

BRAIN TUMOUR DETECTION SYSTEM USING DEEP LEARNING

Submitted in partial fulfillment of the requirements for the award of Bachelor
of Technology degree in Biomedical Engineering

By

VISHAL RS (37240078)



**DEPARTMENT OF BIOMEDICAL ENGINEERING
SCHOOL OF BIO AND CHEMICAL ENGINEERING**

**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

(Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE)

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

MARCH 2021



**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with “A” grade by NAAC | 12B Status by UGC | Approved by AICTE
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119
www.sathyabama.ac.in

DEPARTMENT OF BIOMEDICAL ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Vishal R S (37240078)** who carried out the project entitled “**BRAIN TUMOUR DETECTION SYSTEM USING DEEP LEARNING TECHNIQUE**” under my supervision from **October 2020** to **March 2021**

Dr. T. SUDHAKAR, M.Sc., Ph.D.,
Internal Guide

Dr. T. SUDHAKAR, M.Sc., Ph.D.,
Head of the Department

Submitted for Viva voce Examination held on 10.4.2021

Internal Examiner

External Examiner

DECLARATION

I, **Vishal RS (37240078)** hereby declare that the Project Report entitled **“Brain Tumour Detection System using Deep Technique”** done by me under the guidance of **Dr. T. SUDHAKAR**, Department of Biomedical Engineering is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Biomedical Engineering.

DATE : 10.4.2021

PLACE : CHENNAI

A handwritten signature in blue ink that reads "Vishal Senthil". The signature is written in a cursive style with a horizontal line underneath.

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to the **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. T. Sudhakar** M.Sc., Ph.D., and **Dr. J. Premkumar** M.Sc., Ph.D., **Head of the Department, Dept. of Biomedical Engineering** for providing me necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Dr. T. Sudhakar**, M.Sc., Ph.D., **Department of Bio Medical Engineering** for his valuable guidance, suggestions and constant encouragement paving the way for the successful completion of our project work.

We wish to express our thanks to all **Teaching and Non-teaching** staff members of the **Department of Biomedical Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The occurrence of brain tumours in India is steadily rising. More and more cases of brain tumours are reported every year within the country among people of assorted age groups. In 2018, brain tumours was ranked because the 10th most typical quite tumour among Indians. The International Association of Cancer Registries (IARC) reported that there are over 28,000 cases of brain tumours reported in India annually and over 24,000 people reportedly die because of brain tumours annually. A brain tumours may be a serious condition and might be fatal if not detected early and treated. Typically, visual examination and manual techniques are used for brain tumor diagnosis. This manual interpretation of medical images demands time consumption and is extremely at risk of mistakes. Thus, during this project we apply deep learning algorithms for effective diagnosis of brain tumor without paying a visit to the doctor. This results in earlier determination of the presence of the disease and allows us to require prior actions immediately to avoid further consequences in a good and cheap manner avoiding any human error. Online application employing a javascript framework react JS are going to be developed as an hospital application where an input x-ray image of the brain of a patient are uploaded to perform diagnosis for tumor.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	ix
	LIST OF ABBREVIATIONS	x
1	INTRODUCTION	1
1.1	GENERAL	1
1.2	TECHNOLOGIES USED	2
1.2.1	Deep learning	2
1.2.2	Deep Learning Process	4
1.2.3	Classification of Neural Networks	4
1.2.4	Feed-Forward Neural Networks	5
1.2.5	Recurrent neural networks (RNNs)	5
1.2.6	Common uses of RNN	6
1.2.7	Convolutional Neural Networks (CNN)	7
1.2.8	ReactJS Web Application Development	8
1.3	OBJECTIVE	9
2	REVIEW OF LITERATURE	10
2.1	INTRODUCTION	10
2.2	LITERATURE SURVEY	10
3	SYSTEM ANALYSIS	19
3.1	EXISTING SYSTEM	19
3.1.1	Disadvantages of Existing System	19
3.2	PROPOSED SYSTEM	20
3.3	ADVANTAGES	20
3.3.1	Applications	21

CHAPTER NO.	TITLE	PAGE NO.
4	SYSTEM DESIGN	22
4.1	DETAILED DESIGN OF THE PROJECT	22
4.2	SYSTEM ARCHITECTURE	22
4.2.1	Working	23
4.2.2	Module description	23
4.2.3	Data Collection	24
4.2.4	Scraping from the Net	24
4.2.5	Third-Party	25
4.2.6	Data Augmentation	25
4.3	BRAIN TUMOUR PREDICTION USING THE ARCHITECTURE	27
4.3.1	Global Average Pooling	28
4.3.2	MongoDb integration	29
4.3.3	Database	29
4.3.4	Collection	29
4.3.5	Document	29
4.4	WEB APPLICATION DEVELOPMENT	30
4.4.1	JSX	30
4.4.2	Components	31
4.4.3	Unidirectional data flow and Flux	31
4.5	ADVANTAGES OF REACT JS	31
5	SOFTWARE DESCRIPTION	32
5.1	SOFTWARE LIST	32
5.2	VISUAL STUDIO	32
5.2.1	Python	34
6	RESULTS AND DISCUSSION	35
6.1	FINAL RESULTS OBTAINED	35
CHAPTER NO.	TITLE	PAGE NO.

7	CONCLUSION AND FUTURE WORK	42
7.1	CONCLUSION	52
7.2	FUTURE WORK	52
	APPENDIX	43
	REFERENCES	49

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
1.1	Deep Learning Layers	3
1.2	Deep Learning Process	4
1.3	Types of Deep Learning Networks	5
1.4	CNN	7
1.5	Types of CNN	8
1.6	ReactJS architecture	9
4.1	System Architecture	22
4.2	Data augmentation	27
4.3	Googlenet architecture	28
4.4	Architecture diagram of mongoDB	30
6.1	Dataset Collected	35
6.2	Image Before Augmentation	36
6.3	Image After Augmentation	36
6.4	Moving Inside the coding folder	36
6.5	Executing In the Command Prompt	37
6.6	Training before fine tuning	37
6.7	Training after fine tuning	38
6.8	Training loss graph	38
6.9	Brain tumour login page	39
6.10	Home page	39
6.11	Brain tumour x-ray upload	40
6.12	Normal brain scan image	40
6.13	Brain tumour image	41

LIST OF SYMBOLS AND ABBREVIATIONS

AC	-	Alternate Current
A	-	Amplifier
BTC	-	Brain Tumor Classification
CAD	-	Computer-Aided Detection
CNN	-	Convolutional Neural Networks
DC	-	Direct Current
ERCC1	-	Expression of the excision Repair Cross-Complementing 1
GCN	-	Graph Convolutional Network
ILSVRC	-	ImageNet Large Scale Visual Recognition Challenge
IIoT	-	Industry Internet of Things
IoT	-	Internet of Things
LSTM	-	Long Short-Term Memory
MAP	-	Mean Arterial Pressure
NSCLC	-	Non-Small-Cell Lung Cancer
PPI	-	Protein-Protein Interactions
RF	-	Radio Frequency
RNN	-	Recurrent Neural Networks
SNF	-	Similar Network Fusion
TeUS	-	Temporal Enhanced Ultrasound
V	-	Voltage

CHAPTER 1

INTRODUCTION

1.1 GENERAL

Uncontrolled and abnormal cell growth within the brain is named a neoplasm. The space in our skull is restricted. Therefore, this extra growth inside our brain causes more pressure inside the skull, causing life-threatening complications, and also damaging our brain. Tumours may be either benign or malignant. The benign ones aren't cancerous and can't spread to other parts of the brain or body. The malignant ones are cancerous, grow uncontrollably and might spread to other parts of the body. Brain cancer or tumors is an abnormal growth of cells within the brain. Malignant tumors can grow and spread aggressively to distant parts of the body moreover. Tumors that don't spread or invade nearby tissue are called benign. Benign tumors are less harmful as compared to malignant ones, but a nonmalignant tumour can cause problems within the brain by pressing on a close- by tissue.

Brain tumors that originate in brain cells are called primary brain tumors. the foremost common primary brain tumors are gliomas, meningiomas, pituitary adenomas, vestibular schwannomas, and primitive neuroectodermal tumors (medulloblastomas). The term glioma includes glioblastomas, astrocytomas, oligodendrogliomas, and ependymomas. Metastatic or secondary brain tumors spread to the brain from other tumors. Symptoms of a tumor are usually associated with its location instead of its size. Symptoms develop when the tumor destroys or compresses normal brain tissue. Either the

tissues round the tumor swell or the tumor interferes with the conventional flow of fluid round the brain and medulla spinalis.

Diagnosis of Brain cancer may be done by Physical examination, MRI or CT scan, Biopsy through surgery or Stereotactic brain biopsy. Treatment of brain cancer is typically complex. the foremost widely used treatments are surgery, irradiation, and chemotherapy. many of us with brain tumors undergo surgery or a stereotactic operation within which the tumor is removed with the help of image guidance, leaving the healthy brain relatively intact. Neuroendoscopy is another minimally-invasive surgical treatment where the tumor is removed through small holes within the skull, mouth or nose, enabling neurosurgeons to access areas of the brain that can't be reached with traditional surgery.

Radiation therapy and chemotherapy are other modalities of treatment accustomed treat brain cancers. The treatment of brain tumours could be a complex one as tumours are positioned near vital structures which, if affected during treatment can largely impact the patient post treatment.

1.2 TECHNOLOGIES USED

1.2.1 Deep learning

Deep learning could be a computer software that mimics the network of neurons in a very brain. it's a subset of machine learning and is named deep learning because it makes use of deep neural networks.

Deep learning algorithms are constructed with connected layers.

- The first layer is named the Input Layer
- The last layer is named the Output Layer

- All layers in between are called Hidden Layers. The word deep means the network join neurons in additional than two layers.

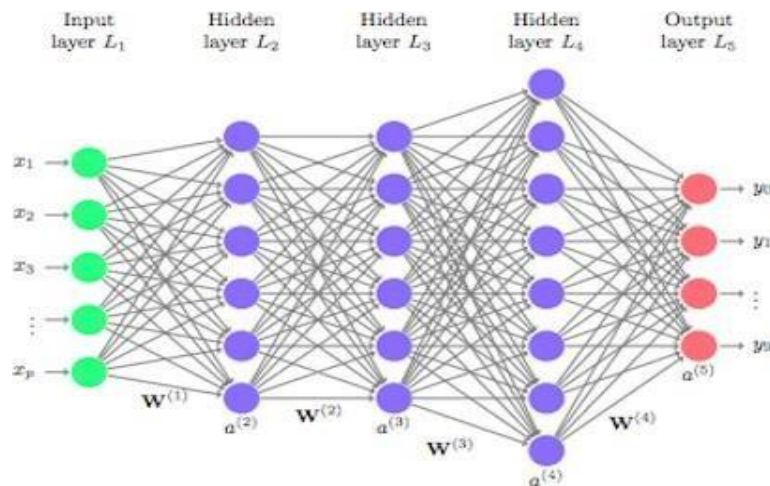


Figure 1.1 Deep Learning Layers

Each Hidden layer consists of neurons. The neurons are connected to every other. The neuron will process so propagate the signal it receives the layer above it. The strength of the signal given the neuron within the next layer depends on the burden, bias and activation function.

The network consumes large amounts of computer file and operates them through multiple layers; the network can learn increasingly complex features of the information at each layer.

Deep learning may be a powerful tool to form prediction an actionable result. Deep learning excels in pattern discovery (unsupervised learning) and knowledge-based prediction. Big data is that the fuel for deep learning. When both are combined, a company can reap unprecedented ends up in term of productivity, sales, management, and innovation.

Deep learning can outperform traditional method. for example, deep learning algorithms are 41% more accurate than machine learning algorithm

in image classification, 27 yet one more accurate in biometric identification and 25% in voice recognition.

1.2.2 Deep Learning Process

A deep neural network provides state-of-the-art accuracy in many tasks, from object detection to speech recognition. they go to be told automatically, without predefined knowledge explicitly coded by the programmers.

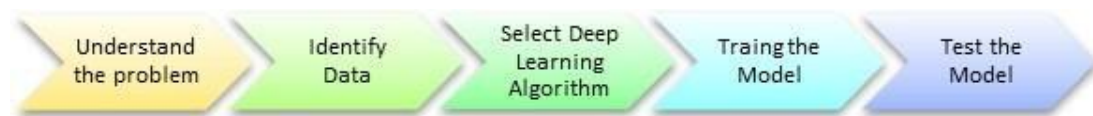


Figure 1.2 Deep Learning Process

Each layer represents a deeper level of data, i.e., the hierarchy of knowledge. A neural network with four layers will learn more complex feature than thereupon with two layers.

The learning occurs in two phases.

- The first phase consists of applying a nonlinear transformation of the inputand make a statistical model as output.
- The second phase aims at improving the model with a mathematical method named as derivative.

The neural network repeats these two phases hundreds to thousands of some time until it's reached a tolerable level of accuracy. The repeat of this two-phase is termed an iteration.

1.2.3 Classification of Neural Networks

Shallow neural network: The Shallow neural network has only 1 hidden layerbetween the input and output.

Deep neural network: Deep neural networks have quite one layer. as an example, Google Net model for image recognition counts 22 layers.

Nowadays, deep learning is employed in many ways sort of a driverless car, mobile, Google program, Fraud detection, TV, and so on

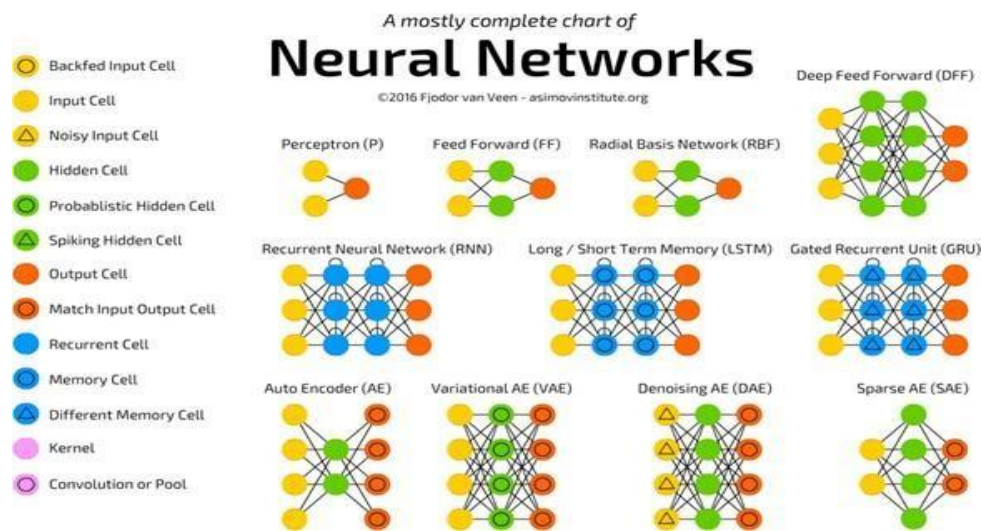


Figure 1.3 Types of Deep Learning Networks

1.2.4 Feed-Forward Neural Networks

The simplest type of artificial neural network. With this type of architecture, information flows in only one direction, forward. It means, the information's flows starts at the input layer, goes to the "hidden" layers, and end at the output layer. The network does not have a loop. Information stops at the output layers.

1.2.5 Recurrent neural networks (RNNs)

RNN may be a multi-layered neural network that may store information in context nodes, allowing it to be told data sequences and output variety or another sequence. In simple words it a man-made neural networks whose connections between neurons include loops. RNNs are similar temperament for processing sequences of inputs.

The process of RNNs are explained below:

- The RNN neurons will receive a symbol that time to the beginning of the sentence.
- The network receives the word "Do" as an input and produces a vector of the quantity. This vector is fed back to the neuron to supply a memory to the network. This stage helps the network to recollect it received "Do" and it received it within the first position.
- The network will similarly proceed to the subsequent words. It takes the word "you" and "want." The state of the neurons is updated upon receiving each word.
- The finish occurs after receiving the word "a." The neural network will provide a probability for every English word which will be accustomed complete the sentence. A well-trained RNN probably assigns a high probability

1.2.6 Common uses of RNN

- Help securities traders to generate analytic reports
- Detect abnormalities in the contract of financial statement
- Detect fraudulent credit-card transaction
- Provide a caption for images
- Power chatbots
- The standard uses of RNN occur when the practitioners are working with time-series data or sequences.

1.2.7 Convolutional Neural Networks (CNN)

CNN may be a multi-layered neural network with a novel architecture designed to extract increasingly complex features of the information at each layer to see the output. CNN's are similar temperament for perceptual tasks.

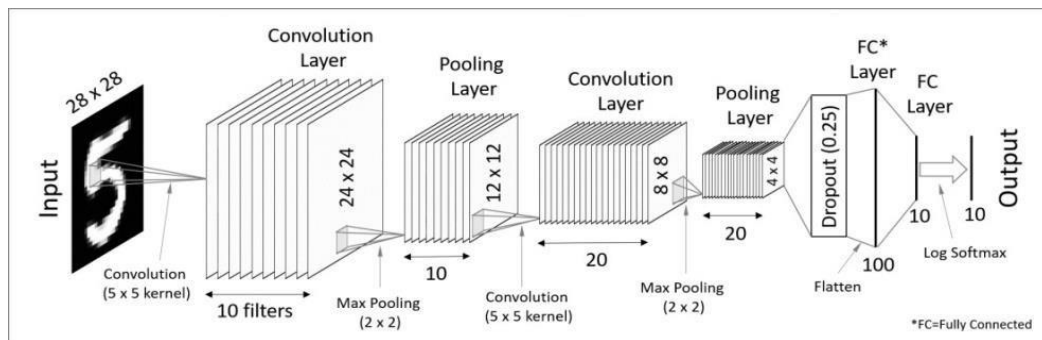


Figure 1.4 CNN

CNN is generally used when there's an unstructured data set (e.g., images) and also the practitioners must extract information from it.

For instance, if the task is to predict a picture caption:

- For example, a picture of a cat is given as input, this image, in computer terms, could be a collection of the pixels. Generally, one layer for the grayscale picture and three layers for a color picture.
- During the feature learning (i.e., hidden layers), the network will identify unique features, as an example, the tail of the cat, the ear, etc.
- When the network thoroughly learned the way to recognize an image, it can provide a probability for every image it knows. The label with the best probability will become the prediction of the network.

A Convolutional Neural Network (CNN, or ConvNet) are a special reasonably multi-layer neural networks, designed to acknowledge visual patterns directly from pixel images with minimal Pre-processing. The ImageNet project could be a large visual database designed to be used in visual seeing software research. The ImageNet project runs an annual software contest, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), where software programs compete to properly classify and detect objects and scenes.

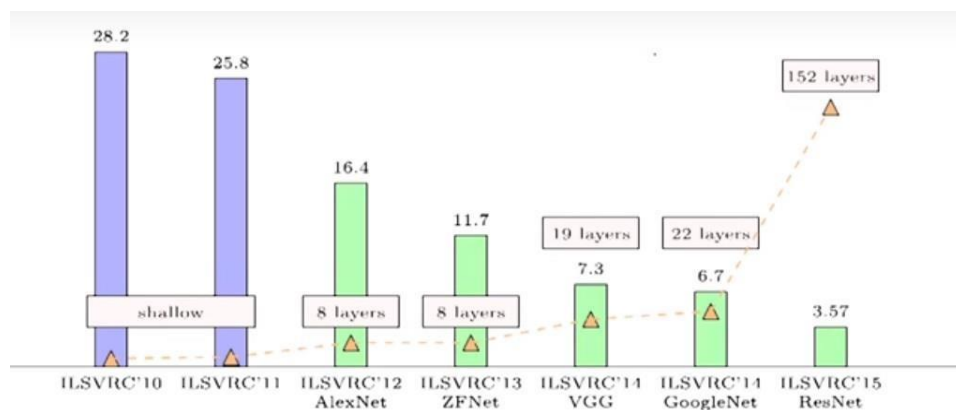


Figure 1.5 Types of CNN

1.2.8 ReactJS Web Application Development

ReactJS basically is an open-source JavaScript library which is employed for building user interfaces specifically for single page applications. It's used for handling view layer for web and mobile apps. React also allows us to make reusable UI components. React was first created by Jordan Walke, a applied scientist working for Facebook. React first deployed on Facebook's newsfeed in 2011 and on Instagram.com in 2012.

React allows developers to form large web applications which may change data, without reloading the page. the most purpose of React is to be fast, scalable, and simple. It works only on user interfaces in application. This corresponds to look at within the MVC template. It is used with a mix of other JavaScript libraries or frameworks, like Angular JS in MVC.

Features

- **JSX:** JSX stands for JavaScript XML. Its an XML/ HTML like syntax employed by React. It extends the ECMAScript so XML/ HTML like text can co-exist together with JavaScript react code. This syntax is employed by the pre- processors like Babel to rework HTML like text found in JavaScript files into standard JavaScript objects. With JSX, we will go a step further by again embedding the HTML code inside the JavaScript. This makes HTML codes easy to know and boosts JavaScript"s performance while making our application robust.

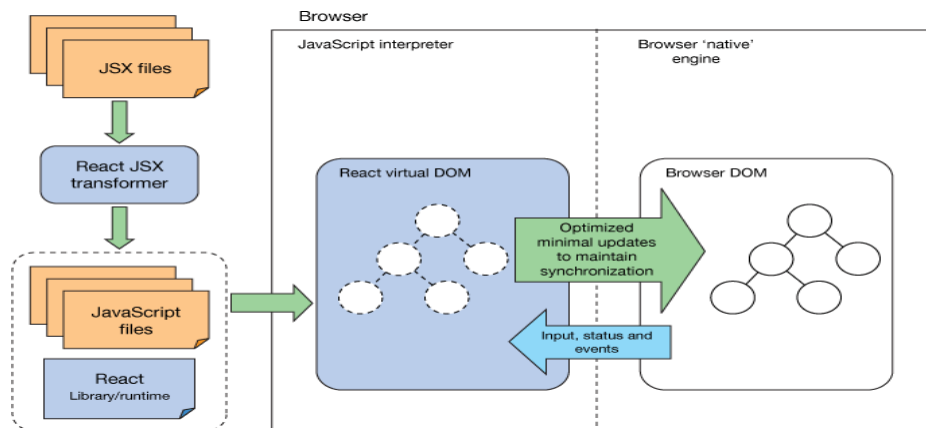


Figure 1.6 ReactJS architecture

1.3 OBJECTIVE

- To determine the presence of cancer at an early stage
- Use CT images or x-ray images for the detection of cancer.
- To reduce the time required to detect the presence of cancer.
- To determine the cheap way to predict the cancer presence.
- To predict five different types of cancer.

CHAPTER 2

REVIEW OF LITERATURE

2.1 INTRODUCTION

The following shows survey did for Brain cancer prediction System. The mostpopular of the existing techniques is been discussed as follows.

2.2 LITERATURE SURVEY

Brain tumor is one of the most dangerous cancers in people of all ages, and its grade recognition is a challenging problem for radiologists in health monitoring and automated diagnosis. Recently, numerous methods based on deep learning have been presented in the literature for brain tumor classification (BTC) in order to assist radiologists for a better diagnostic analysis. In this overview, we present an in-depth review of the surveys published so far and recent deep learning-based methods for BTC. Our survey covers the main steps of deep learning-based BTC methods, including preprocessing, features extraction, and classification, along with their achievements and limitations. We also investigate the state-of-the-art convolutional neural network models for BTC by performing extensive experiments using transfer learning with and without data augmentation. Furthermore, this overview describes available benchmark data sets used for the evaluation of BTC. Finally, this survey does not only look into the past literature on the topic but also steps on it to delve into the future of this area and enumerates some research directions that should be followed in the future, especially for personalized and smart healthcare.

New, stable, and broadband skin-equivalent semisolid phantoms for mimicking interactions of millimeter waves with the human skin and skin tumors. Realistic skin phantoms serve as an invaluable tool for exploring the feasibility of new technologies and improving design concepts related to millimeter-wave skin cancer detection methods. Normal and malignant skin tissues are separately mimicked by using appropriate mixtures of deionized water, oil, gelatin powder, formaldehyde, TX-150 (a gelling agent, widely referred to as „super stuff“), and detergent. The dielectric properties of the phantoms are characterized over the frequency band of 0.5-50 GHz using a slim-form open-ended coaxial probe in conjunction with a millimeter-wave vector network analyzer. The measured permittivity results show excellent match with ex-vivo, fresh skin (both normal and malignant) permittivities determined in our prior work over the entire frequency range.

We introduce a new 3D CNN for computer-aided cancer detection in ABUS volumes. We believe we are the very first to utilize deep learning methods for this challenge. For our network, a new threshold map is devised to render voxel level threshold to categorize cancer voxels from normal tissue regions, therefore getting low false positives. Additionally, a densely deep supervision is utilized to augment the sensitivity largely by effectively utilizing multi-layer discriminative features. The effectiveness of our network is verified on a collected dataset of 219 patients, with 614 ABUS volumes, including 745 cancer regions; and 144 healthy women with 900 volumes, without abnormal findings. Experiments demonstrate our network outperforms the most up-to-date models, obtaining a sensitivity of 95% with 0.84 false positives per volume. Our technique provides an automatic and accurate CADe scheme for breast screening by having high sensitivity and low false positives, and may assist other advanced applications, such as size measurement, lesion characterization, and BI-RADS categorization.

Comparison of phantom and patient data between a conventional clinical PET- CT scanner and a dedicated high resolution brain scanner, the HRRT, has shown that there is benefit of high resolution scanners to reduce the partial volume effect and spill-in from surrounding tissues for small structures and lesions. However, this comes with the consequence of higher voxel noise. RM image reconstruction to restore PET- CT can provide some benefit but is unable to achieve the resolution performance of the HRRT, and has the potential for introducing image artifacts.

An exhaustive empirical study to identify biomarkers using two approaches: frequency-based and network-based, over seventeen different bi-clustering algorithms and six different cancer expression datasets. To systematically analyze the bi-clustering algorithms, we perform enrichment analysis, subtype identification and biomarker identification. Bi-clustering algorithms such as C&C, SAMBA and Plaid are useful to detect biomarkers by both approaches for all datasets except prostate cancer. We detect a total of 102 gene biomarkers using frequency-based method out of which 19 are for blood cancer, 36 for lung cancer, 25 for colon cancer, 13 for multi-tissue cancer and 9 for prostate cancer. Using the network-based approach we detect a total of 41 gene biomarkers of which 15 are from blood cancer, 12 from lung cancer, 6 from colon cancer, 7 from multi-tissue cancer and 1 from prostate cancer dataset.

Temporal Enhanced Ultrasound (TeUS), comprising the analysis of variations in backscattered signals from a tissue over a sequence of ultrasound frames, has been previously proposed as a new paradigm for tissue characterization. In this manuscript, we propose to use deep Recurrent Neural Networks (RNN) to explicitly model the temporal information in TeUS. By investigating several RNN models, we demonstrate that Long Short-Term Memory (LSTM) networks achieve the highest accuracy in separating cancer from benign tissue in the prostate. We also present algorithms for in-depth analysis of LSTM networks. Our in vivo study includes data from 255 prostate biopsy cores of 157 patients. We

achieve area under the curve, sensitivity, specificity, and accuracy of 0.96, 0.76, 0.98 and 0.93, respectively. Our result suggests that temporal modeling of TeUS using RNN can significantly improve cancer detection accuracy over previously presented works.

Pros: This technique achieve area under the curve, sensitivity, specificity, and accuracy of 0.96, 0.76, 0.98 and 0.93, respectively.

Cons: LSTM networks used in this method require 4 linear layer (MLP layer) per cell to run at and for each sequence time-step. Linear layers require large amounts of memory bandwidth to be computed, in fact they cannot use many compute unit often because the system has not enough memory bandwidth to feed the computational units.

An exact functional test for directional association by examining the strength of functional dependency. It is effective in promoting functional patterns by reducing statistical power on non-functional patterns. We designed an algorithm to carry out the test using a fast branch-and-bound strategy, which achieved a substantial speedup over brute-force enumeration. On data from an epidemiological study of liver cancer, the test identified the hepatitis status of a subject as the most influential risk factor among others for the cancer phenotype. On human lung cancer transcriptome data, the test selected 1068 transcription start sites of putative noncoding RNAs directionally associated with lung cancers, stronger than 95% TSSs of 694 curated cancer genes. These predictions include non-monotonic interaction patterns, to which other routine tests were insensitive. Complementing symmetric (non-directional) association methods such as Fisher's exact test, the exact functional test is a unique exact statistical test for evaluating evidence for causal relationships.

A new breast cancer detection method is presented that combines thermography and high frequency excitation techniques. The proposed method uses distribution and variation of the temperature on the breast

surface in order to estimate the location and size of a malignant tissue (a cancerous tumor). The idea of this work stems from the analysis of the radio frequency (RF) radiation effects on human body. In this paper, a description of RF effects on human body and simulation results are provided. In order to validate the proposed method, a multilayer 3D breast model is simulated. First, breast tissue is excited with different types of RF antennas following which an electromagnetic analysis is conducted. Next, the heat transfer equation is applied for thermal analysis to estimate the distribution of surface temperature.

A novel Computer-Aided Detection (CAD) system for pulmonary nodules using multi-view convolutional networks (ConvNets), for which discriminative features are automatically learnt from the training data. The network is fed with nodule candidates obtained by combining three candidate detectors specifically designed for solid, sub solid, and large nodules. For each candidate, a set of 2-D patches from differently oriented planes is extracted. The proposed architecture comprises multiple streams of 2-D ConvNets, for which the outputs are combined using a dedicated fusion method to get the final classification. Data augmentation and dropout are applied to avoid overfitting. On 888 scans of the publicly available LIDC-IDRI dataset, our method reaches high detection sensitivities of 85.4% and 90.1% at 1 and 4 false positives per scan, respectively. An additional evaluation on independent datasets from the ANODE09 challenge and DLCST is performed.

Smoking is the major cause of lung cancer and the leading cause of cancer-related death in the world. The most current view about lung cancer is no longer limited to individual genes being mutated by any carcinogenic insults from smoking. Instead, tumorigenesis is a phenotype conferred by many systematic and global alterations, leading to extensive heterogeneity and variation for both the genotypes and phenotypes of individual cancer cells. Thus, strategically it is foremost important to develop a methodology to capture any consistent and global alterations presumably shared by

most of the cancerous cells for a given population. This is particularly true that almost all of the data collected from solid cancers (including lung cancers) are usually distant apart over a large span of temporal or even spatial contexts. Here we report a multiple non-Gaussian graphical model to reconstruct the gene interaction network using two previously published gene expression datasets. Our graphical model aims to selectively detect gross structural changes at the level of gene interaction networks. Our methodology is extensively validated, demonstrating good robustness, as well as the selectivity and specificity expected based on our biological insights. In summary, gene regulatory networks are still relatively stable during presumably the early stage of neoplastic transformation. But drastic structural differences can be found between lung cancer and its normal control, including the gain of functional modules for cellular proliferations such as EGFR and PDGFRA, as well as the lost of the important IL6 module, supporting their roles as potential drug targets. Interestingly, our method can also detect early modular changes, with the ALDH3A1 and its associated interactions being strongly implicated as a potential early marker, whose activations appear to alter LCN2 module as well as its interactions with the important TP53-MDM2 circuitry. Our strategy using the graphical model to reconstruct gene interaction work with biologically-inspired constraints exemplifies the importance and beauty of biology in developing any bio-computational approach.

We have demonstrated the potential of e-nose technology to distinguish lung cancer patients from matched high-risk smokers, adding to the evidence that measurements of exhaled VOCs (as measured by an e-nose) can be used as a lung cancer screening tool. Smell-prints of high-risk smokers were significantly distinct from those diagnosed with lung cancer and these differences seem to depend to some degree on subject sex and smoking status. Further measurements on multiple devices can be demonstrated to be repeatable and reproducible.

In order to effectively integrate multiple genomic data (including genetic expression, copy number alteration, DNA methylation and exon expression) and clinical data and apply them to predictive studies on cancer survival, similar network fusion algorithm (SNF) was proposed in this paper to integrate multiple genomic data and clinical data so as to generate sample similarity matrix, min redundancy and max- relevance algorithm (mRMR) was used to conduct feature selection of multiple genomic data and clinical data of cancer samples and generate sample feature matrix, and finally two matrixes were used for semi-supervised training through graph convolutional network (GCN) so as to obtain a cancer survival prediction method integrating multiple genomic data and clinical data based on graph convolutional network (GCGCN). Performance indexes of GCGCN model indicate that both multiple genomic data and clinical data play significant roles in the accurate survival time prediction of cancer patients. It is compared with existing survival prediction methods, and results show that cancer survival prediction method GCGCN which integrates multiple genomic data and clinical data has obviously superior prediction effect than existing survival prediction methods. All study results in this paper have verified effectiveness and superiority of GCGCN in the aspect of cancer survival prediction.

This study aims to develop a new quantitative image feature analysis scheme and investigate its role along with 2 genomic biomarkers namely, protein expression of the excision repair cross-complementing 1 (ERCC1) genes and a regulatory subunit of ribonucleotide reductase (RRM1), in predicting cancer recurrence risk of Stage I non-small-cell lung cancer (NSCLC) patients after surgery. By using chest computed tomography images, we developed a computer-aided detection scheme to segment lung tumors and computed tumor related image features. After feature selection, we trained a Naïve Bayesian network based classifier using 8 image features and a Multilayer Perceptron classifier using 2 genomic biomarkers to predict cancer recurrence risk, respectively. Two classifiers were trained

and tested using a dataset with 79 Stage I NSCLC cases, a synthetic minority oversampling technique and a leave-one-case out validation method. A fusion method was also applied to combine prediction scores of two classifiers. AUC (areas under ROC curves) values are 0.78 ± 0.06 and 0.68 ± 0.07 when using the image feature and genomic biomarker based classifiers, respectively. AUC value significantly increased to 0.84 ± 0.05 ($p<0.05$) when fusion of two classifier-generated prediction scores using an equal weighting factor. A quantitative image feature based classifier yielded significantly higher discriminatory power than a genomic biomarker based classifier in predicting cancer recurrence risk. Fusion of prediction scores generated by the two classifiers further improved prediction performance. We demonstrated a new approach that has potential to assist clinicians in more effectively managing Stage I NSCLC patients to reduce cancer recurrence risk.

In this experiment, a gene selection technique was proposed to select a robust gene signature from microarray data for prediction of breast cancer recurrence. In this regard, a hybrid scoring criterion was designed as linear combinations of the scores that were determined in the mutual information (MI) domain and protein-protein interactions (PPI) network. Whereas, the MI-based score represents the complementary information between the selected genes for outcome prediction; and the number of connections in the PPI network between the selected genes builds the PPI-based score. All genes were scored by using the proposed function in a hybrid forward-backward gene-set selection process to select the optimum biomarker-set from the gene expression microarray data. The accuracy and stability of the finally selected biomarkers were evaluated by using five-fold cross-validation to classify available data on breast cancer patients into two cohorts of poor and good prognosis. The results showed an appealing improvement in the cross-dataset accuracy in comparison with similar studies whenever we applied a primary signature, which was selected from one dataset, to predict survival in other independent datasets. Moreover, the proposed method

demonstrated 58-92% overlap between 50-genes signatures, which were selected from seven independent datasets individually.

A framework for design of personalized cancer therapy requires the ability to predict the sensitivity of a tumour to anticancer drugs. The predictive modelling of tumour sensitivity to anti-cancer drugs has primarily focused on generating functions that map gene expressions and genetic mutation profiles to drug sensitivity. In this paper, we present a new approach for drug sensitivity prediction and combination therapy design based on integrated functional and genomic characterizations. The modelling approach when applied to data from the Cancer Cell Line encyclopedia shows a significant gain in prediction accuracy as compared to elastic net and random forest techniques based on genomic characterizations. Utilizing a Mouse Embryonal Rhabdomyosarcoma cell culture and a drug screen of 60 targeted drugs, we show that predictive modelling based on functional data alone can also produce high accuracy predictions. The framework also allows us to generate personalized tumour proliferation circuits to gain further insights on the individualized biological pathway

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Brain tumor is one in all the foremost dangerous cancers in people of all ages, and its grade recognition may be a challenging problem for radiologists in health monitoring and automatic diagnosis. Recently, numerous methods supported deep learning are presented within the literature for neoplasm classification (BTC) so on help radiologists for a stronger diagnostic analysis. during this overview, we present an in-depth review of the surveys published up to now and up to the present point deep learning-based methods for BTC. Our survey covers the foremost steps of deep learning-based BTC methods, including preprocessing, features extraction, and classification, together with their achievements and limitations. We also investigate the state-of-the-art convolutional neural network models for BTC by performing extensive experiments using transfer learning with and without data augmentation. Furthermore, this overview describes available benchmark data sets used for the evaluation of BTC. Finally, this survey doesn't only cross-check the past literature on the subject but also steps thereon to delve into the long term of this area and enumerates some research directions that ought to be followed within the long term, especially for personalized and smart healthcare..

3.1.1 Disadvantages of Existing System

- The existing system only presents a survey
- Does not have a real time implementation
- Is not applicable for extensive usage..

3.2 PROPOSED SYSTEM

The occurrence of brain tumours in India is steadily rising. More and more cases of brain tumours are reported annually within the country among people of various age groups. In 2018, brain tumours was ranked because the 10th most typical reasonably tumour among Indians. The International Association of Cancer Registries (IARC) reported that there are over 28,000 cases of brain tumours reported in India annually and quite 24,000 people reportedly die because of brain tumours annually. A brain tumours could be a serious condition and may be fatal if not detected early and treated. Typically, visual examination and manual techniques are used for neoplasm diagnosis. This manual interpretation of medical images demands time consumption and is very at risk of mistakes. Thus, during this project we apply deep learning algorithms for effective diagnosis of tumor without paying a visit to the doctor. This ends up in earlier determination of the presence of the disease and allows us to require prior actions immediately to avoid further consequences in a good and cheap manner avoiding any human error. an internet application employing a javascript framework react JS are developed as an hospital application where an input x-ray image of the brain of a patient are uploaded to perform diagnosis for brain tumor..

3.3 ADVANTAGES

- Accuracy of the prediction will be increased by using different efficient techniques and algorithms.
- Saves time and money required for consultation of doctor.
- Helps people in early prediction of the brain tumour and can save their life.
- Hospital web application.

3.3.1 Applications

- Can be Used in hospitals.
- Can be used in cancer test centres.
- Early prediction of cancer.

CHAPTER 4

SYSTEM DESIGN

4.1 DETAILED DESIGN OF THE PROJECT

This chapter describes the overall and the detailed architectural design. It also describes each module that is to be implemented along with Data Flow diagram.

4.2 SYSTEM ARCHITECTURE

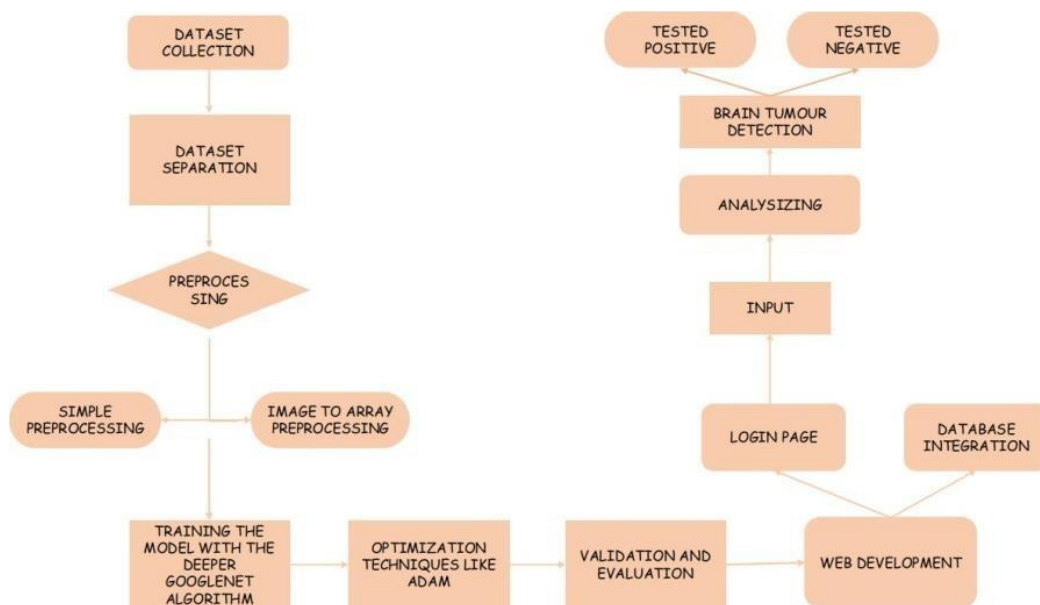


Figure: 4.1: System Architecture

4.2.1 Working

In this project, we are going to determine presence of brain tumour. By this project, we can able to determine the presence of brain tumour in the CT images or x- rays. So, initially the first step will be dataset collection where we will be collecting dataset such as CT images or x-rays which are used by the laboratories to perform analysis of the presence of brain tumor from various resources through internet. After that, we will be splitting those datasets into different categories that is we will be splitting the dataset into training and testing dataset. In training datasets, we will be using the dataset for training the module whereas testing dataset is used to evaluate the model when it is been completely ready. So training dataset first undergoes the process called dataset augmentation, where the dataset is multiplied into many datasets then it will undergo the process called preprocessing, which is to make all sizes into single size. We train that datasets by extracting the features using deeper googlenet algorithm. It undergoes a process called optimization which will optimize the model and loss minimization which will reduce the noises generated during training. In the last it will be undergoing a process is called model seriation which will be evaluated after generating model using the testing dataset and predict the presence of cancer. A web application using a javascript framework reactJS will be developed where an input x-ray image will be given to predict and detect the presence of brain tumour. Thus, this method provides an effective and cheap method to determine the presence of brain tumor than the methodologies used nowadays.

4.2.2 Module description

- Dataset Collection
- Dataset Augmentation
- Brain tumour prediction using the architecture

- MongoDB integration
- Web Application Development

4.2.3 Data Collection

A data set may be a collection of information. Deep Learning has become the go-to method for solving many challenging real-world problems. It's definitely far and away the simplest performing method for computer vision tasks. The image above showcases the facility of deep learning for computer vision. With enough training, a deep network can segment and identify the "key points" of each person within the image. These deep learning machines that are working so well need fuel plenty of fuel; that fuel is data. The more labelled data available, the higher our model performs. The idea in the previous architecture like AlexNet, the fully connected layers are used at the tip of the network. These fully connected layers contain the bulk of parameters of the many architectures that causes a rise in computation cost. A lot of more data resulting in better performance has even been explored at a large-scale by Google with a dataset of 300 Million images! When deploying a Deep Learning model during a real-world application, data must be constantly fed to continue improving its performance. And, within the deep learning era, data is extremely well arguably the foremost valuable resource. There are three steps of collecting data

4.2.4 Scraping from the Net

Manually finding and downloading images takes a protracted time simply thanks to the quantity of human work involved. The task probably has some reasonably common objects are to be detected. then that becomes the keyword for web- scraping. It also becomes the category name for that object. Every single pixel within the image is required. to urge those, it's best to use some really great image annotation tools that are already out there. Given a rough set of polygon points around an object, can generate the pixel labels for segmentation. Deep extreme cut is additionally quite similar

except they use only the four extreme points round the object. this can then give some nice bounding box and segmentation labels. an alternative choice is to use an existing image annotation GUIs.

4.2.5 *Third-Party*

Since data has become such a valuable commodity within the deep learning era, many start-ups have began to offer their own image annotation services they'll gather and label the info. Given an outline of what quite data and annotations needed. Mighty is one that has been doing self-driving car image annotation and has become pretty big within the space were at CVPR 2018 too. Payment AI are less specialized than Mighty AI, offering image annotation for any domain.

4.2.6 *Data Augmentation*

The performance of deep learning neural networks often improves with the amount of data available.

Data augmentation could be a technique to artificially create new training data from existing training data. this can be done by applying domain-specific techniques to examples from the training data that make new and different training examples.

Image data augmentation is probably the foremost well-known kind of data augmentation and involves creating transformed versions of images within the training dataset that belong to the identical class because the original image.

Transforms include a spread of operations from the sector of image manipulation, like shifts, flips, zooms, and far more.

The intent is to expand the training dataset with new, plausible examples. This means, variations of the training set images that are likely to be seen

by the model. For instance, a horizontal flip of an image of a cat may be, because the photo could be taken from the left or right. A vertical flip of the photo of a cat doesn't make sense and would probably not be appropriate only if the model is extremely unlikely to work out a photograph of an upside-down cat.

As such, it's clear that the selection of the particular data augmentation techniques used for a training dataset must be chosen carefully and within the context of the training dataset and knowledge of the matter domain. Additionally, it is useful to experiment with data augmentation methods in isolation and as one to work out if they end in a measurable improvement to model performance, perhaps with a little prototype dataset, model, and training run.

Modern deep learning algorithms, like the convolutional neural network, or CNN, can learn features that are invariant to their location within the image. Nevertheless, augmentation can further aid during this transform invariant approach to learning and might aid the model in learning features that are invariant to transforms like left-to-right to top-to-bottom ordering, light levels in photographs, and more.

Image data augmentation is usually only applied to the training dataset, and not to the validation or test dataset. This is often different from data preparation like image resizing and pixel scaling; they have to be performed consistently across all datasets that interact with the model.

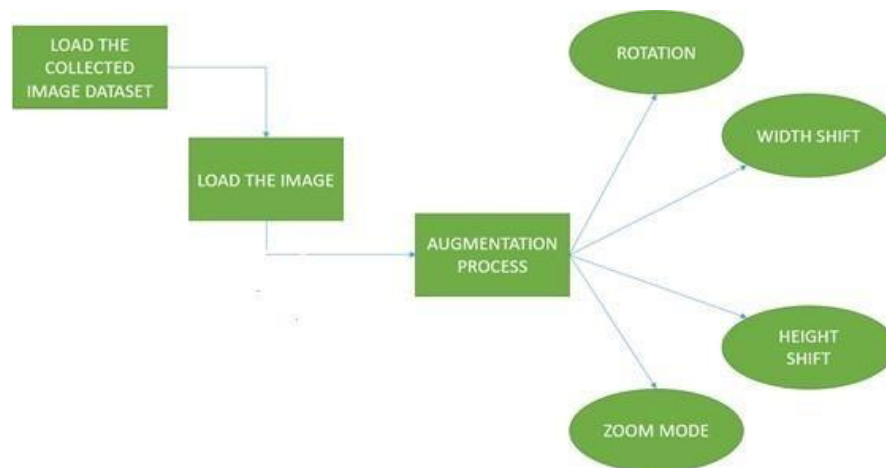


Figure 4.2 Data augmentation

4.3 BRAIN TUMOUR PREDICTION USING THE ARCHITECTURE

In this project, the deeper googlenet algorithm is used for training the brain tumour dataset.

GoogLeNet could be a sort of convolutional neural network supported the Inception architecture. It utilises Inception modules, which permit the network to settle on between multiple convolutional filter sizes in each block. An Inception network stacks these modules on top of every other, with occasional max-pooling layers with stride 2 to halve the resolution of the grid.

The GoogLeNet architecture is incredibly different from previous state-of-the-art architectures like AlexNet and ZF-Net. It uses many various varieties of methods like 1×1 convolution and global average pooling that permits it to form deeper architecture. within the architecture, we'll discuss a number of these methods:

1×1 convolution : The inception architecture uses 1×1 convolution in its architecture. These convolutions accustomed decrease the amount of parameters (weights and biases) of the architecture. By reducing the parameters we also increase the depth of the architecture.

- Let's look at an example of a 1×1 convolution below:
- For Example, If we want to perform 5×5 convolution having 48 filters without using 1×1 convolution as intermediate:
- Total Number of operations : $(14 \times 14 \times 48) \times (5 \times 5 \times 480) = 112.9 \text{ M}$
- With 1×1 convolution :

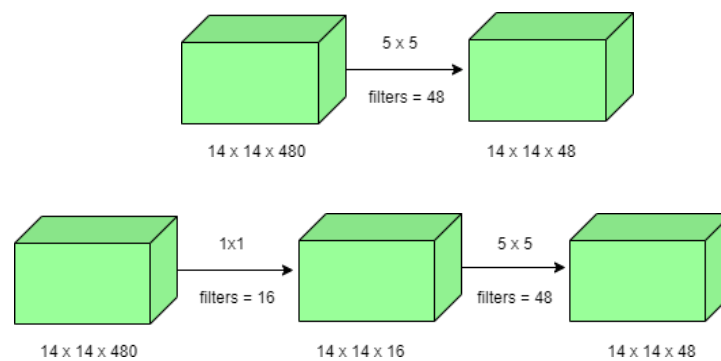


Figure 4.3 : Googlenet architecture

- $(14 \times 14 \times 16) \times (1 \times 1 \times 480) + (14 \times 14 \times 48) \times (5 \times 5 \times 16) = 1.5\text{M} + 3.8\text{M} = 5.3\text{M}$ which is much smaller than 112.9M.

4.3.1 Global Average Pooling

In GoogLeNet architecture, there's a way called global average pooling is employed at the top of the network. This layer takes a feature map of 7×7 and averages it to 1×1 . This also decreases the quantity of trainable parameters to 0 and improves the top-1 accuracy by 0.6%

4.3.2 *MongoDb integration*

In this project, Mongo DB is used for database integration.

MongoDB could be a cross-platform, document-oriented database that has, high performance, high availability, and straightforward scalability. MongoDB works on concept of collection and document.

4.3.3 *Database*

Database may be a physical container for collections. Each database gets its own set of files on the filing system. one MongoDB server typically has multiple databases.

4.3.4 *Collection*

Collection may be a group of MongoDB documents. it's the equivalent of an RDBMS table. a group exists within one database. Collections don't enforce a schema. Documents within a group can have different fields. Typically, all documents during a collection are of comparable or related purpose.

4.3.5 *Document*

A document could be a set of key-value pairs. Documents have dynamic schema. Dynamic schema means documents within the same collection don't have to have the identical set of fields or structure, and customary fields in a very collection's documents may hold differing kinds of knowledge

The following image shows the flow of MongoDB.

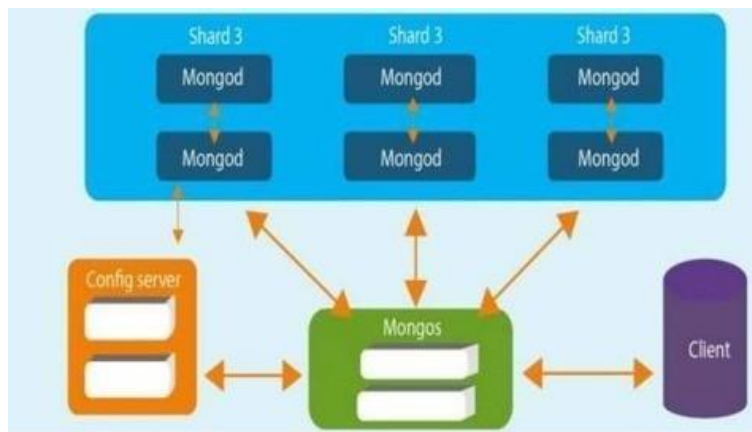


Figure 4.4: Architecture diagram of mongoDB

4.4 WEB APPLICATION DEVELOPMENT

ReactJS is JavaScript library used for building reusable UI components. consistentwith React official documentation, following is that the definition –

React could be a library for building composable user interfaces. It encourages the creation of reusable UI components, which present data that changes over time. plenty of people use React because the V in MVC. React abstracts away the DOM from you, offering a less complicated programming model and better performance. React can even render on the server using Node, and it can power native apps using React Native. React implements one-way reactive data flow, which reduces the boilerplate and is simpler to reason about than traditional data binding.

4.4.1 JSX

JSX stands for JavaScript XML. JSX allows us to write down HTML in React. JSXmakes it easier to put in writing and add HTML in React.

It is faster than normal JavaScript because it performs optimizations while translating to regular JavaScript. It makes easier for us to make templates. rather than separating the mark-up and logic in separated files, React uses

components for this purpose. we'll study components in details in further articles.

4.4.2 Components

Components are independent and reusable bits of code. They serve the identical purpose as JavaScript functions, but add isolation and returns HTML via a render function.

Components are available two types, Class components and performance components, during this tutorial we'll target Class components.

4.4.3 Unidirectional data flow and Flux

React implements one-way data flow which makes it easy to reason about your app. Flux could be a pattern that helps keeping your data unidirectional.

4.5 ADVANTAGES OF REACT JS

- Uses virtual DOM which could be a JavaScript object. This may improve apps performance, since JavaScript virtual DOM is quicker than the regular DOM.
- Can be used on client and server side in addition like other frameworks.
- Component and data patterns improve readability, which helps to keep up larger apps.

CHAPTER 5

SOFTWARE DESCRIPTION

5.1 SOFTWARE LIST

- Microsoft Visual studio
- Python

5.2 VISUAL STUDIO

In this project the Microsoft visual studio is used as an IDE.

Visual Studio Code combines the simplicity of a ASCII document editor with powerful developer tooling, like IntelliSense code completion and debugging.

First and foremost, it's an editor that gets out of your way. The delightfully frictionless edit-build-debug cycle means less time wiggling with your environment, and longer executing on your ideas.

Visual Studio Code supports macOS, Linux, and Windows - so you'll hit the underside running, no matter the platform.

At its heart, Visual Studio Code features a lightning fast ASCII document editor, perfect for day-to-day use. With support for several languages, VS Code helps you be instantly productive with syntax highlighting, bracket-matching, auto-indentation, box-selection, snippets, and more. Intuitive keyboard shortcuts, easy customization and community-contributed keyboard shortcut mappings allow you to navigate your code with ease.

For serious coding, you'll often like tools with more code understanding than simply blocks of text. Visual Studio Code includes built-in support for IntelliSense code completion, rich semantic code understanding and navigation, and code refactoring.

And when the coding gets tough, the tough get debugging. Debugging is usually the one feature that developers miss most in an exceedingly leaner coding experience, so we made it happen. Visual Studio Code includes an interactive debugger, so you'll step through ASCII document, inspect variables, view call stacks, and execute commands within the console.

VS Code also integrates with build and scripting tools to perform common tasks making everyday workflows faster. VS Code has support for Git so you'll work with source control without leaving the editor including viewing pending changes diffs. Customize every feature to your liking and install any number of third-party extensions. While most scenarios work "out of the box" with no configuration, VS Code also grows with you, which we encourage you to optimize your experience to suit your unique needs.

VS Code includes enriched built-in support for Node.js development with JavaScript and TypeScript, powered by the identical underlying technologies that drive Visual Studio. VS Code also includes great tooling for web technologies like JSX/React, HTML, CSS, SCSS, Less, and JSON.

Architecturally, Visual Studio Code combines the foremost effective of web, native, and language-specific technologies. Using Electron, VS Code combines web technologies like JavaScript and Node.js with the speed and adaptableness of native apps. VS Code uses a newer, faster version of the identical industrial-strength HTML-based editor that has powered the "Monaco" cloud editor, Internet Explorer's F12 Tools, and other projects. Additionally, VS Code uses a tools service architecture that enables it to integrate with many of the identical technologies that power Visual Studio,

including Roslyn for .NET, TypeScript, the Visual Studio debugging engine, and more.

Visual Studio Code includes a public extensibility model that lets developers build and use extensions, and richly customize their edit-build-debug experience.

5.2.1 Python

In this project, python is used as the programming language.

In technical terms, Python is an object-oriented, high-level artificial language with integrated dynamic semantics primarily for web and app development. it's extremely attractive within the field of Rapid Application Development because it offers dynamic typing and dynamic binding options.

Python is comparatively simple, so it is easy to find out since it requires a singular syntax that focuses on readability. Developers can read and translate Python code much easier than other languages. In turn, this reduces the value of program maintenance and development because it allows teams to figure collaboratively without significant language and skill barriers.

Additionally, Python supports the employment of modules and packages, which suggests that programs will be designed during a modular style and code are often reused across a range of projects.

One of the foremost promising benefits of Python is that both the quality library and therefore the interpreter are available freed from charge, in both binary and source form. there's no exclusivity either, as Python and every one the mandatory tools are available on all major platforms. Therefore, it's a tasty option for developers who don't need to stress about paying high development costs.

CHAPTER 6

RESULTS AND DISCUSSION

6.1 FINAL RESULTS OBTAINED

Dataset collection involves the process of collecting scanned images of the brain tumour dataset

Various datasets were collected and one example among the collected dataset can be found below

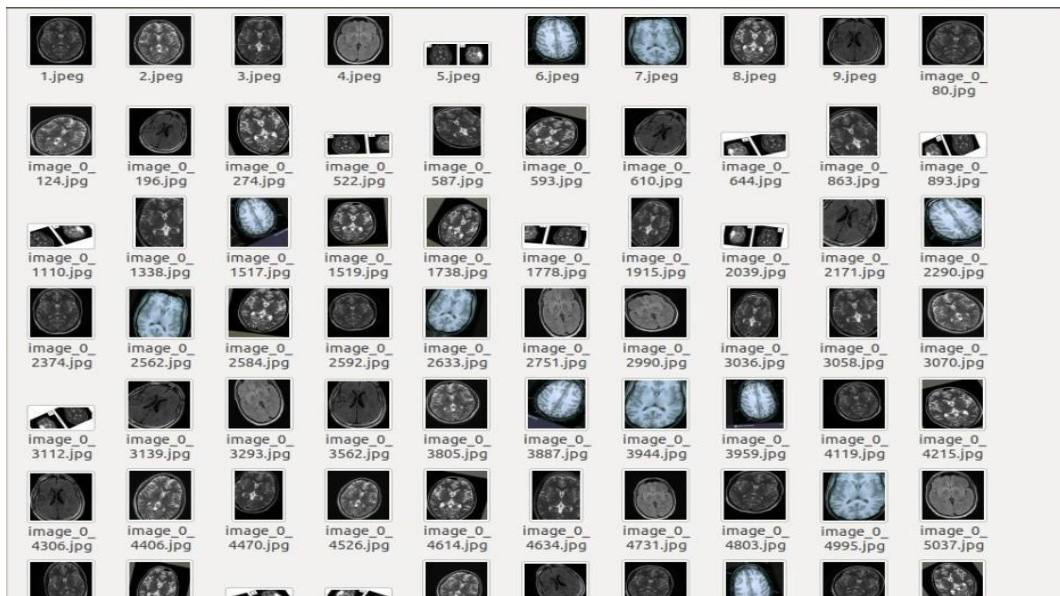


Figure 6.1 Dataset Collected

These datasets are then preprocessed to form an equal aspect ratio so that it can be made ready for training with the model.

The next step is data augmentation which duplicates the collected data into many images so that the prediction percentage can be increased. A sample augmentation of an image can be seen in the below figure.

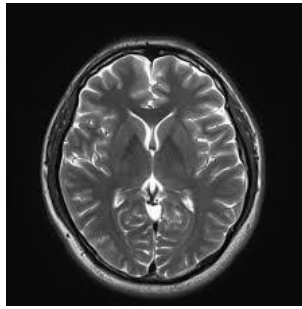


Figure 6.2 Image Before Augmentation

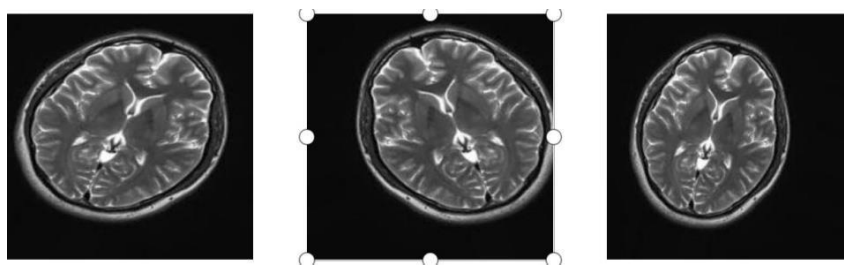


Figure 6.3 Image After Augmentation

After this the final implementation is done where the training process takes place and the results are obtained.

Initially the anaconda navigator is started which is used to perform all the required execution of the project. This can be seen in the following figure

```
Anaconda Prompt
(base) C:\Users\Vadav>cd C:\Users\Vadav\projects\trincancer
(base) C:\Users\Vadav\projects\trincancer>
```

Figure 6.4 Moving Inside the coding folder

Then the code is run using certain commands in the command prompt which can be seen in the following figure

```

C:\Users\VP\project\trancercr\Dataset_Augmentation>python augmentation_demo.py --image 1.jpg --output output.jpg
C:\Users\VP\project\trancercr\Dataset_Augmentation>python train.py --dataset dataset/images --model trancercr_model.model
[INFO] processed 500/1256
[INFO] processed 1000/1256
WARNING:tensorflow:From C:\Users\VP\Anaconda3\lib\site-packages\tensorflow\python\ops\resource_variable_ops.py:435: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
100% 5.0MB 5.0MB/s
File "C:\Users\VP\Anaconda3\lib\urllib\request.py", line 1317, in do_open
    encode_chunked(req, headers['Transfer-Encoding'])
File "C:\Users\VP\Anaconda3\lib\urllib\request.py", line 1264, in request
    self._send_request(method, url, body, headers, encode_chunked)
File "C:\Users\VP\Anaconda3\lib\urllib\request.py", line 1259, in _send_request
    self._send_output(body, encode_chunked=encode_chunked)
File "C:\Users\VP\Anaconda3\lib\urllib\request.py", line 1229, in _send_output
    self._send_output(message_body, encode_chunked=encode_chunked)
File "C:\Users\VP\Anaconda3\lib\urllib\request.py", line 1826, in _send_output
    self.send(msg)
File "C:\Users\VP\Anaconda3\lib\urllib\request.py", line 966, in send
    self.connect()
File "C:\Users\VP\Anaconda3\lib\urllib\request.py", line 1486, in connect
    super().connect()
File "C:\Users\VP\Anaconda3\lib\urllib\request.py", line 938, in connect
    (self.host,self.port), self.timeout, self.source_address)
File "C:\Users\VP\Anaconda3\lib\socket.py", line 787, in create_connection
    for res in getaddrinfo(host, port, 0, SOCK_STREAM):
File "C:\Users\VP\Anaconda3\lib\socket.py", line 748, in getaddrinfo
    for res in _socket.getaddrinfo(host, port, family, type, proto, flags):
socket.gaierror: [Errno 11001] getaddrinfo failed
During handling of the above exception, another exception occurred:

```

Figure 6.5 Executing In the Command Prompt

As the execution gets completed the accuracy is been plotted for the training obtained and it can be seen in the following figure which is around 100%.

```

134/12 [=====] - 1s 6ms/sample - loss: 2.0671 - acc: 0.8582
12/12 [=====] - 12s 997ms/step - loss: 1.7525 - acc: 0.9756 - val_loss: 1.9969 - val_acc: 0.8582
[INFO] evaluating after initialization...
      precision    recall  f1-score   support

 Pneumonia       0.76       1.00       0.86        56
   ignore       0.97       1.00       0.99        38
    normal       1.00       0.53       0.69        40

 accuracy                   0.86        134
 macro avg                   0.91        134
weighted avg                   0.89        134

[INFO] re-compiling model...
[INFO] fine-tuning model...
Epoch 1/20
11/12 [=====>...] - ETA: 1s - loss: 1.6986 - acc: 0.9970
Epoch 1/20
134/12 [=====]

```

Figure 6.6 Training before fine tuning

On fine tuning the algorithm the training accuracy is further improved which can be seen in the image below:

```
Epoch 20/20
11/12 [=====>...] - ETA: 0s - loss: 1.6992 - acc: 0.9911Epoch
ch 1/20
134/12 [=====]
=====
=====] - 1s 6ms/sample - loss: 1.6744 - acc: 1.0000
12/12 [=====] - 12s 1s/step - loss: 1.6976 - acc: 0.991
9 - val_loss: 1.6746 - val_acc: 1.0000
[INFO] evaluating after fine-tuning...
      precision    recall  f1-score   support

 Pneumonia         1.00         1.00         1.00         56
   ignore         1.00         1.00         1.00         38
    normal         1.00         1.00         1.00         40

 accuracy
macro avg         1.00         1.00         1.00         134
weighted avg         1.00         1.00         1.00         134

[INFO] serializing model...
```

Figure 6.7 Training after fine tuning

The below image shows the training loss graph:

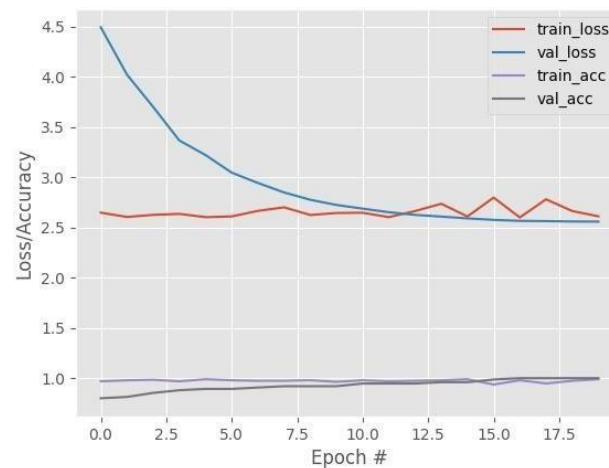


Figure 6.8 Training loss graph

A web application is developed using a javascript framework reactJS for interactively uploading brain scan images and viewing the results. The below image shows the login page of the web application:

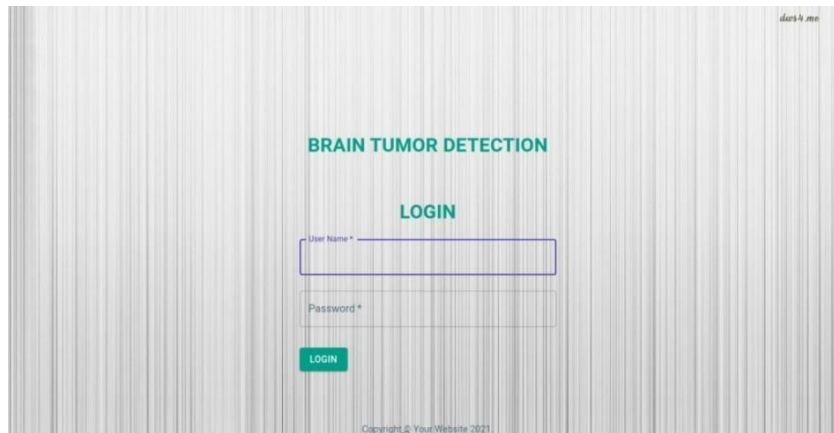


Figure 6.9 Brain tumour login page

On successful login the home page is displayed to the user. The below image shows the user uploading the brain scan image for effective brain tumour diagnosis:

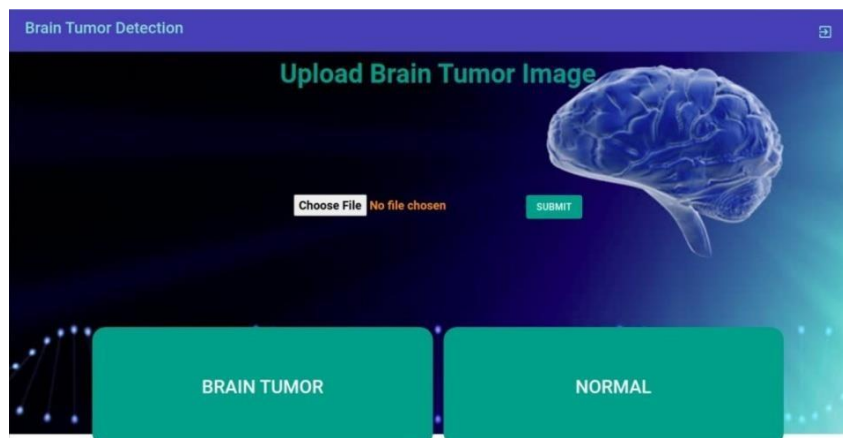


Figure 6.10 Home page

The user can upload brain scan images to check for brain tumour. The below image shows the user uploading brain scan image for effective diagnosis:

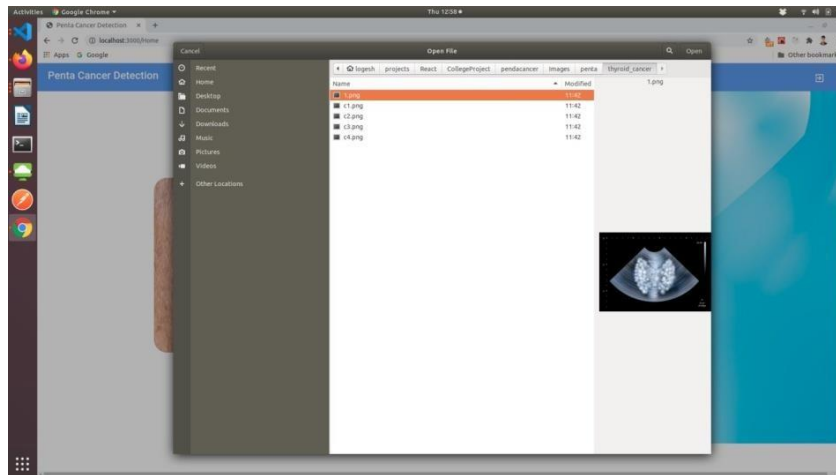


Figure 6.11 Brain tumour x-ray upload

The uploaded image is then sent to the database and the response is given back to the web page which are the results.

The below image shows that the uploaded image is of a normal brain scan.

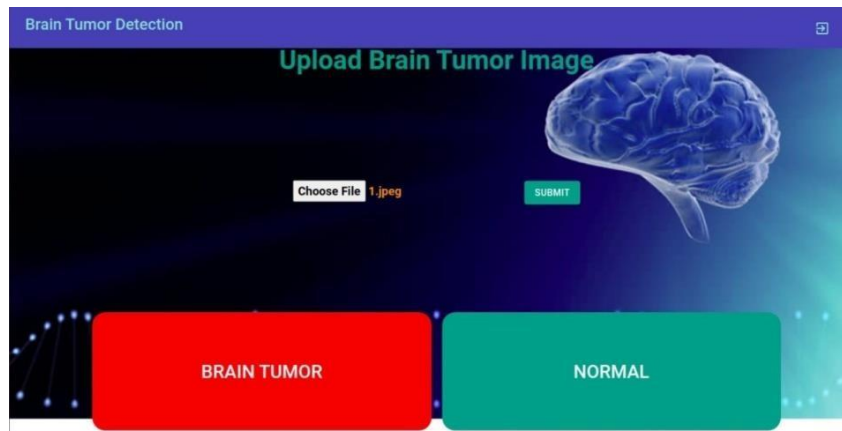


Figure 6.12 Normal brain scan image

If the uploaded image is that of a brain tumour is mentioned along with it. The below image shows that the uploaded image is that of a thyroid cancer.

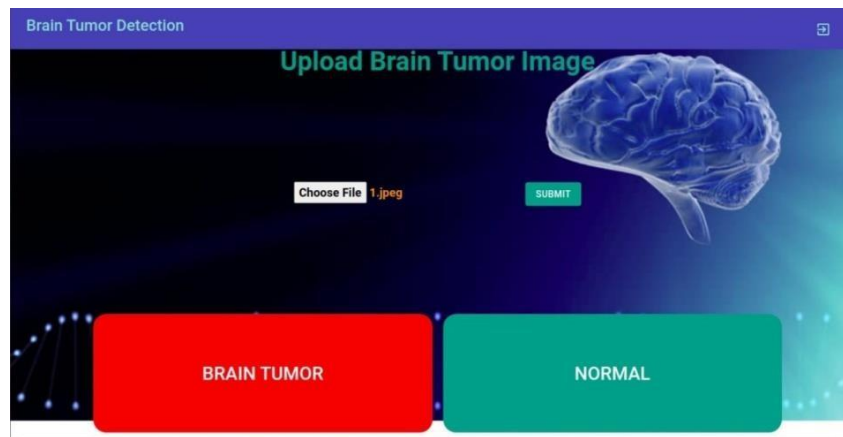


Figure 6.13 Brain tumour image

Thus, from the above results and discussion, it is clear that we have efficiently made a project for detecting the presence of brain tumour. Thus, we have successfully implemented the scope of the project

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

The project has been successfully implemented to detect the presence of brain tumour and provide prior measures to avoid the disease. This also help in providing efficient treatment in a most cheap way and eventually reduce the time required for finding the brain tumour. In the current state it is done manually which consumes more time and also involves human error rate. So, by this project reduces the time required for manual classification and eliminates the human error rate by this project.

7.2 FUTURE WORK

In the coming future, we review the application of the brain tumour determination technology in the healthcare field and it can promote for detecting various types of cancer with more accuracy. In medical field they have more chance to develop or convert this project in many ways. Thus, this project has an efficient scope in coming future where manual predicting can be converted to computerized production in a cheap way.

APPENDIX

```
# USAGE
# python3 food.py --dataset dataset/images --model cavity.model

# import the necessary packages
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from pyimagesearch.preprocessing import ImageToArrayPreprocessor
from pyimagesearch.preprocessing import AspectAwarePreprocessor
from pyimagesearch.datasets import SimpleDatasetLoader
from pyimagesearch.nn.conv import FCHadNet
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from imutils import paths
import numpy as np
import argparse
import os

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required=True,
                help="path to input dataset")
ap.add_argument("-m", "--model", required=True,
                help="path to output model")
args = vars(ap.parse_args())

# construct the image generator for data augmentation
aug = ImageDataGenerator(rotation_range=30, width_shift_range=0.1,
                          height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,
                          horizontal_flip=True, fill_mode="nearest")

# grab the list of images that we'll be describing, then extract the
# class label names from the image paths
print("[INFO] loading images...")
imagePaths = list(paths.list_images(args["dataset"]))
classNames = [pt.split(os.path.sep)[-2] for pt in imagePaths]
classNames = [str(x) for x in np.unique(classNames)]

# initialize the image preprocessors
```

```

aap = AspectAwarePreprocessor(224, 224) iap =
ImageToArrayPreprocessor()

# load the dataset from disk then scale the raw pixel intensities to#
the range [0, 1]
sdl = SimpleDatasetLoader(preprocessors=[aap, iap])(data, labels) =
sdl.load(imagePaths, verbose=500) data = data.astype("float") / 255.0

# partition the data into training and testing splits using 75% of# the
data for training and the remaining 25% for testing (trainX, testX,
trainY, testY) = train_test_split(data, labels,
test_size=0.25, random_state=42)

# convert the labels from integers to vectors trainY =
LabelBinarizer().fit_transform(trainY) testY =
LabelBinarizer().fit_transform(testY)

# load the VGG16 network, ensuring the head FC layer sets are left#
off
baseModel = VGG16(weights="imagenet", include_top=False,
input_tensor=Input(shape=(224, 224, 3)))

# initialize the new head of the network, a set of FC layers# followed
by a softmax classifier
headModel = FCHeadNet.build(baseModel, len(classNames), 256)

# place the head FC model on top of the base model -- this will#
become the actual model we will train
model = Model(inputs=baseModel.input, outputs=headModel)

# loop over all layers in the base model and freeze them so they# will
*not* be updated during the training process
for layer in baseModel.layers: layer.trainable = False

# compile our model (this needs to be done after our setting our#
layers to being non-trainable
print("[INFO] compiling model...") opt = Adam(lr=0.001)
model.compile(loss="categorical_crossentropy", optimizer=opt,
metrics=["accuracy"])

# train the head of the network for a few epochs (all other# layers are
frozen) -- this will allow the new FC layers to # start to become
initialized with actual "learned" values # versus pure random

```

```

print("[INFO] training head...") model.fit_generator(aug.flow(trainX,
trainY, batch_size=32),
validation_data=(testX, testY), epochs=15,
steps_per_epoch=len(trainX) // 32, verbose=1)

# evaluate the network after initialization print("[INFO] evaluating after
initialization...") predictions = model.predict(testX, batch_size=32)
print(classification_report(testY.argmax(axis=1),
predictions.argmax(axis=1), target_names=classNames))

# now that the head FC layers have been trained/initialized, lets #
unfreeze the final set of CONV layers and make them trainable for
layer in baseModel.layers[15:]:
    layer.trainable = True

# for the changes to the model to take affect we need to recompile#
the model, this time using SGD with a *very* small learning rate
print("[INFO] re-compiling model...")
opt = SGD(lr=0.0001)
model.compile(loss="categorical_crossentropy", optimizer=opt,
metrics=["accuracy"])

# train the model again, this time fine-tuning *both* the final set# of
CONV layers along with our set of FC layers
print("[INFO] fine-tuning model...")
model.fit_generator(aug.flow(trainX, trainY, batch_size=32),
validation_data=(testX, testY), epochs=10,
steps_per_epoch=len(trainX) // 32, verbose=1)

# evaluate the network on the fine-tuned model print("[INFO]
evaluating after fine-tuning...") predictions = model.predict(testX,
batch_size=32) print(classification_report(testY.argmax(axis=1),
predictions.argmax(axis=1), target_names=classNames))

# save the model to disk print("[INFO] serializing model...")
model.save(args["model"])

# USAGE
# python3 recognize.py --image 1.jpg --conf config/config.json

# import the necessary packages from pyimagesearch.utils import
Conf

```

```

from pyimagesearch.notifications import TwilioNotifier
from imutils.video import VideoStream
from imutils import paths
from threading import Thread
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from datetime import datetime
from datetime import date
# import RPi.GPIO as GPIO
import numpy as np
import argparse
import imutils
import pickle
import time
import cv2
import os

```

```
font = cv2.FONT_HERSHEY_SIMPLEX
```

```

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
                help="path to the input image")
ap.add_argument("-c", "--conf", required=True,
                help="path to the input configuration file")

```

```
args = vars(ap.parse_args())
```

```

# load the configuration file and initialize the Twilio notifier
conf = Conf(args["conf"])
tn = TwilioNotifier(conf)

```

```

# grab the paths to gesture icon images and then initialize the icons
# dictionary where the key is the gesture name (derived from the
# image# path) and the key is the actual icon image
print("[INFO] loading icons...")
imagePaths = paths.list_images(conf["assets_path"])
icons = {}

```

```

# loop over the image paths for imagePath in imagePaths:
# extract the gesture name (label) the icon represents from the #
# filename, load the icon, and then update the icons dictionary
label = imagePath.split(os.path.sep)[-1].split(".")[0]
icon = cv2.imread(imagePath)
icons[label] = icon

```

```

# grab the top-left and bottom-right (x, y)-coordinates for the #
# gesture capture area
TOP_LEFT = tuple(conf["top_left"])
BOT_RIGHT = tuple(conf["bot_right"])

```

```

# load the trained gesture recognizer model and the label binarizer
print("[INFO] loading model...")
model = load_model(str(conf["model_path"]))
# lb = pickle.loads(open(str(conf["lb_path"]), "rb").read())

```

```

# start the video stream thread
print("[INFO] starting video stream thread...")#vs =
VideoStream(src=0).start()
#vs = VideoStream(usePiCamera=True).start() #time.sleep(2.0)

#frame = load_img(args["image"]) #cv2
# loop over frames from the video streamwhile True:
# grab the frame from the threaded video file stream and grab the#
current timestamp
imagePaths = list(paths.list_images("dataset/images")) classNames =
[pt.split(os.path.sep)[-2] for pt in imagePaths] classNames = [str(x) for
x in np.unique(classNames)]
frame = cv2.imread(args["image"]) timestamp = datetime.now()

# resize the frame and then flip it horizontally frame =
imutils.resize(frame, width=400) frame = cv2.flip(frame, 1)

# clone the original frame and then draw the gesture capture area
#clone = frame.copy()
#cv2.rectangle(clone, TOP_LEFT, BOT_RIGHT, (0, 0, 255), 2)

#roi = frame[TOP_LEFT[1]:BOT_RIGHT[1],
TOP_LEFT[0]:BOT_RIGHT[0]]
# roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
# roi = cv2.threshold(roi, 75, 255, cv2.THRESH_BINARY)[1] #visROI =
roi.copy()

# now that we have the hand region we need to resize it to be
# the same dimensions as what our model was trained on, scale# it
to the range [0, 1], and prepare it for classification
roi = cv2.resize(frame, (224, 224))roi = roi.astype("float") / 255.0
roi = img_to_array(roi)
roi = np.expand_dims(roi, axis=0)

# classify the input imageproba = model.predict(roi)[0]
label = classNames[proba.argmax()]

# if label == "images" or "plates":

# print("No Food") #elif label == "chapathi" :

print(label)

```

```

if label == "idly":
    cv2.putText(frame,'Each Idly gives 33 calories',(0,20), font, 0.5,
    (0,0,0), 1,
    cv2.LINE_AA)
elif label == "dosa":
    cv2.putText(frame,'Each dosa gives 133 calories',(0,20), font, 0.5,
    (0,0,0), 1,
    cv2.LINE_AA)
elif label == "rice":
    cv2.putText(frame,'Each bowl of rice gives 136 calories',(0,20), font,
    0.5, (0,0,0),
    1, cv2.LINE_AA)
elif label == "chapathi":
    cv2.putText(frame,'Each chapati gives 104 calories',(0,20), font,
    0.5, (0,0,0), 1,cv2.LINE_AA)

```

```

# check to see if the label from our model matches the label

```

```

cv2.imshow("ROI", frame)
# cv2.imshow("Security Feed", clone)# cv2.imshow("Passcode",
canvas) key = cv2.waitKey(1) & 0xFF

```

```

# if the `q` key was pressed, break from the loop
if key == ord("q"):reset_lights() break

```

```

# do a bit of cleanup cv2.destroyAllWindows() vs.stop()
1

```

REFERENCES

- [1] Amir Mirbeik-Sabzevari, Student Member, IEEE, Negar Tavassolian, Senior Member, IEEE “Ultra-Wideband, Stable Normal and Cancer Skin Tissue Phantoms for Millimeter-Wave Skin Cancer Imaging”, IEEE Transactions on Biomedical Engineering.
- [2] Annette McWilliams, Parmida Beigi, Akhila Srinidhi, Stephen Lam, and Calum E. MacAulay “Sex and Smoking Status Effects on the Early Detection of Early Lung Cancer in High-Risk Smokers using an Electronic Nose” IEEE Transactions on Biomedical Engineering.
- [3] Arnaud A. A. Setio, Francesco Ciompi, Geert Litjens, Paul Gerke, Colin Jacobs, Sarah J. van Riel, Mathilde Marie Winkler Wille, Matiullah Naqibullah, Clara I. S´anchez, Bram van Ginneken “Pulmonary nodule detection in CT images: false positive reduction using multi-view convolutional networks” IEEE Transactions on Medical Imaging.
- [4] Hua Zhong and Mingzhou Song “A fast exact functional test for directional association and cancer biology applications” IEEE/ACM Transactions on Computational Biology and Bioinformatics.
- [5] Jose M. Anton-Rodriguez, Peter Julyan, Ibrahim Djoukhadar, David Russell, D. Gareth Evans, Alan Jackson, and Julian C. Matthews “Comparison of a Standard Resolution PET-CT Scanner With an HRRT Brain Scanner for Imaging Small Tumors Within the Head” IEEE transactions on radiation and plasma medical sciences, vol. 3, no. 4, july 2019.
- [6] Jun Xu, Lei Xiang, Qingshan Liu, Hannah Gilmore, Jianzhong Wu, Jinghai Tang, and Anant Madabhushi “Stacked Sparse Autoencoder (SSAE) for Nuclei Detection on Breast Cancer Histopathology Images” IEEE Transactions on Medical Imaging.
- [7] Jue Jiang, Yu-chi Hu, Chia-Ju Liu, Darragh Halpenny, Matthew D. Hellmann, Joseph O. Deasy, Gig Mageras and Harini Veeraraghavan^{1*} “Multiple Resolution Residually Connected Feature Streams For Automatic Lung Tumor Segmentation From CT Images” IEEE Transactions on Medical Imaging.
- [8] Koyel Mandal, Rosy Sarmah, and Dhruba Kumar Bhattacharyya “Biomarker Identification for Cancer Disease Using Biclustering Approach: An Empirical Study” IEEE/ACM Transactions on Computational Biology and Bioinformatics.

- [9] Sepideh Rahmatinia, and Babak Fahimi University of Texas at Dallas, Department of Electrical and Computer Engineering, Richardson, TX, USA “Magneto-Thermal Modeling of Biological Tissues: A Step towards Breast Cancer Detection” IEEE Transactions on Magnetics.
- [10] Shekoofeh Azizi, Sharareh Bayat, Pingkun Yan, Amir Tahmasebi, Jin Tae Kwak, Sheng Xu, Baris Turkbey, Peter Choyke, Peter Pinto, Bradford Wood, Parvin Mousavi, Purang Abolmaesumi “Deep Recurrent Neural Networks for Prostate Cancer Detection: Analysis of Temporal Enhanced Ultrasound” IEEE Transactions on Medical Imaging.
- [11] Sara Reis, Patrycja Gazinska, John H. Hipwell, Thomy Mertzaniidou, Kalnisha Naidoo, Norman Williams, Sarah Pinder, David J. Hawkes “Automated Classification of Breast Cancer Stroma Maturity from Histological Images” IEEE Transactions on Biomedical Engineering.
- [12] Yanbo Wang, Weikang Qian, Bo Yuan, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, 200240, P.R.China 2 UM- SJTU Joint Institute, Shanghai Jiao Tong University, Shanghai, 200240, P.R.China “A graphical model of smoking-induced global instability in lung cancer” IEEE/ACM Transactions on Computational Biology and Bioinformatics.
- [13] Yi Wang, Na Wang, Min Xu, Junxiong Yu, Chenchen Qin, Xiao Luo, Xin Yang, Tianfu Wang, Anhua Li, and Dong Ni “Deeply-Supervised Networks with Threshold Loss for Cancer Detection in Automated Breast Ultrasound IEEE Transactions on Medical Imaging.
- [14] Ye Tian, Sean S. Wang, Zhen Zhang, Olga C. Rodriguez, Emanuel Petricoin III, le-Ming Shih, Daniel Chan, Maria Avantaggiati, Guoqiang Yu, Shaozhen Ye, Robert Clarke, Chao Wang, Bai Zhang, Yue Wang, and Chris Albanese “Integration of Network Biology and Imaging to Study Cancer Phenotypes and Responses” IEEE/ACM transactions on computational biology and bioinformatics, vol. 11, no. 6, november/december 2014.
- [15] H. Yan, M. Romero-López, H.B. Frieboes, C.C.W. Hughes, J.S. Lowengrub “Multiscale Modeling of Glioblastoma Suggests that the Partial Disruption of Vessel/Cancer Stem Cell Crosstalk Can Promote Tumor Regression without Increasing Invasiveness” IEEE Transactions on Biomedical Engineering.