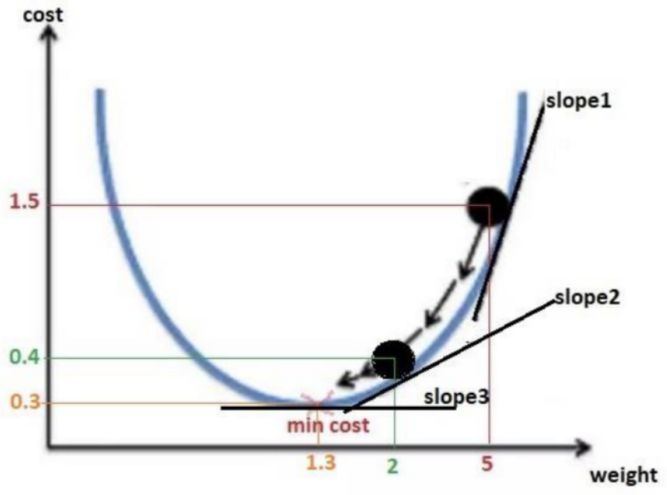


Figure 6. The Graph of Cost with Respect to Weights

The slopes of the cost-weight/bias graphs are used to find the local minimum point because the finding slope of a graph yields to find the derivative of the cost values according to weights and bias. The updated weights and bias values can be found with these differentiable values. To achieve that in backward propagation, usually the Sigmoid function is used as an activation function. Since this is a derivative and integrable function, it can be used in both forward propagation and backward propagation. Thus, the wrong output can be corrected by adjusting the weight and bias values with the help of the differentiable property of the sigmoid function in the backward propagation process.

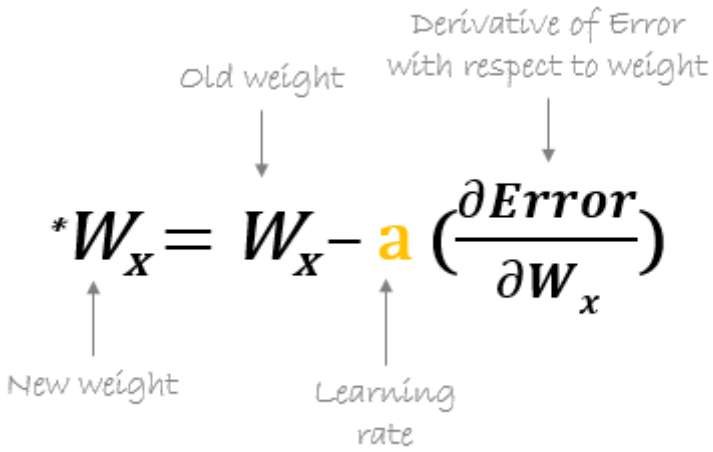


Figure 7. The Mathematical Formula to Update Parameters

It can be seen from the equation that the new, updated weight equals to the difference between the old weight and the multiplication of the learning parameter with the derivative of the error with respect to weight. Learning rate is a hyperparameter that needs to be tuned intuitively.

Prediction with Learnt Parameters Weight and Bias

After the training process, the model should be tested to check whether the program gives correct results or not. To achieve that process, the test data is used to control the model instead of the train data. Forward propagation is applied to the given input for the testing step. After this step, the output value is compared with the threshold value that is specified for the classification of data. According to the comparison, the given test input is classified as its corresponding label. Therefore, the model makes a prediction about the input.

4.2. Theory of Convolutional Neural Network

The Convolutional Neural Network has solid similarities with the human brain. Most particularly, it is used for image recognition or classification. In this project, the CNN model is used to identify the images by separating them corresponding to their classes. A couple of steps must be followed to create a Convolutional Neural Network.

4.2.1. Convolutional Neural Networks

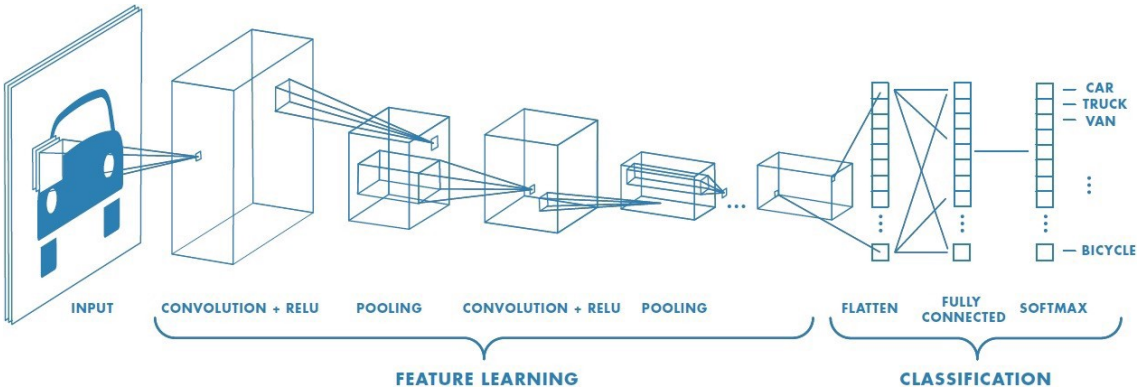


Figure 8. The Architecture of Convolutional Neural Networks (CNN)

4.2.2. What is an Image?

Images are inputs to CNN. Images are formed from numbers. In other words, they consist of array of pixels. They can be thought as matrices. Each pixel has an RGB value between 0-255.

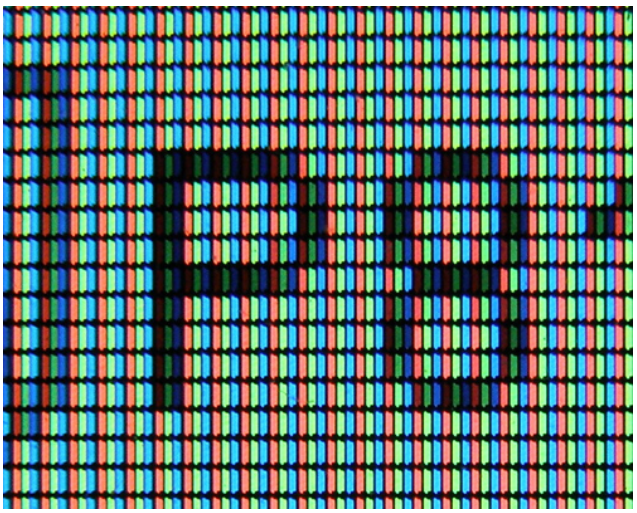


Figure 9. Pixels with Red, Green and Blue Values

4.2.3. Normalization, Reshape and Label Encoding

Normalization is used in pre-processing step to standardize pixels of data to between 0 and 1. Because if the general form of pixels which has the values between the range 0-255 is used, then a neural network works slower. In addition, this normalization can also be named as grayscale normalization because of the representation of 0 as black and 1 as white.

Reshaping is denoting input pixels and values in a 3D matrix in a way that CNN can understand. For example, if colorful images that include red, green and blue tones are given to a neural network, the input matrix should be (n, n, 3) where n is the vertical and horizontal pixel numbers. Since the digits are in grayscale format after normalization process that is why the input matrix for the network is defined as (n, n, 1).

State (Nominal Scale)	State (Label Encoding)
Maharashtra	3
Tamil Nadu	4
Delhi	0
Karnataka	2
Gujarat	1
Uttar Pradesh	5

Figure 10. Example of Label Encoding for the String Values

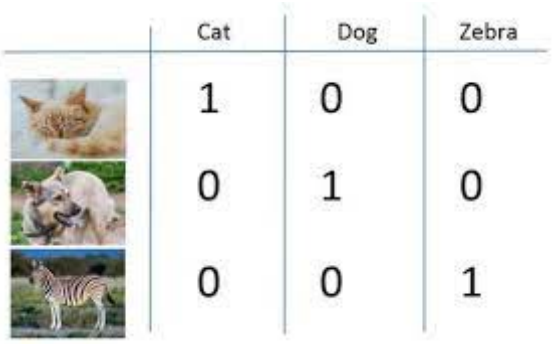


Figure 11. Example of Label Encoding for Images

Label encoding is that labels are encoded to different formats in a way that computers can understand. Computers understand just numbers instead of strings. Thus, string labels are converted to numbers that represent each class. Keras does this in a similar manner to byte-array.
For instance, it uses [0, 0, 0, 1, 0, 0, 0, 0, 0, 0] for label 3.

4.2.4. Train Test Split

Input data must be split into two parts which are Train Data and Test Data. Because if all images are used without splitting, the neural network memorizes all the images. This is called as overfitting. It's a massive problem. Hence, Convolutional Neural Network Model is created with a train dataset and a test dataset is used in testing process to measure the accuracy of the model and learn how well the model.

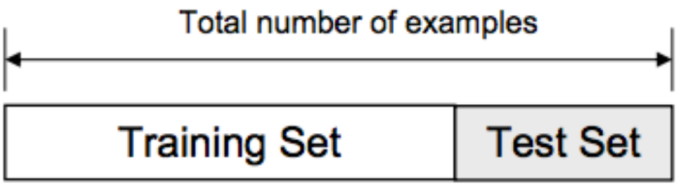


Figure 12. Splitting of Dataset

4.2.5. Train Validation Split

Train data is split one more time. Because the validation set is used to evaluate a given model, but this is for frequent evaluation. This data is used to fine-tune the model hyperparameters. Validation does not mean testing. Validation is immediate testing, just to tune the neural network immediately.

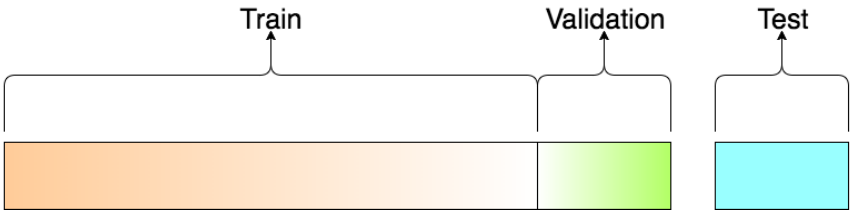


Figure 13. Splitting of Train Set to Get Validation Set

4.2.6. Data Augmentation

One of the biggest problems of neural networks is overfitting. It basically means that the neural network have memorized all the dataset. This is not desired. The problem with memorizing a dataset is the fact that if a variation of the data is fed as the input the to a neural network since it has memorized the dataset, it fails to predict the data correctly. A technique, called data augmentation, is needed for a neural network to define different variations of the data.

Data Augmentation is artificially creating a huge variety of the dataset, in other words enlarge the train dataset, so that the neural network does not memorize the dataset and "encounters" almost all variations.

Data Augmentation can be done following the steps like rotations, flips, color balance, zoom-ins and zoom-outs, and translations in X and Y directions. The examples of methods in data augmentation are given below.

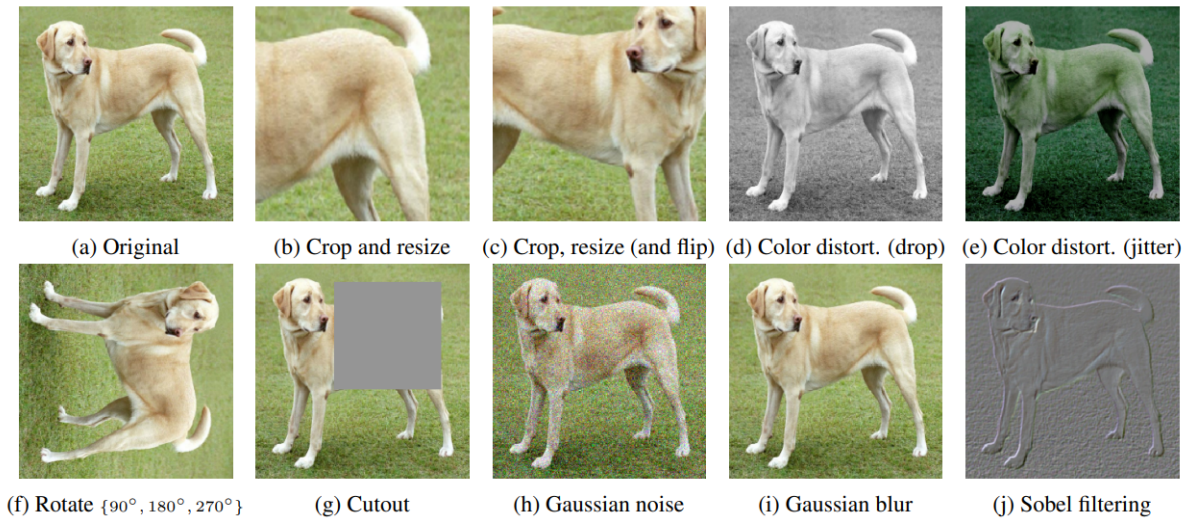


Figure 14. Methods Used in Data Augmentation

4.2.7. Feature Detector

A filter, named as also kernel, is used to distinguish and identify plenty of different features like vertical lines, horizontal lines, edges, bends, and convex shapes (complex features) on images. These features are extremely important in classification. Because classification is made by using these features. Feature detectors are commonly 3x3, 5x5 and 7x7 matrices but they can be selected in any size in the form of n by n.

Feature detectors are generally used in convolution operation.

4.2.8. Convolution Layer

This layer is the main building block of Convolutional Neural Network. It is responsible for perceiving the features of image. This layer applies some filters to the image to extract lower and higher level features from the image.

After making convolution operation with feature detector, feature map which is newer matrix having less features is obtained. There are some calculations for the convolution operation. First of all, the feature detector is positioned in the upper left corner of the image. Indices between image matrix and filter matrix are multiplied with each other and all results are summed. Afterwards, the result, convolved feature, is stored in the output matrix and the filter is moved 1 pixel/step to the right.

This movement is named as stride value that gives the number of jumping to the right. This process is repeated for the first line of the image matrix. After the 1st line is finished, it is passed to the 2nd line and the operations are repeated. After all the processes are finished, the output matrix is created which is the feature map. There is given an example of convolutional operation in figure below.

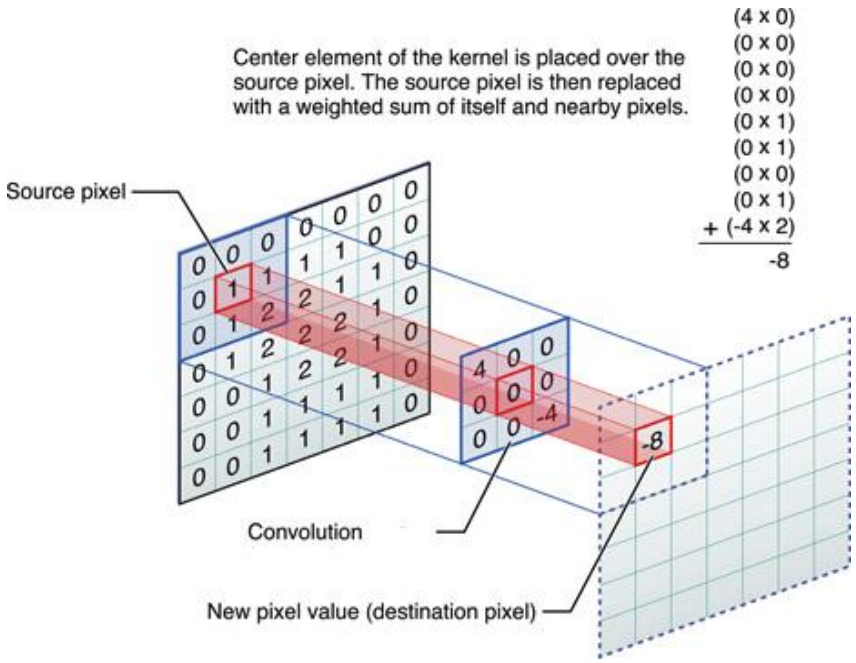


Figure 15. Convolution Operation for the Upper Left Corner of the Image Matrix

There are lots of different features in an image. Thus, multiple feature filters are used in an image to detect these features. This causes to get also many feature maps.

The purpose of the convolution operation is to reduce the size of the image. It is very efficient to speed up the execution of code. However, some information is lost due to this technique.

Secondly, Activation Function ReLU (Rectified Linear Units) is applied to the feature map during Convolution Operation to break up linearity and increase nonlinearity.

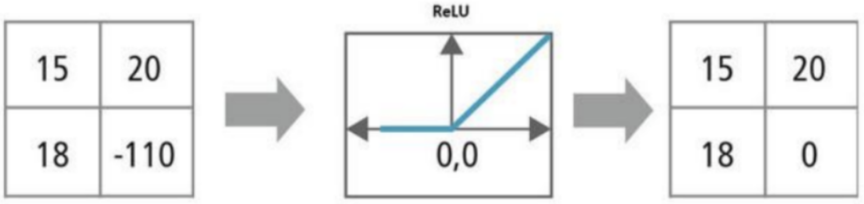


Figure 16. Removing Negative Values of Pixels After ReLU to Break Linearity

As can be seen, negative values are removed after ReLU. If the given input to ReLU is smaller than zero, the output will be zero. Otherwise, it will be the same as the given input. After this calculation, linearity is broken. Non-linearity is desired in Artificial Intelligence. Because if linearity continues, all the operations would be affected by each other and there would be no mean to apply tons of operations.

4.2.9. Same Padding

The size of the feature map is always less than the size of the original input due to the convolution operation. Therefore, some information is lost because of the decreasing size of volume after convolution operation to get features in the image but information is not wanted to be lost. Information about the original input volume should be preserved in the early layers of the network so that the low-level features can be extracted instead of to be lost. Same padding is found to be able to achieve that. It is named as “same” because the aim is to keep input size and output size the same.

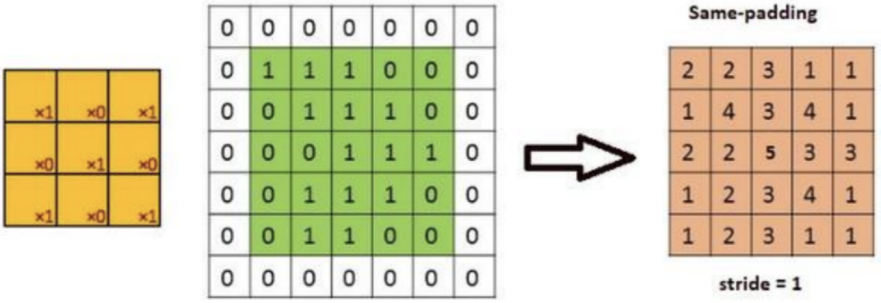


Figure 17. Application of the Same Padding

In the given figure above, the green matrix can be thought as an image. A frame filled with zeros is drawn around this matrix. A resulting matrix is obtained by shifting the filter over this newly formed matrix. So, the resulting matrix has the same size with the input matrix. This helps to prevent loss of information and the reduction of size.

4.2.10. Max Pooling

This layer is a layer that is usually added between consecutive convolutional layers in CovNet. All the information in images cannot be kept until the end of a neural network. This results in creating a vector of thousands of pixel information, making it impossible to train the ANN. Because if the size is large, the neural network will run slowly. Hence, max pooling aims for dimensionality reduction by making down-sampling. In this way, both the required processing complexity is decreased and the unnecessary caught features are ignored and more important features are focused on. That is why some representatives are to be chosen. This means the image is divided into pools, and a value is chosen to represent that pool, in the end, even though some pixel information is lost, this would make it real to train the ANN.

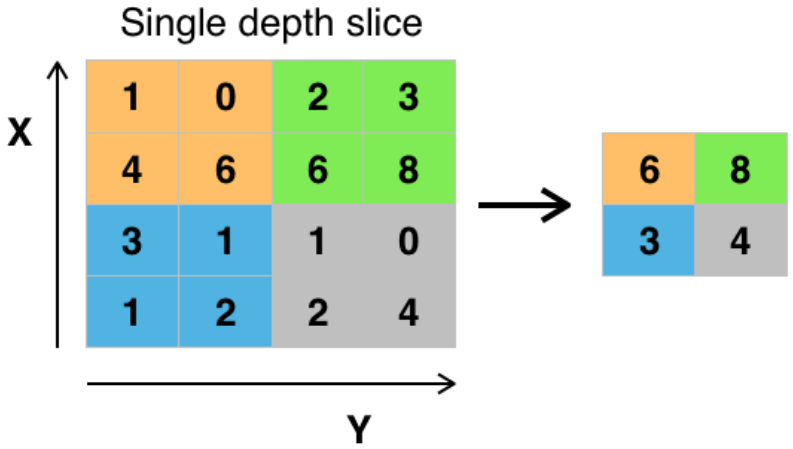


Figure 18. Application of the Max Pooling to the Feature Map

The pooling layer is usually applied to the generated feature maps. In the given figure above, there is shown a 4x4 feature matrix. If max pooling is applied by dividing 4x4 matrix to 2x2 matrices, the 2x2 resulting matrix on the right is created.

4.2.11. Flattening

After the Convolutional layer and the Pooling layer, the flattening part follows. The Convolution and Pooling layers can be used multiple times in the model. After these processes are complete, the resulting matrix must be flattened so that it can be used in the Fully Connected layer. Because the main task of this layer is to prepare the data in the input of the last and most important layer, Fully Connected Layer, feature map.

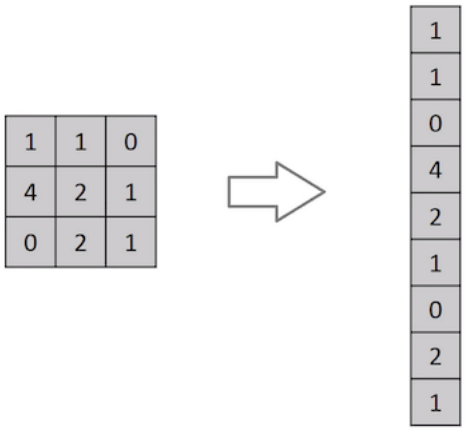


Figure 19. Flattening a Matrix

Generally, neural networks take the input data from a one-dimensional array. In other words, the matrices having 2D or 3D dimensions should be converted to 1D vector and then ANN can be fed with them because Artificial Neural Networks work with 1D vectors. The data in this neural network is the one-dimensional array of matrices getting from the Convolutional and Pooling layers. After this stage, the inputs of the Fully Connected Layer are prepared.

4.2.12. Full Connection

In the full connection, the classification is done by taking features. It also includes an activation function to achieve complicated calculations for predictions. It is almostly same with an artificial neural network. The only difference is that all nodes in a layer don't have to be connected to nodes in the next layer in ANN, but when it is said "full connectivity", it means that all nodes in a layer have to be connected to nodes in the next layer.

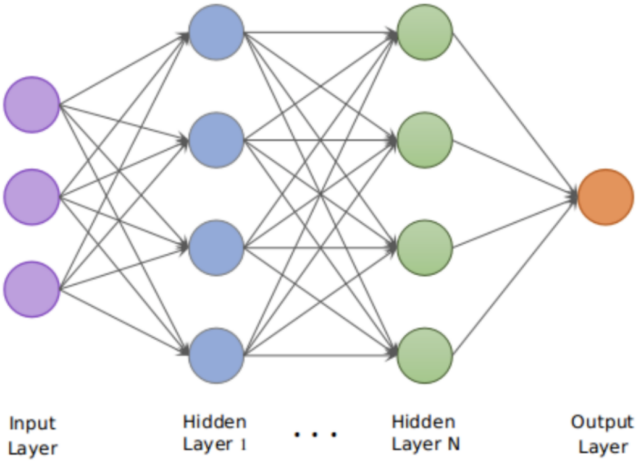


Figure 20. Neural Networks Structure

This structure can also be seen as the fully connected layer structure. After processing an image with convolution operations and pooling operations, a 1D vector representing this image is gotten, this will get into an Artificial Neural Network as an input to make predictions. First of all, the input layer part is created with the data obtained by flattening. Then, the values coming from the input layer are processed in the hidden layers with the coefficients (mostly functions) determined by the model. As a result of this process, the output value is produced according to the activation function determined.

4.2.13. Applying Dropout

Dropout is a technique used to break connections between some randomly selected neurons during training by specifying randomly selected threshold value. The mathematical counterpart is ignoring some neurons while making a forward-propagation. The aim is to make the Artificial Neural Network (ANN) more stable and to avoid memorizing and overfitting the data. Because there is a risk of overfitting if a neuron network is too large, if a network is trained for too long, or if the number of data of a network is too small.

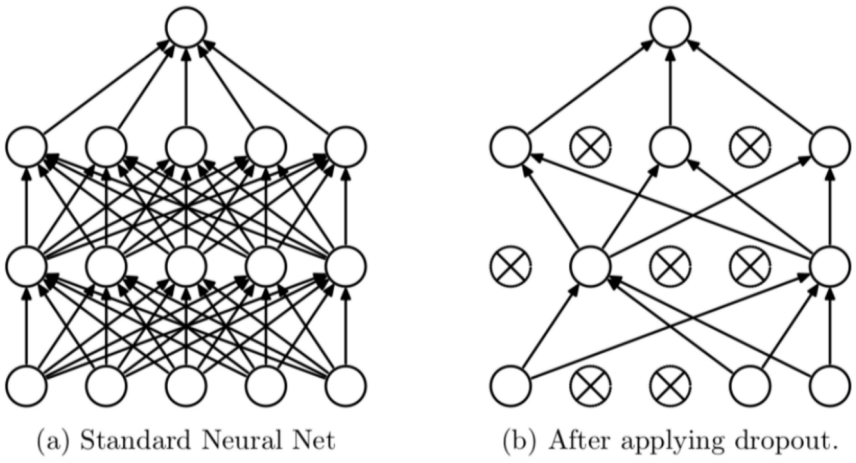


Figure 21. Applying Random Dropout

4.2.14. Batch Normalization

Batch normalization is another method used to make the convolutional neural network more regular. With batch normalization, layers in the network do not have to wait for the previous layer to learn. It allows simultaneous learning. This reduces training time and enables the model to perform better.

If a high learning rate is used without using batch normalization, the problem of disappearing gradients may be encountered. However, with batch normalization, higher learning rates can be used because the change in one layer does not spread to the other layer. In addition, the batch normalization makes the network more stable and organized.

4.2.15. Setting Up An Optimizer

While doing backpropagation, the learning rate is one of the most critical factors while learning, (i.e. taking the derivative of the cost function), if the learning rate is high, the neural network might not be able to learn. If it is too low, it might not learn at all.

Therefore, the learning rate must be some kind of dynamic. This optimizer achieves by changing the learning rate so that a neural network can be trained and can learn faster. It sets the learning rate to higher or lower depending on the speed of reaching the minimum cost.

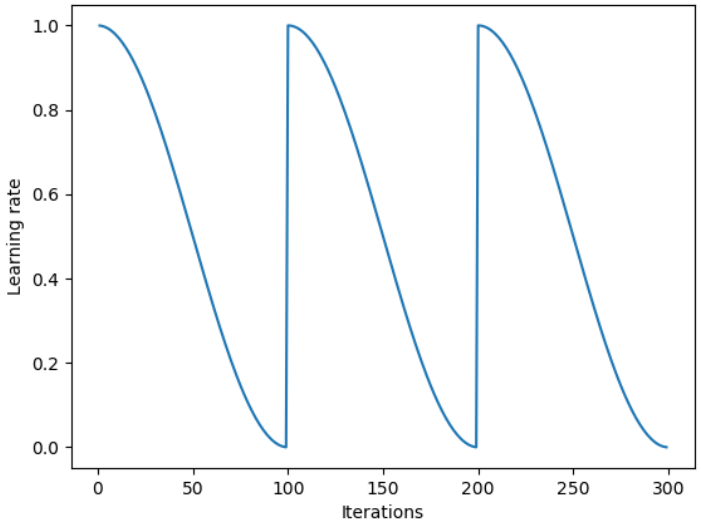


Figure 22. Changing Learning Rate with the Optimization

4.2.16. Epochs and Batch Size

The epoch (cycle) number is the number of times that all training data is shown to the network during training. An epoch represents that a whole dataset is passed forward and backward through the neural network only one time. Since an epoch is too large to be given to the computer at the same time, we divide it into several smaller groups.
Even if the accuracy of the training increases, the number of epochs is increased until the validation accuracy starts to decrease. The number of epochs is optimal where accuracy is highest.

Batch size is the number of samples processed before the model is updated. In Deep Learning applications, learning by processing all the data in the dataset at the same time is a costly task in terms of time and memory. Because, in each iteration of the learning, the gradient descent is calculated retrospectively on the network with the back propagation process and the weight values are updated in this way. The higher number of data in this calculation process causes the longer the calculations. To solve this problem, the dataset is divided into small groups and the learning process is performed on these selected small groups. Processing multiple entries in chunks is called a "mini-batch". The value determined as the mini-batch parameter while designing the model means how many data the model will process at the same time.

5. Dataset Overview

The Dataset: https://www.kaggle.com/datasets/preetviradiya/brian-tumor-dataset

The dataset is published on Kaggle. Organized into 2 different folders, namely “Brain Tumor” and “Healthy”. It includes 4600 Brain X-Ray images of both healthy and people having Brain Tumor. To be more specific, there are 2513 X-Rays of Brain Tumor and, there are 2087 Images of Healthy people’s brains.

All images are labeled by default since they are categorized inside distinct folders containing the label information. Additionally, the images vary in format, ranging from JPEG to PNG.

In addition, images vary in terms of colors. Some images are monochrome and some are RGB. It is also worth mentioning that, the sizes of the images vary as well. Therefore, it is not constant and must be resized.

The figure below illustrates a variety of images taken among the dataset as a sample.

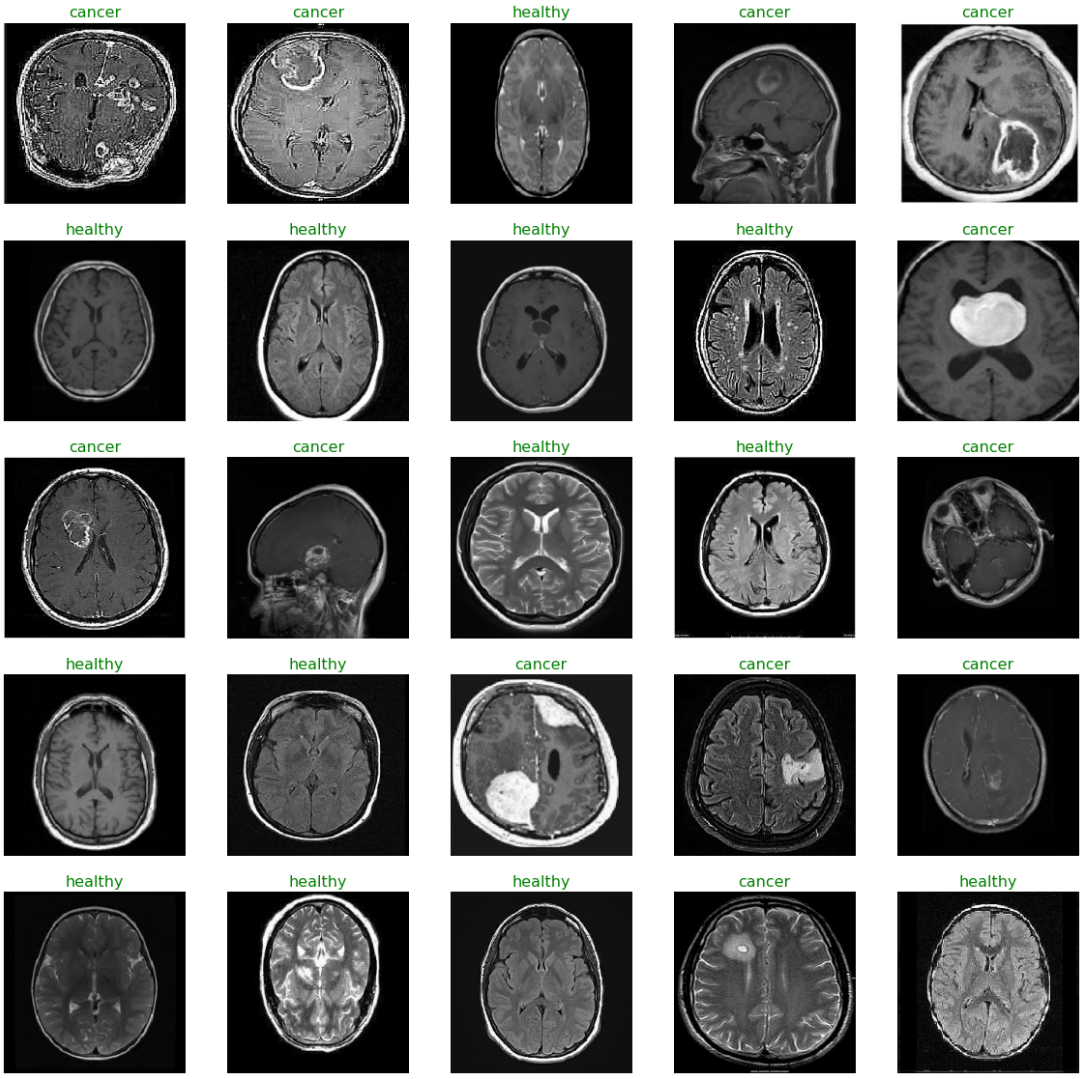


Figure 23 illustrates a couple of images taken among the dataset for the purpose of sampling.

6. Implementation

All the theory related to Artificial Neural Networks and Convolutional Neural Networks are discussed. Till this part of the report, “why” part has been covered in detail and in this part, “how” part will be clarified.

In this part, a couple of points are illustrated. First is the architecture of this newly-developed convolutional neural network. Second, manipulation of the dataset before getting trained. Third is the result of the training. Moving beyond theory towards implementation.

6.1. Loading Images and Preprocessing

6.1. Loading Images

Datasets are large sums of data, and as known, data is stored in computers' storage. However, programs do not work with storage, they work with memory (RAM). For this reason, dataset must be loaded into the memory. Even though, it seems as an easy task, this step is one of the building blocks of a neural network. If a mis-loading occurs, it causes an instability.

Loading the Dataset step is mentioned as Image Reading in the code. Image Reading is implemented using Keras' default **ImageDataGenerator** class and with is **flow_from_directory** method. What is basically does is that reading the image into the memory with a provided path. The only consideration took place is that image path must be accurate.

After reading the images, the images (data) is instantly resized 150x150 pixels, which is also done by the same method.

The last criterion is that in RAM, it is a must to store images with their corresponding labels. Otherwise, training steps would not be performed without labels. In order a neural network to learn, it has to have a label, indicating whether an image is belonging to a patient with Brain Tumor or normal patient. Since dataset is well organized into distinct folders, naming the labels, the labeling is done by the same method as well. There is no need for another labeling step.

At the end of this step, images are loaded to the memory, images are resized and images are mapped with their correct labels.

6.1.2. Train-Validation Split

The dataset is split into two parts as training and validation data. This is also done by Keras' flow_from_directory class by providing the folder name. Train data is used for training the neural network. And validation data is used for both testing and validation of the neural network. The split ratio is perfectly fine-tuned to 22%. Meaning 78% of the images are directly used for training, which is a high number.

In this project, validation data is used both for the purpose of validation and testing. This decision is made so that no more data is lost for testing and all the data could be used for training, resulting in an extremely high number of accuracy. Therefore, there is no test data. Validation data is used as test data as well.

6.2. Normalization

In a typical neural network, trillions of mathematical operations are done. From taking derivatives to matrix multiplications. If higher values are used for mathematical operations, an example is that a pixel is between 0 - 255, it would slow down the neural network. Therefore, statisticians have come up with the idea of normalization.

Normalization is basically, squeezing the data into a range. There are a lot of normalization techniques. Moreover, in this project, the Divider Method normalization has been chosen for training. In Divider Method normalization suggests squeezing the pixel values from 0 - 255 to 0 - 1 by dividing each pixel data by the maximum value for a pixel which is 255.

This is done in the step for loading the images. Before loading the images, since ImageDataGeneration class is also used for augmentation purposes as well, the normalization ratio of 1/255 is provided. This means that while getting the images from the directory, they are normalized to 0-1.

6.3. The Network Architecture of This Project

All electronic devices and all software solutions have a brain, and "the brain" of a convolutional neural network application is ultimately its convolutional layer architecture. Therefore, this step is the most significant step in this solution.

In this step, the overview of the architecture of the Project, architecture choices, configuration choices, neural network hyper-parameter choices, optimization decisions and software approach are revealed. At the end of this step, a function that is responsible for building the neural network with requested parameters are obtained.

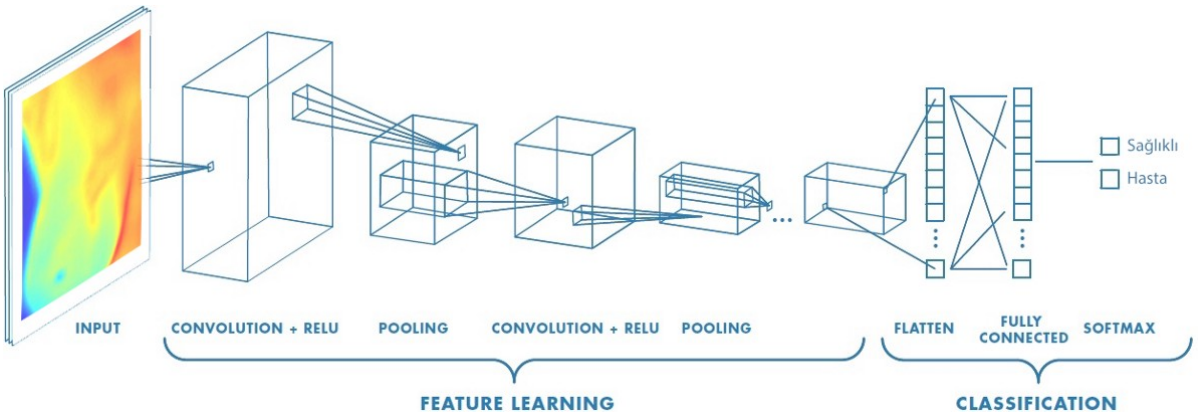


Figure 24. Overview of a Convolutional Neural Network

A Modular Approach

The principle of separation of concerns is applied. Therefore, a modular approach is adopted. a **BuildTheModel()** function is responsible building the model with given parameters. It could further be developed to a Python package or class later on if desired.

The Unparalled Architecture

As expressed above, this step actually determines the flow of the whole neural network. Responsible from training time to the accuracy of the network. Therefore, a psecial architecture, named the Unparalleled Architecture, has been developed.

The Unparalleled Architecture states that there are 3 convolution layers. In each layer, the following are applied step by step:

- Feature Filtering With a Specific Kernel Size
- Stride Operation
- Padding Operation
- Activation Function
- Dropout
- Batch Normalization
- Max Pooling

As mentioned above, these steps are actually the brain of the neural network and extremely important. Fine-tuning all the parameters about training has taken a lot of time and mathematical skills. Moreover, a couple of influential decisions are taken along the way.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 150, 150, 32)      896
_________________________________________________________________
dropout (Dropout)            (None, 150, 150, 32)      0
_________________________________________________________________
batch_normalization (BatchNo (None, 150, 150, 32)      128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 75, 75, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 75, 75, 64)        18496
_________________________________________________________________
dropout_1 (Dropout)          (None, 75, 75, 64)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 75, 75, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 37, 37, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 37, 37, 32)        18464
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 18, 18, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 10368)             0
_________________________________________________________________
dense (Dense)                (None, 156)               1617564
_________________________________________________________________
batch_normalization_2 (Batch (None, 156)               624
_________________________________________________________________
dropout_2 (Dropout)          (None, 156)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                10048
_________________________________________________________________
batch_normalization_3 (Batch (None, 64)                256
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 1,666,797
Trainable params: 1,666,165
Non-trainable params: 632
_________________________________________________________________
```

Figure 25. The Model Summary of the Architecture of the Neural Network

To sum up, in this step, the convolutional layers are developed and fine-tuned.

6.4. The Actual Implementation of the Neural Network

A couple concepts must be illuminated while getting deeper into the implementaion of the neural network.

6.4.1. Loss Function

Loss function is a key component in training a neural network. Loss function determines how much the prediction made in forward propagation step deviates from the actual result. By getting a feedback, via a loss function, the weights and biases could be updated in backward propagation step. Therefore, loss function matters plenty.

Loss function can be examined in two categories. First category is binary cross entropy functions, and second category is categorical cross entropy functions.

Categorical Cross Entropy functions are used as a loss function for multi-class, multi-label classification, where there are more than two output labels. Whereas, Binary Cross Entropy functions are used as a loss function for binary classification, which means there are only two output values.

In this project, Binary Cross Entropy is used as loss function, since there are only two outputs.

5.4.2. Activation Function

Activation functions are developed by mimicing the human brains' behavior. As researchers explore human brains' nature, it has been figured out that neurons in brain decides whether or not to fire based on the incoming signals. Activation functions simply do the same thing, mathematically. First, an artifical neuron (node) sums up the incoming parameters with its weights and after that, it has to decide whether or not fire, and how tough to fire (i.e. its result). This decision is taken by an activation function. To put in better words, an action function how the weighted sum of the input is mutated to an actual input from a node.

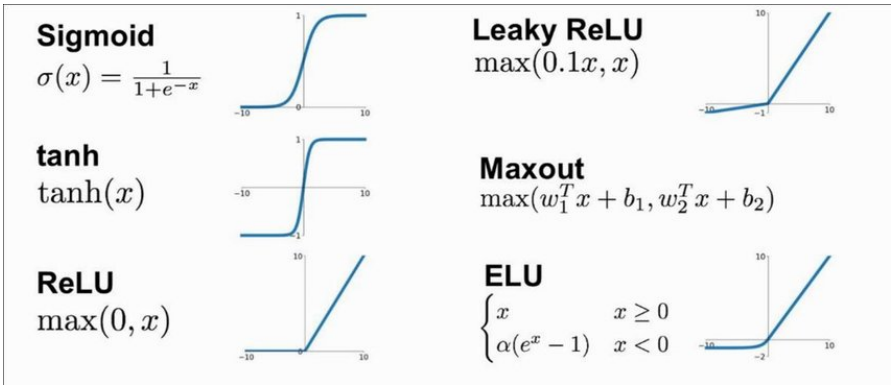


Figure 26. The Best Known Activation Functions

In this project, ReLU is found out to be an extremely great choice for this neural network. One of the main benefit of ReLU is the fact that it is kind of linear, meaning that it is extremely easy and fast to take its derivative. It has been tried whether sigmoid could be an alternative, however, sigmoid turned out to be extremely slow in the training process. Therefore, ReLU is concluded.

The activation function resides at the end of this convolutional neural network is Sigmoid. This has been also a decision to be optimized, and sigmoid is found out to be a great binary classification last-step activation function. Therefore, chosen to power the last step of this neural network.

6.5. Learning Rate Optimization

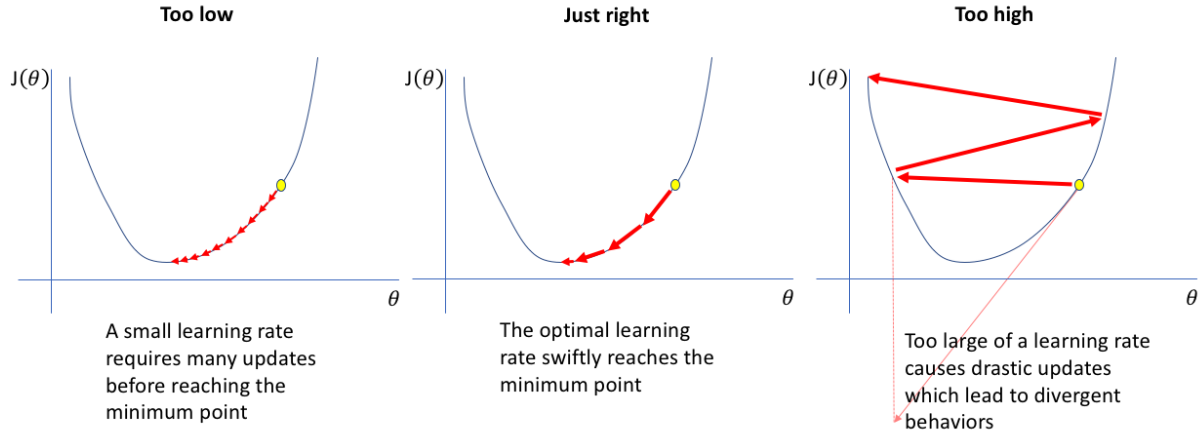


Figure 27. The Results of Learning Rate's Being Too Low or Too High

The choice of optimization algorithm when training deep learning model can mean the difference between good results in minutes, hours, and days.

The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing.

In this project, Adam optimizer is used for fine-tuning the learning rate all the time.

6.6. Training the Model

Training a neural network means reaping the benefits of all the steps till here. In training, the neural network is fed with pre-processed, augmented, cleaned dataset, and expected to learn, i.e. trained, to predict correctly in the future. The result of the learning process is trained, correct, or close-to-correct weights and biases.

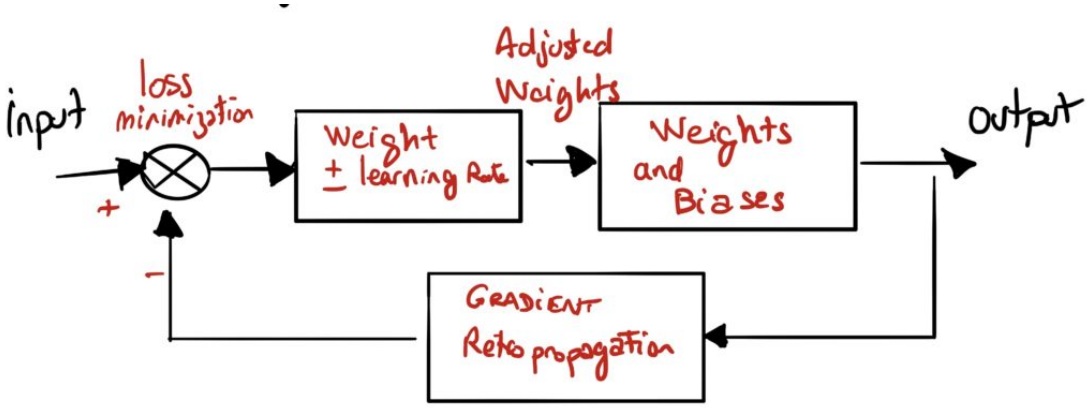


Figure 28. The Training Process of a Neural Network

A couple of configurations must be done before getting into training.

Fine-Tunement of the Number of Epochs

Among them is **epochs**. It is significant because it strictly determines the accuracy of the neural network and the training time. If number of epochs is too low, the neural network does not have enough chances to learn, it too high, it would take excessive amount of time to train. It also must be fine-tuned, and 12 is figured out to be a great number, since the loss of the network seems to be dipped.

```
2022-05-12 22:48:53.029294: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization Passes are enabled (registered 2)
Epoch 1/12
29/29 [==============================] - 114s 4s/step - loss: 0.4654 - accuracy: 0.7782 - val_loss: 0.6709 - val_accuracy: 0.5786
Epoch 2/12
29/29 [==============================] - 105s 4s/step - loss: 0.2676 - accuracy: 0.8866 - val_loss: 1.1375 - val_accuracy: 0.5460
Epoch 3/12
29/29 [==============================] - 111s 4s/step - loss: 0.1743 - accuracy: 0.9317 - val_loss: 2.0387 - val_accuracy: 0.5460
Epoch 4/12
29/29 [==============================] - 111s 4s/step - loss: 0.1886 - accuracy: 0.9234 - val_loss: 1.2695 - val_accuracy: 0.5480
Epoch 5/12
29/29 [==============================] - 105s 4s/step - loss: 0.1077 - accuracy: 0.9579 - val_loss: 1.8032 - val_accuracy: 0.5460
```

Figure 29. Showing the Result of the Learning Process from the Training Process

As shown in the figure above, the results are marvelous. **The validation accuracy is above 98%**, and the loss is reduced close to 0, which is an outstanding value. Furthermore, the training process takes approximately twenty minutes on Kaggle's cloud training platform.

After this step, the model has been trained successfully, efficiently and relatively fast. In addition to this, the result is weights and biases, that are highly accurate, that detect brain tumor. The neural network is now complete, trained and ready to be served for business uses.

6.7. Saving the Model

The neural networks are generally developed for business or research purposes. Therefore, it is essential that after training a neural network, in other words, after getting the accurate weights and biases, those values must be remembered, must be saved. Because, neural networks are generally developed for AI as a Service purposes, in other words, the neural network will be served via an HTTP Web API, for uses from an interface. Thus, in this step, after having trained the neural network, its result is saved to an .h5 file.

How is This Model Going to be Used Later on by a Web API?

In the future, a Web API can then load the model in the memory, and then can make predictions with the images it has taken via HTTP, from a client.

6.8. Model Evaluation

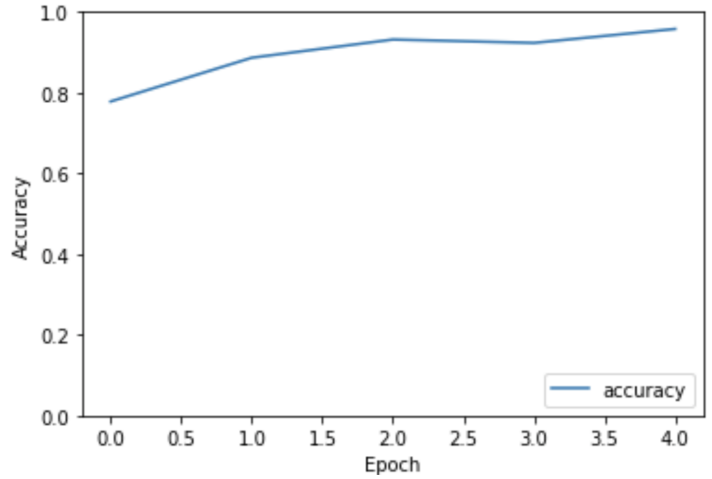
One of the main questions of training a neural network is how to make sure that the neural network performs accurate and in a requested way. Recall from the loading step, the data is divided into two parts, namely, train and validation data. Validation data is used for testing how well or poorly this neural network performs. Keep in mind that, neural networks are evaluated based on their accuracy.

```
29/29 [==============================] - 105s 4s/step - loss: 0.1077 - accuracy: 0.9579 - val_loss: 1.8032 - val_accuracy: 0.5460
```

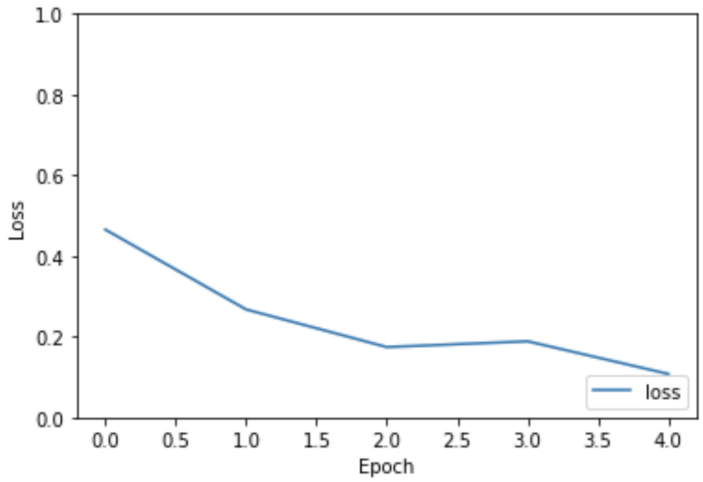
Figure 30 reveals one of the steps of training.

As the figure above and the graphs below clearly indicate, the neural network has an accuracy of greater than 98%, which is a spectacular number. Far better, far accurate than a doctor will ever be. Another thing to keep in mind the fact that training process took very little time. While developing a high-accurate neural network, the training time is not ignored, or underestimated. Training time and accuracy are both optimized.

The Graph of Accuracy with Respect to Number of Epochs



The Graph of Loss with Respect to Number of Epochs



7. Conclusion

The revolution of computers and intelligent machines have altered the world irreversibly and in an unprecedent manner. In the past, for healthcare, people relied solely on doctors. Considering the fact that it takes more than 25 years to educate a doctor, this fact has had serious consequences in years. One consequence has been the fact that it is extremely difficult to access a great physician in less-developed countries and continents, such as Africa, Far East or rural areas, which has mostly resulted in misdiagnoses, late-diagnoses or no diagnoses at all.

However, this situation is currently seemed to be changing. Due to the artificial intelligence revolution, healthcare will be and is becoming to be more accessible, more affordable and most importantly, more accurate. In Cerebri Tumor Deprehensio, this social purpose has always been paid regard to, and has been the main motivation for development this project.

To conclude, developing a highly accurate and efficient neural network has been a challenge and lies at the heart of this project. Nevertheless, despite all the technical and engineering challenges, a highly accurate, close to an accuracy of 99%, neural network has been trained. While training this marvelous neural network, even though accuracy has always been the main concern, the training time has never been a low priority, rather, it has always been a thing that must be optimized. Therefore, this neural network trains insanely fast, probably a record time.

Finally, furthermore, saving the results, i.e. weights and biases, of the neural network has also been a challenge and implemented an elegant solution successfully.

REFERENCES

Artificial Neural Network - applications, algorithms and examples, 2021, TechVidvan, https://techvidvan.com/tutorials/artificial-neural-network

Babs, 2020, The Mathematics of Neural Networks, Medium, https://medium.com/coinmonks/the-mathematics-of-neural-network-60a112dd3e05

Bronshtein, 2020, Train/test split and cross validation in Python, Medium, https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

Cai, L., Gao, J., & Zhao, 2020, A review of the application of deep learning in Medical Image Classification and segmentation, Annals of Translational Medicine, https://atm.amegroups.com/article/view/36944/html

DeepAI, 2019, Max pooling, DeepAI, https://deepai.org/machine-learning-glossary-and-terms/max-pooling

Johnson, 2021, Back Propagation Neural Network:
What is backpropagation algorithm in machine learning?, https://www.guru99.com/backpropogation-neural-network.html

K., H. M., 2019, Backpropagation step by step, RSS, https://hmkcode.com/ai/backpropagation-step-by-step/

Kathuria, 2020, Intro to optimization in Deep learning: Gradient descent, Paperspace Blog, https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent

kanncaa1, 2020, Convolutional Neural Network (CNN) tutorial, Kaggle, https://www.kaggle.com/kanncaa1/convolutional-neural-network-cnn-tutorial

Kooi, 2021, Photometric data augmentation in projection radiography, Medium, https://medium.com/lunit/photometric-data-augmentation-in-projection-radiography-bed3ae9f55c3

Mishra, 2020, Convolutional Neural Networks, explained, Medium, https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

Brain X-Ray Images, Kaggle, https://www.kaggle.com/datasets/preetviradiya/brian-tumor-dataset

Mulla, 2020, Cost, activation, loss function, neural network deep learning, what are these? Medium, https://medium.com/@zeeshanmulla/cost-activation-loss-function-neural-network-deep-learning-what-are-these-91167825a4de

Artificial neural network, TeX, https://tex.stackexchange.com/questions/132444/diagram-of-an-artificial-neural-network

Ramadhan, 2021, Neural network: The essence of artificial neural network, Medium, https://towardsdatascience.com/neural-network-the-essence-of-artificial-neural-network-c605eb32de5

Shah, 2020, About train, validation and test sets in machine learning, Medium, https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

Wikimedia Foundation, 2021, Artificial Neural Network, Wikipedia, https://en.wikipedia.org/wiki/Artificial_neural_network

Made by Muhammed Fatih Çinar, 15290744
Computer Engineering (100% English)