

6COSC023W – FINAL PROJECT REPORT

BRAIN TUMOUR DETECTION

STUDENT: KARAN KUMAR (W1815590)

SUPERVISOR: VASSILIS KONTOGIANNIS

**THIS REPORT IS SUBMITTED IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS
FOR THE**

**BENG (HONS) SOFTWARE ENGINEERING DEGREE
AT THE UNIVERSITY OF WESTMINSTER.**

**SCHOOL OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF WESTMINSTER**

20/03/2024

Document Scope

The purpose of this document is to describe and reflect on the processes that took place in developing the Final Project. Discuss any ethical issues associated with your project and describe the methodology that was adopted to develop its design, implementation and testing.

All chapter word counts in this document are approximate and are not intended to be prescriptive.

Declaration

This report has been prepared based on my own work. Where other published and unpublished source materials have been used, these have been acknowledged in references.

Word Count: 10,302

Student Name: KARAN KUMAR

Date of Submission: 01/05/2024

Abstract

With brain tumours responsible for an astounding 76% of cancer-related deaths worldwide, they represent a serious threat to public health. (1) Despite improvements in medical imaging, most tumours are discovered using outdated methods when they are already advanced, drastically reducing the number of available treatments and affecting patient outcomes. The absence of effective early detection techniques contributes to this serious problem, leading to delayed diagnoses and not enough treatments. In response to these obstacles, the goal of this project is to create a comprehensive system for detecting brain tumours by utilising cutting-edge algorithms, especially Support Vector Machines (SVM) and Convolutional Neural Networks (CNN).

The main goal is to enhance the diagnosis of brain tumours, which will result in more precise and effective medical assessments that will meet the urgent need for early tumour detection in relation to prediction and patient outcomes. A multi-pronged approach was used, which included gathering data from MRI image datasets containing cases of tumours and non-tumours, pre-processing methods, training models with CNN and SVM algorithms, and a thorough assessment using metrics like accuracy, F1 score, precision, and sensitivity. A thorough comparative study evaluated the strengths and weaknesses of each model's performance in identifying brain tumours.

With an astounding 96.33% accuracy and 97.28% F1 score, the SVM algorithm outperformed the CNN model, which had 92.24% accuracy and 94.46% F1 score. The results showed how much better the SVM algorithm performed than CNN. The power of SVM resides in its capacity to locate the best hyperplanes that maximise the margin between classes, even in the face of MRI images' irregular features. On the other hand, although CNNs are frequently used for image classification, they might find it difficult to pick up on small details that are essential for differentiating between tumorous and non-tumorous areas in brain scans.

To sum up, this project effectively created a strong brain tumour detection system using machine learning algorithms to improve medical evaluations and support

international efforts for cancer treatment and prevention. The SVM algorithm turned out to be the superior performer, but the CNN model's encouraging outcomes suggest that more optimisation may be possible. By means of ongoing improvement, integration of input, and investigation of new methods, this system displays potential as an important instrument in the battle against brain tumours, ultimately enhancing patient results and standard of living. The real value is in how it can help with early detection, accurate diagnosis, and the continuous innovation in healthcare that comes from advances in medical imaging and computational techniques.

Acknowledgements

In this moment, I am overwhelmed with gratitude for the deep impact of those who have shaped my journey.

To my late Bua (Aunty), who sadly lost her battle with cancer. Though she is no longer with us, her unwavering strength and sacrifice continues to resonate in my heart, providing the courage and determination to pursue my dreams.

To my parents, whose support and encouragement have been the guiding light through every challenge and triumph. Your belief in me, even in moments of doubt, has been the fuel driving me forward on this journey to make you proud.

Finally, to Vassilis Kontogiannis, my supervisor, I would like to express my appreciation for your mentorship, continuous support, and encouragement. Your guidance has been instrumental in shaping my Final Year Project journey.

Table of contents

Document Scope.....	2
Declaration	3
Abstract	4
Acknowledgements	6
Table of contents	7
List of figures	9
List of tables.....	10
1. Introduction	11
1.1 Problem statement	11
1.2 Aims and Objectives.....	13
2. Background	14
2.1 Literature survey.....	14
<i>Paper 1: A survey on Brain Tumour Detection using Image Processing Techniques. (10)</i>	14
<i>Paper 2: Brain Tumour Detection using CNN. (2)</i>	14
<i>Paper 3: Brain Tumour Detection and multi - classification using advanced deep learning techniques. (11)</i>	15
<i>Paper 4: Review of Brain Tumour Detection from MRI Images. (12)</i>	15
<i>Paper 5: Brain Tumour Detection based on Deep Learning approaches and Magnetic Resonance Imaging. (13)</i>	16
<i>Paper 6: Automated categorization of Brain Tumour from MRI using CNN features and SVM. (14)</i>	16
2.2 Review of projects / applications	18
<i>Method developed by Damodharan & Raghavan:</i>	18
<i>Method developed by Alfonse & Salem:</i>	19
<i>Method developed by Zanaty:</i>	20
2.3 Review of tools, frameworks and techniques.....	23
3. Legal, social and ethical issues.....	26
<i>Legal Considerations:</i>	26
<i>Ethical Considerations:</i>	26
<i>Social Considerations:</i>	26
<i>Professional Considerations:</i>	27
<i>Security Considerations:</i>	27
<i>Project Life Cycle Stages:</i>	28
<i>Methodology and Development Techniques:</i>	29
<i>Implementation:</i>	30
<i>Testing:</i>	30

5. Design.....	31
<i>User Interface:</i>	31
<i>Infrastructure:</i>	31
<i>Functionality:</i>	32
<i>Algorithm Development:</i>	32
<i>Other Considerations:</i>	32
6. Tools and implementation	34
6.1 Tools.....	34
<i>Python:</i>	34
<i>Jupyter Notebook:</i>	35
6.2 Implementation.....	36
<i>Support Vector Machine (SVM) Algorithm Implementation:</i>	36
<i>Convolutional Neural Network (CNN) Algorithm Implementation:</i>	38
7. Testing	43
<i>Support Vector Machine (SVM) Algorithm Testing:</i>	43
<i>Convolutional Neural Network (CNN) Algorithm Testing:</i>	45
<i>Conclusion:</i>	47
7.1 Test coverage.....	48
<i>Generalisation Test (Performance Requirement):</i>	48
<i>Misclassified Analysis Test (Evaluation of Model Requirement):</i>	49
<i>Visualisation Test (User Interface requirement):</i>	49
<i>Pre-processing Test (Data Pre-processing Requirement):</i>	50
7.2 Test methodology	51
<i>Generalisation Test:</i>	51
<i>Misclassified Analysis Test:</i>	51
<i>Data Pre-processing Test:</i>	52
<i>Visualisation Test:</i>	52
<i>Feedback Collection:</i>	52
8. Conclusions and reflections.....	54
9. References	57
10. Bibliography	59
Appendix I.....	61

List of figures

[Figure 1](#) – Latest statistics of Brain Tumour (Cancer Research UK).

[Figure 2](#) – Proposed Brain Tumour Detection method by Damodharan and Raghavan.

[Figure 3](#) – Proposed Brain Tumour Detection method by Alfonse & Salem.

[Figure 4](#) – Proposed Brain Tumour Detection method by Zanaty.

List of tables

[Table 1](#) – Comparison of different languages.

[Table 2](#) – Support Vector Machine (SVM) Algorithm Testing.

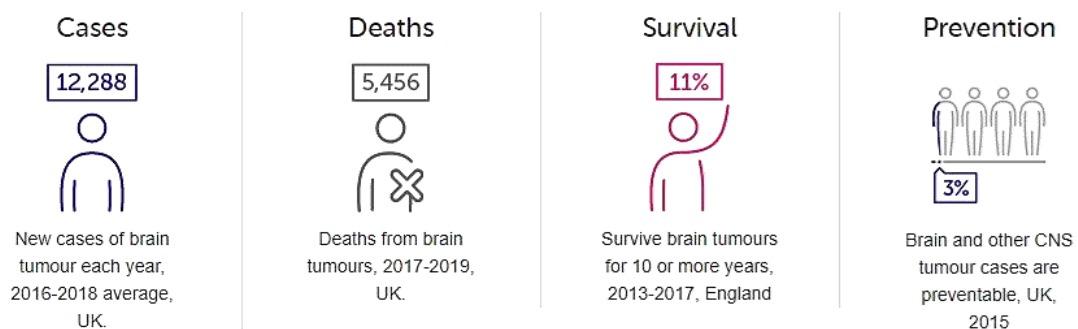
[Table 3](#) – Convolutional Neural Network (CNN) Algorithm Testing.

1. Introduction

1.1 Problem statement

Since brain tumours account for a sizable percentage of cancer-related fatalities globally, they pose a serious threat to public health. They are responsible for an astounding 76% of all cancer-related fatalities, according to the International Agency for Research on Cancer (IARC). (1) Even with improvements in medical imaging technology, most tumours are found at advanced stages using the detection techniques currently in use, which severely restricts available treatment options and degrades patient outcomes. (4) Moreover, the problem is made worse by the lack of efficient early detection methods, which exposes patients to late-stage diagnoses and less effective treatments. The limitations of current methodologies highlight the urgent need for enhanced systems for the detection of brain tumours. Although systems developed by the World Health Organisation provide insightful information about identifying anomalies in MRI scans, they are not precise enough to accurately forecast the behaviour of tumours. (2) As a result, patients and healthcare providers frequently lack individualised guidance, which can result in less than ideal treatment choices and results. Furthermore, the existing approaches mostly focuses on finding tumours after symptoms appear, missing chances for early intervention—even in people who have a genetic link to specific kinds of brain tumours. (5)

Figure 1 below showing the latest statistics of Brain Tumour.



Source: Cancer Research UK

This project attempts to create a comprehensive brain tumour detection system using advanced algorithms, particularly Convolutional Neural Network (CNN) and Support Vector Machine (SVM) techniques, in response to these urgent challenges. The significant influence that early detection has on patient outcomes emphasises the

urgency of this commitment. Studies show that early diagnosis increases treatment effectiveness and improves prediction, potentially saving lives and minimising the overall burden of the illness and mortality rates. (3) The project's anticipated software/application will enable more accurate and efficient medical assessments, revolutionising the diagnostic process. The system will enable early intervention and decrease the negative consequences of late-stage diagnoses by utilising the power of CNN and SVM algorithms. There is great potential for both strengthening patient outcomes and lowering healthcare costs related to longer hospital stays and treatments with this new approach towards early detection. This project is important in ways that go beyond providing healthcare and medical research. It supports larger societal objectives, such as lowering cancer-related illnesses and fatalities and improving the quality of life for afflicted people and their families, by addressing the urgent need for early detection tools. Moreover, the advancement of sophisticated detection systems improves the basis for upcoming cancer treatment and medical imaging technology research and innovation.

In summary, the urgent need for better brain tumour detection systems demands creative solutions that can get past current constraints. The goal of this project is to transform the diagnostic process by creating a software/application that utilises CNN and SVM algorithms. This will allow for early intervention and ultimately result in lifesaving. We can change the way brain tumours are found and treated, bringing in a new era of better patient outcomes through collaborative research and technological innovation.

1.2 Aims and Objectives

The aim and objective of this project is to use Convolutional Neural Network (CNN) and Support Vector Machine (SVM) algorithms to develop reliable and accurate brain tumour detection systems. The main objective is to improve brain tumour diagnosis, which will ultimately lead to more accurate and efficient medical evaluations.

Enhancing cancer prevention and treatment is in line with the critical need for early tumour identification, which is essential for prediction and, in the end, lowers death rates by detecting cancer before it reaches advanced stages.

Several goals have been outlined in order to accomplish this goal. First, a variety of MRI image datasets, including cases with and without tumours, will be collected. To guarantee there are enough examples for neural network training, the dataset will be pre-processed and data-augmented before being implemented with CNN. After that, the CNN architecture will be built, and the model will be trained and assessed using metrics like Accuracy, F1 score, Precision & Sensitivity.

Principal Component Analysis (PCA) and feature scaling are two pre-processing techniques that will be used to reduce dimensionality in the collected data before the SVM implementation. The SVM model will be built, trained, and assessed using the exact same metrics as the CNN implementation after the dataset has been divided into training and testing sets. After both models are independently developed, an in-depth comparison will be carried out to evaluate how well they detect brain tumours. The benefits and drawbacks of each model will be assessed using crucial metrics like Accuracy, F1 score, Precision & Sensitivity. Future diagnostic techniques will benefit from this comparative analysis's insights into the most effective method for finding brain tumours.

In conclusion, the goal of this project is to improve cancer diagnosis by creating and testing cutting edge systems for detecting brain tumours using CNN and SVM algorithms. The ultimate aim is to improve patient outcomes and contribute to the global effort in cancer prevention and treatment by improving early detection capabilities.

2. Background

2.1 Literature survey

Paper 1: A survey on Brain Tumour Detection using Image Processing Techniques. (10)

- **Publication Year:** 2017
- **Author:** Luxit Kapoor, Sanjeev Thakur
- **Journal Name:** IEEE 7th International Conference on Cloud Computing, Data Science & Engineering
- **Summary:** In order to identify brain tumours from MRI images, a number of methods that are commonly used in medical image processing are surveyed in this paper. This paper, which lists the various techniques in use, was written based on that research. There's also an overview of every technique available, Segmentation is also the most important of all the steps in the process of detecting tumours.

Paper 2: Brain Tumour Detection using CNN. (2)

- **Publication Year:** 2021
- **Author:** Pandey Saumya, Goyal Arya, Vansuha D
- **Journal Name:** Turkish journal of computer and mathematics education, 2021, Vol.12 (11), p.4597-4603
- **Summary:** The medical industry has benefited immensely from artificial intelligence. One of its many varied uses is in diagnosis, where AI is used to reduce human error in judgement. While many imaging methods, such as magnetic resonance imaging (MRI) and computed tomography scans (CT) have been used recently, MRI is the most dependable method due to its safety and dependability. This paper aims to present a similar technique wherein we propose to use a convolutional neural network to detect the presence of the tumour.

Paper 3: Brain Tumour Detection and multi - classification using advanced deep learning techniques. (11)

- **Publication Year:** 2021
- **Author:** Sadad Tariq, Rehman Amjad, Munir Asim, Saba Tanzila, Tariq Usman, Ayesha Noor, Abbasi Rashid
- **Journal Name:** Microscopy research and technique, 2021, Vol.84 (6), p.1296-1308
- **Summary:** If brain cancer is not discovered in its early stages, an unchecked growth of brain cells results in a brain tumour. The survival rate of patients with brain tumours and the planning of their treatment depend heavily on early diagnosis. Brain tumours come in different shapes, sizes, and can require different treatments. As a result, brain tumour detection by hand is difficult, time-consuming, and prone to mistakes. Therefore, there is currently a need for highly precise automated computer-assisted diagnosis. This paper reports segmentation using Unet architecture on the Fig share data set, with a level of 0.9504 of the intersection over union (IoU) attained using ResNet50 as the backbone. To improve the classification rate, the concepts of pre-processing and data augmentation were presented in the article.

Paper 4: Review of Brain Tumour Detection from MRI Images. (12)

- **Publication Year:** 2016
- **Author:** Deepa, Akansha Singh
- **Journal Name:** IEEE International Conference on Computing for Sustainable Global Development
- **Summary:** This paper reviews some of the recent research on the segmentation and detection of brain tumours. Various methods employed by different researchers to identify the brain tumour from the MRI pictures are explained. Based on this review, we discovered that one of the most active research areas is the automation of brain tumour detection and segmentation from MRI images.

Paper 5: Brain Tumour Detection based on Deep Learning approaches and Magnetic Resonance Imaging. (13)

- **Publication Year:** 2023
- **Author:** Abdusalomov Akmalbek Bobomirzaevich, Mukhiddinov Mukhridin, Whangbo Taeg Keun
- **Journal Name:** Cancers, 2023, Vol.15 (16), p.4172
- **Summary:** A brain tumour's rapid growth of unusual brain cells poses a serious risk to an adult's health since it can seriously impair organ function or even result in death. There is great variation in the sizes, textures, and locations of these tumours. Magnetic resonance imaging, or MRI, is a vital tool for locating cancerous tumours. However, manually identifying brain tumours is a challenging and time-consuming task that may result in errors. To address this, we present in this article an enhanced brain tumour detection system that includes an improved You Only Look Once version 7 (YOLOv7) model for the precise diagnosis of meningioma, glioma, and pituitary gland tumours. Image enhancement techniques, which apply various filters to the original images, improve the visual representation of the MRI scans. Therefore, this framework holds great promise as a useful tool for experts in the field of brain tumour diagnosis to make decisions.

Paper 6: Automated categorization of Brain Tumour from MRI using CNN features and SVM. (14)

- **Publication Year:** 2020
- **Author:** Deepak S, Ameer P.M
- **Journal Name:** Journal of ambient intelligence and humanized computing, 2021-08, Vol.12 (8), p.8357-8369
- **Summary:** An important part of the computer-aided diagnosis (CAD) system for the human brain is automated tumour classification. Although CAD of brain tumours is a well-researched topic, there are significant obstacles in certain particular areas. The category-based classification of brain tumours using magnetic resonance imaging (MRI) images among glioma, meningioma,

and pituitary tumours is one such difficult issue. With encouraging results, image classification tasks have been tackled by deep learning and machine learning algorithms. However, the small size of medical image databases is a related limitation to the classification of medical images. In order to address this difficulty, in this paper we combine support vector machines (SVM) and convolutional neural network (CNN) features for the purpose of classifying medical images.

2.2 Review of projects / applications

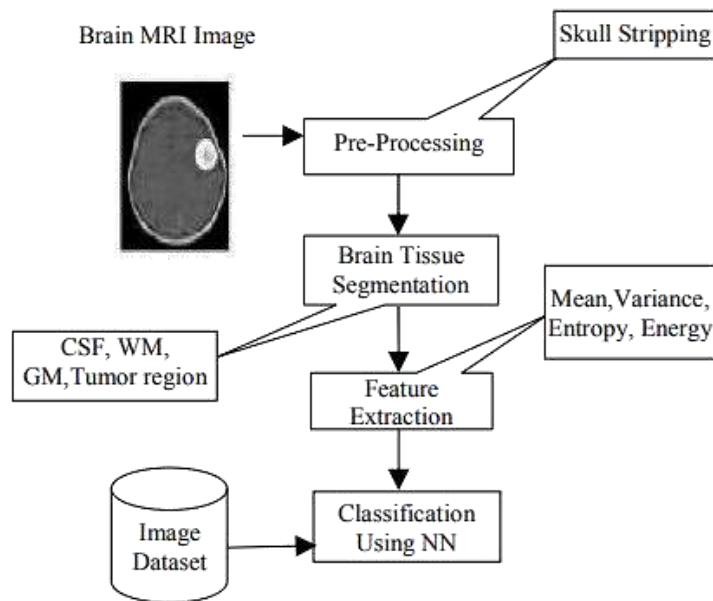
New techniques have been used to improve Brain Tumour Detection models and their efficiency as science and technology have advanced. To accurately predict the presence of a brain tumour, a number of methods, projects and applications have been introduced, most based on MRI scans. Most notably, fuzzy clustering means (FCM), support vector machine (SVM), artificial neural network (ANN), knowledge-based techniques, and expectation-maximization (EM) algorithm technique are some of the popular techniques and algorithms used. Below are multiple methods developed by researchers that have been evaluated in studies for Brain Tumour Detection:

Method developed by Damodharan & Raghavan:

A neural network-based method for the identification and categorization of brain tumours has been presented by Damodharan and Raghavan. This method uses a neural network-based classifier to produce a quality rate for each segmentation: WM, GM, CSF, and tumour region.

In order to distinguish between white matter (WM), grey matter (GM), cerebrospinal fluid (CSF), and tumour regions, Damodharan & Raghavan's method uses a neural network-based classifier to identify and categorise brain tumours. The method appears promising for analysing brain images, with a reported accuracy of 83%. (6) However, the lack of specific information about the dataset used for training and validation, along with its relatively low accuracy, may limit its applicability. Furthermore, the effectiveness of the method for various kinds of brain tumours is still unknown, indicating the need for additional testing and improvement.

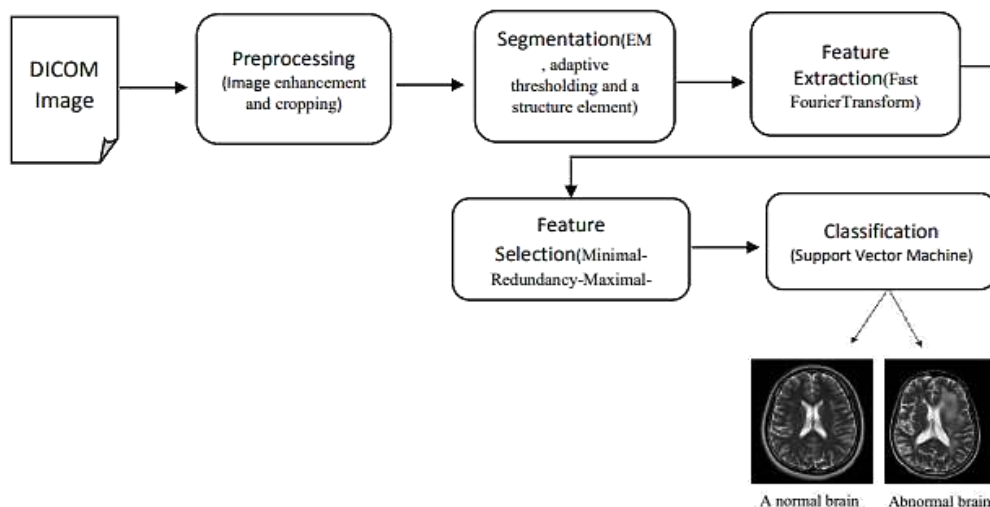
Figure 2 below showing the proposed method by Damodharan and Raghavan.



Method developed by Alfonse & Salem:

Using an SVM-based classifier, Alfonse and Salem have presented a method for automatically classifying brain tumours from MRI images. Minimal-Redundancy-Maximal-Relevance (MRMR) technique is used to reduce features and fast Fourier transform (FFT) is used to extract features in order to increase the classifier's accuracy. The accuracy achieved by this method is 98.9%. (7)

Figure 3 below showing the method developed by Alfonse & Salem.

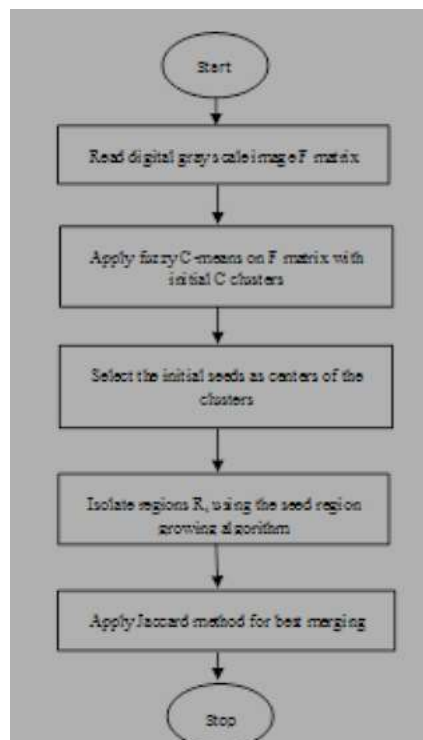


The process involves several stages: acquiring the dataset, pre-processing it, segmenting it using the Expectation Maximisation (EM) algorithm and adaptive thresholding, extracting features from the MRI data set using the Fast Fourier Transform (FFT), selecting the most valuable features using the Minimal-Redundancy-Maximal-Relevance (MRMR) requirement, and finally, the classification stage, where SVM is used to classify brain images as normal or abnormal.

Alfonse and Salem provide a reliable technique for automatically classifying brain tumours from MRI images. This technique makes use of an SVM-based classifier that has advanced feature extraction and selection methods. The method achieves an impressive 98.9% accuracy by using Fast Fourier Transform (FFT) for feature extraction and Minimal-Redundancy-Maximal-Relevance (MRMR) for feature reduction showing the models advantage. Although the method exhibits excellent accuracy and a thorough workflow, its demands and complexity might make it difficult to implement and generalise to different tumour types and datasets, which shows its disadvantage.

Method developed by Zanaty:

In order to measure segmented grey matter and white matter tissues from MRI images, Zanaty proposed a methodology for brain tumour segmentation based on a hybrid type of approach that combines FCM, seed region growing and Jaccard algorithm. The segmentation score S of 90% was achieved on average by this method at 3% and 9% noise levels, as well. (8)

Figure 4 below showing the technique developed by Zanaty.

Creating important regions within an image is the aim of image segmentation. Any mistakes made at this point would have an impact on all future actions. Every segment in a perfectly segmented image should be identical in terms of some requirement, such as texture, colour, or grey level, and adjacent regions should differ significantly in terms of features or characteristics. Accurate white matter (WM) and grey matter (GM) segmentation in MRIs is essential for recognising structural changes linked to disorders of the central nervous system, including diseases such as multiple sclerosis, Alzheimer's disease, and the ageing process in general.

The three steps that make up the algorithm are: 1) FCM algorithm for selecting the best seed; 2) Seed region growing to identify appropriate regions; and 3) Performance measure procedure for combining regions and extracting the final segmentation.

Zanaty presents a new method for segmenting brain tumours that combines the Jaccard algorithm, seed region growth, and fuzzy C-Means (FCM) pre-segmentation for improved accuracy. With noise levels of 3% and 9%, the method still manages to achieve an impressive segmentation score of 90%, demonstrating its ability to distinguish between white matter and grey matter tissues in MRI images. (8) Its effective combination of methods is what makes it appealing; it provides smoothness and speed in segmentation with little initial input. However, there may be difficulties

in some situations due to the method's reliance on exact seed point selection and the possibility of rough pre-segmentation with FCM.

In summary, a review of existing applications shows a wide range of methods—from segmentation to feature extraction and classification—that have been developed for the detection of brain tumours. However, a number of approaches share common flaws, such as insufficient tumour detection accuracy and restricted information extraction, which hinders progress in the field of cancer diagnosis and treatment in general as well as in the identification of brain tumours. Furthermore, one significant error from the reviewed methods is the calculation of overlap metrics, such as the dice similarity index, which is essential for evaluating algorithm accuracy in tumour identification. In the future, filling in these gaps will be essential to improving brain tumour detection systems' accuracy and dependability.

2.3 Review of tools, frameworks and techniques

Programming Languages & Development Environments:

- **Python:**
 - **Advantages:** With a thriving developer and research community, Python provides a wealth of online resources, tutorials, and documentation. Because of its widespread use in the data science and machine learning fields, a large selection of libraries and frameworks are available for all phases of the development process.
 - **Disadvantages:** Python is a great language for experimentation and prototyping, but because it is interpreted, it might not execute as quickly as compiled languages like C/C++. However, this disadvantage can be lessened by optimisations made possible by libraries like NumPy.
- **Jupyter Notebooks:**
 - **Advantages:** Jupyter Notebooks offer an interactive computing environment where explanation text, code, and visualisations are all seamlessly integrated. Because of this, it's a fantastic tool for collaborative research, exploratory data analysis, and teaching.
 - **Disadvantages:** Despite their widespread use, Jupyter Notebooks lack strong version control capabilities, which can cause fragmented codebases and make reproducibility challenging. Nevertheless, JupyterLab is a tool that tries to overcome these shortcomings.

Libraries:

- **TensorFlow:**
 - **Advantages:** TensorFlow, a tool created by Google Brain, is well-liked in both academia and business due to its scalability, flexibility, and production readiness. TensorFlow Extended (TFX) for complete machine learning pipelines and TensorFlow Serving

for model deployment in production are two components of its extensive ecosystem.

- **Disadvantages:** Beginners may find Tensor Flow's detailed API and mathematical graph abstraction challenging to understand. Nevertheless, TensorFlow 2.0 made model development and debugging easier by introducing eager execution and Keras integration.
- **PyTorch:**
 - **Advantages:** The essential approach to programming and dynamic mathematical graph of PyTorch provide flexibility and debugging ease. Both researchers and practitioners have embraced it quickly due to its user-friendly API and language.
 - **Disadvantages:** The system of PyTorch may not be as developed as Tensor Flow's, despite its rising popularity, especially when it comes to scalability and production deployment. But programmes like PyTorch Lightning, which offer high-level abstractions for creating scalable models, seek to close this gap.
- **Scikit-learn:**
 - **Advantages:** Because of its well-known simplicity, consistency, and usability, Scikit-learn is a great option for baseline model development and prototyping. Reliability and reproducibility are guaranteed by its comprehensive documentation and thoroughly tested implementations.
 - **Disadvantages:** Compared to TensorFlow or PyTorch, Scikit-learn offers better support for deep learning models, but it is less effective in traditional machine learning tasks. For complex neural network architectures, integration of additional libraries like TensorFlow or Keras may be needed as a result.

The analysis of relevant frameworks and tools for brain tumour detection systems concludes with a dynamic landscape full of opportunities. The foundation of this environment is Python, which provides a robust and versatile programming language backed by a thriving community. When combined with Jupyter Notebooks' interactive

features, its vast library of machine learning and scientific computing tools offers an effective platform for investigation and testing. Moreover, two significant deep learning frameworks with advantages and disadvantages are TensorFlow and PyTorch. Tensor Flow's extensive system and widespread use make it an excellent choice for large-scale production implementations while PyTorch's dynamic mathematical graph and user-friendly API encourage experimentation and iteration quickly. Scikit-learn, on the other hand, keeps getting better at standard machine learning tasks and offers an intuitive interface and well-documented APIs for quick prototyping and benchmarking.

As a result, the survey conducted highlights how important it is to design brain tumour detection systems with careful consideration and strategic decision making. The utilisation of machine learning and deep learning techniques can be empowered by leveraging the benefits of TensorFlow, PyTorch, Scikit-learn, and Python. This can lead to significant advancements in medical research, especially in the field of brain tumour detection, and improve patient care.

Table below showing the comparison between different languages. (9)

METRICS	PYTHON	JAVA	R
Learning Curve	Quite easy	Comparatively Hard	Hard
Nature	Strong, dynamically typed.	Statically typed	Strong, dynamically typed
Best for	Data analytics, machine learning	Creation of complete dynamic applications	Statistical computing and graphics
Libraries	Yes	Limited	Limited
Advantages	Enhanced Productivity, easy to learn and write.	Easy programming language	Supports various data types
Disadvantages	High memory consumption	Slow and poor performance	No support for dynamic or 3D graphics

3. Legal, social and ethical issues

Legal Considerations:

- **Data Privacy and Protection:** Determine compliance to data protection regulations, such as GDPR or HIPAA, based on the location of data processing and collection. Maintaining legality and trust requires protecting patient data above all else.
- **Intellectual Property:** To safeguard intellectual property rights, take into account applying for patents or copyrights for any innovative algorithms or techniques created throughout the project.
- **Liability:** Clearly state who is responsible for what in the event of an incorrect diagnosis or system failure. Potential legal problems can be reduced by putting strict testing and validation procedures in place.

Ethical Considerations:

- **Informed Consent:** Patients whose data is utilised for system testing or training should give their informed consent. Clearly explain the goal, dangers, and rewards of taking part.
- **Bias and Fairness:** Reduce biases in the way data is gathered and algorithms are created to guarantee equity, particularly when it comes to social and economic or cultural factors that could influence a diagnosis.
- **Patient Autonomy:** Honour patients' freedom to choose their medical care with knowledge. Give patients the choice to refuse data analysis or sharing if they so choose.

Social Considerations:

- **Accessibility:** To avoid escalating inequality in healthcare, make sure underprivileged populations can access the detection system.

- **Equity:** Examine potential differences in healthcare technology accessibility by taking availability and affordability into account for various groups of people.
- **Public Perception:** To control expectations and foster confidence, openly explain to the public the advantages and restrictions of the detection system.

Professional Considerations:

- **Continuing Education:** To make sure the system stays functional and relevant, keep up with developments in machine learning algorithms and medical imaging technology.
- **Collaboration:** Encourage cooperation between researchers and medical professionals to confirm the accuracy and performance of the system in clinical settings.
- **Ethical Standards:** Respect the ethical standards and codes of conduct set forth by important professional associations, such as the IEEE or ACM.

Security Considerations:

- **Data Security:** Use strong security measures, such as encryption and access controls, to guard patient data against breaches or unwanted access.
- **System Integrity:** Review and keep an eye out for any malicious activity or vulnerabilities that might jeopardise the integrity of the detection system on a regular basis.
- **Cybersecurity Awareness:** Teach stakeholders and users about cybersecurity best practices to guard against online threats and guarantee system dependability.

4. Methodology

Project Life Cycle Stages:

I used an agile methodology—which prioritises iterative development, continuous feedback, and adaptability to changing requirements—for the design and implementation of the brain tumour detection system. But I also included important benchmarks in a waterfall plan to give a high-level summary:

- **Project Initiation:** During this phase, the project's requirements, goals, and scope were defined. Setting project constraints, creating user stories, and holding stakeholder meetings and surveys were all important tasks.
- **Planning:** I made a Gantt chart at this phase to show the project schedule, distribute resources, and list dependencies. In addition, I created communication channels, decided on tools and frameworks, and defined the development environment.
- **Design:** I made a prototype during the design stage that included schematic diagrams showing the operation of the Convolutional Neural Network (CNN) and Support Vector Machine (SVM) algorithms. These diagrams made it easy to comprehend the structure and operation of the model by visualising the underlying architecture and the data flow through each algorithm. I made sure that the algorithm design complied with the project's goals and specifications by creating these prototypes. Furthermore, the prototypes functioned as an advantageous means of communication for the involved parties, enabling dialogues and input regarding the suggested methodology. The algorithms were able to be improved and optimised through this iterative design process prior to moving on to the development stage.
- **Development:** Two-week-long continuous sprints made up the development phase. I used TensorFlow and PyTorch to implement machine learning models (including the CNN and SVM algorithms), Scikit-learn for evaluation metrics, and Python and Jupyter Notebook for coding.
- **Testing:** I ran system tests, integration tests, and unit tests during development to make sure the system worked, performed well, and was reliable. I used black-box testing techniques for functional evaluation and white-box testing

techniques for algorithmic code (Python implementations of CNN and SVM in Jupyter Notebooks).

- **Deployment:** Deploying the brain tumour detection system was the next step after the system's functionality was successfully tested and validated in the development environment. With this deployment, the system was moved from the development environment into a production-ready state so that it could be used for its intended purpose and be accessed. The brain tumour detection system was put into practice and set up for deployment using the Jupyter Notebook platform. Jupyter Notebook is a great tool for development and prototyping because it offers an interactive computing environment that makes it easy to integrate code, documentation, and visualisations. The main step in the deployment process was to arrange and structure the code in a Jupyter Notebook environment, making sure that it was understandable and easy to read by using comments and documentation. Throughout the code, comments have been added to offer clarifications, background information, and direction for upcoming development and maintenance tasks.
- **Maintenance:** After the system was deployed, I kept an eye on its functionality and made any necessary updates and improvements.

Methodology and Development Techniques:

The Agile methodology was selected due to its adaptability, flexibility, and emphasis on providing stakeholders with value in small steps. I was able to adapt to changing requirements, take feedback into account, and provide small improvements throughout the development process by segmenting the project into manageable sprints. Agile's iterative structure enabled me to gradually improve the system and make sure it complied with user requirements and project goals. A waterfall plan's important milestones also offered a high-level road map for managing the project and monitoring advancement towards certain objectives.

Implementation:

Ensuring accessibility and usability of the Jupyter Notebook environment required careful consideration of user interaction. The notebook layout was intended to walk users through the project's different phases, offering concise explanations and visual aids at each turn. Markdown cells were also thoughtfully positioned throughout the notebook to offer instructions, explanations, and context, which improved user understanding, this was crucial since it made the analysis process run more smoothly and productively for users. To arrange the content and improve user navigation, clear headings, subheadings, and formatting were used. Additionally, using tools like Matplotlib, visualisations created during the phases of data exploration, model training, and evaluation were integrated right into the notebook. As a result, users could see important trends, insights, and model outputs right within the notebook environment—no need for additional software or tools for visualisation.

Testing:

A thorough testing methodology was used throughout the brain tumour detection system's development process to guarantee its effectiveness, dependability, and quality. To verify each component and algorithm's functionality and behaviour separately, unit tests were run on the CNN and SVM implementations in the Jupyter Notebook environment using Python. Integration testing confirmed that various system modules and components, such as the CNN and SVM algorithms, data pre-processing, feature extraction, and model evaluation, worked together seamlessly. After that, system testing was carried out to compare the system's overall performance, dependability, and compliance with both functional and non-functional requirements to predetermined requirements and specifications. The system's quality, dependability, and usability were guaranteed by using a multifaceted testing strategy, which enhanced the system's ability to correctly identify brain tumours.

5. Design

I used a modular and scalable architecture for the final software structure of my brain tumour detection system in order to successfully fulfil the project requirements. I'll go over all aspects of the software architecture in the sections that follow, including the User Interface (UI), Infrastructure, Functionality, Content creation, Algorithm development, and other relevant details.

User Interface:

My system's user interface was made with ease of use, accessibility, and simplicity in mind for medical professionals. I made use of the Jupyter Notebook environment's web-based interface, which offered an interactive platform for data analysis and model building. It's important to emphasise the design principles that guided the development of the user interface (UI), even though it was primarily developed within the Jupyter Notebook environment. Medical professionals in particular were the end users considered when designing the user interface. To make sure users could easily navigate the system, a focus on accessibility, simplicity, and understanding was made. While button controls and other integrated response mechanisms are absent from the Jupyter Notebook environment, efforts have been made to improve user experience with features like interactive data visualisation. The UI prioritised user-centric design in order to make data analysis and model building easier for healthcare professionals.

Infrastructure:

Jupyter Notebook and Python were used in the development and deployment of my system; no other cloud-based platforms or services were used. Jupyter Notebook was used to manage all computational tasks, data storage, and application hosting. This method allowed for the smooth integration of data processing, model development, and algorithms on a single platform by providing simplicity and ease of development. Although cloud-based platforms such as AWS or Google Cloud Platform might provide advantages in terms of scalability and accessibility, they were not employed in this project. This choice guaranteed the effective use of computational resources while avoiding needless complexity.

Functionality:

My system's primary functions were evaluation, prediction, pre-processing, training models, and data collection. The Python programming language and a number of libraries, including NumPy, Pandas, TensorFlow, and Scikit-learn, were used to implement these features. MRI image datasets with both tumour and non-tumour cases were gathered for data collection, and pre-processing methods included data augmentation, normalisation, and resizing. In order to identify brain tumours based on image features, model training used Convolutional Neural Networks (CNN) and Support Vector Machines (SVM) algorithms. Together, these features enhanced the system's capacity to reliably and effectively analyse and interpret medical imaging data.

Algorithm Development:

A key component of the project was the creation of the CNN and SVM algorithms. I carefully implemented and optimised these algorithms to get the best results using Jupyter Notebook and Python. I was able to investigate several tactics, such as hyper parameter tuning and transfer learning, to improve model accuracy and generalisation by utilising Scikit-learn and TensorFlow for model development and evaluation. Through ongoing algorithmic improvement using labelled MRI image datasets, my goal was to identify complex patterns and traits linked to brain tumours, enhancing the system's capacity for reliable diagnosis.

Other Considerations:

During the software development process, I gave scalability, maintainability, and security aspects top priority in addition to addressing core functionalities. Cautiously preventing unwanted access to private patient data. Furthermore, the implementation of best practices in software engineering guaranteed the codebase's readability, maintainability, and quality, which in turn made it easier to make future improvements and modifications. I established a strong basis for the brain tumour

detection system's long-term sustainability and success by proactively taking these factors into account.

In conclusion, the project's goals were successfully understood thanks to the implementation of a modular, scalable, and user-centric architecture. I have created a brain tumour detection system that is both dependable and efficient, meeting the various requirements of medical professionals and researchers through careful planning and execution. The system's performance and usefulness in clinical settings will continue to be improved by ongoing optimisation and refinement efforts, confirming the system's importance in the field of medical imaging analysis.

6. Tools and implementation

6.1 Tools

Python:

Python was specifically chosen as the main programming language for this project because it is a flexible language that can be tailored to meet the various requirements of developing a brain tumour detection system. Python is a great choice for a wide range of tasks, from data collection to model evaluation, because of its extensive system and versatility. The choice to use Python complies with a number of project requirements:

- **Data Collection & Pre-processing:** NumPy and Pandas are just two of the many libraries available in Python that make pre-processing and data collection more effective. These libraries provide strong tools for working with various MRI image datasets and applying necessary pre-processing methods, like normalisation and resizing.
- **Data Augmentation:** Because of Python's flexibility, data augmentation techniques can be integrated with ease. This is crucial for increasing the diversity and size of the dataset, especially for Convolutional Neural Network (CNN) training. You can investigate various augmentation strategies to enhance the robustness of their models by utilising libraries such as imgaug and Augmentor.
- **Model Training & Evaluation:** Because of its widespread use in the machine learning community, Python is well-supported by deep learning frameworks such as TensorFlow and PyTorch. You can quickly implement and train CNN & SVM models for image-based tumour detection by utilising these frameworks. Furthermore, a comprehensive model evaluation based on project requirements, such as accuracy, F1 score, precision, and sensitivity, is made easier by Python's rich system of evaluation metrics libraries, which includes Scikit-learn.

Jupyter Notebook:

The project's efficiency and collaborative capabilities are further enhanced by the choice of Jupyter Notebook as the development environment. The interactive computing environment provided by Jupyter Notebook is in line with the project's focus on collaborative research, iterative model development, and descriptive data analysis. Data scientists and healthcare professionals can work together more smoothly thanks to Jupyter Notebook, which combines code, graphics, and narrative text into a single document.

- **Interactive Document:** You can explore and visualise data, test various algorithms, and continuously enhance models in real-time with Jupyter Notebook's interactive interface. In particular, this interactivity speeds up the development cycle and increases productivity in the early phases of model prototyping and experimentation.
- **Collaborative Research:** Data scientists, medical professionals, and subject matter experts can collaborate diverse between fields thanks to Jupyter Notebook's sharing and collaboration features. Stakeholders can ensure the development of robust and clinically relevant tumour detection systems by effectively collaborating, contributing domain expertise, and providing feedback on model predictions through the sharing of executable code, visualisations, and insights.

The project satisfies its functional and non-functional requirements by utilising Python and Jupyter Notebook. This not only facilitates effective data processing, model training, and evaluation, but also encourages creativity.

Building this project has given me the priceless opportunity to further develop a number of skills. I've made use of my prior experience with Python programming, taking advantage of its adaptability and robust library system for tasks varying from gathering data to assessing models. I've also gained a deeper understanding of evaluation metrics libraries like Scikit-learn and deep learning frameworks like TensorFlow and PyTorch.

6.2 Implementation

Support Vector Machine (SVM) Algorithm Implementation:

For classification tasks, the Support Vector Machine (SVM) algorithm is an effective supervised learning technique. SVM can efficiently classify MRI images into tumour and non-tumour classes based on extracted features in the context of brain tumour detection.

- **Data Pre-processing:** Pre-processing the MRI image data is the initial stage in putting the SVM algorithm into practice. This entails loading the dataset of brain MRI pictures and getting the information ready for testing and training. Pre-processing tasks may involve resizing the images to a uniform size, converting them to grayscale, and flattening the image arrays to create feature vectors. The images are usually stored as pixel counts in a matrix format. The SVM classifier receives these feature vectors as input.

The pre-processing steps are executed as follows in the code that is provided: To store the image data and associated labels, two lists, 'X' and 'Y', are made. The code iterates through the images in the corresponding folder for each class, resizes the image to 200x200 pixels, reads the image using OpenCV's cv2.imread function, and appends the image data to the 'X' list and the matching label to the 'Y' list. Next, NumPy arrays are created from the 'X' and 'Y' lists. The image data in 'X' is transformed into a 2D array, with a flattened image (1D feature vector) represented by each row.

```
import os
import cv2

dataset_path = r'D:\UON SOFTWARE ENGINEERING Y3\FINAL YEAR PROJECT\FYP\Dataset\Training'

X = []
Y = []
for cls in classes:
    pth = os.path.join(dataset_path, cls)
    for j in os.listdir(pth):
        img = cv2.imread(os.path.join(pth, j), 0)
        img = cv2.resize(img, (200,200))
        X.append(img)
        Y.append(classes[cls])

X = np.array(X)
Y = np.array(Y)

X_updated = X.reshape(len(X), -1)

np.unique(Y)

pd.Series(Y).value_counts()

X.shape, X_updated.shape
```

- **Data Splitting:** The "train_test_split" function from the Scikit-learn library is used to split the pre-processed data into training and testing sets. In order to evaluate the model's generalisation performance, this step makes sure that it is trained on a subset of the data and assessed on unseen data. The data is divided into training and testing sets using the train test split function in the code, with a test set size of 20% (determined by the test_size parameter).

SPLIT DATA

```
xtrain, xtest, ytrain, ytest = train_test_split(X_updated, Y, random_state=10, test_size=.20)
```

```
xtrain.shape, xtest.shape
```

- **Model Training:** The SVM classifier is trained using the "fit" method on the training data after the data has been split. By identifying the ideal hyperplane that maximises the margin between the classes, the SVM algorithm gains the ability to distinguish between the various classes in the feature space during training. The SVM classifier (SVC class from Scikit-learn) instance is created in the code, and the model is trained by calling the fit method with the training data (xtrain and ytrain).

FEATURE SELECTION: PCA

```
from sklearn.decomposition import PCA
```

```
print(xtrain.shape, xtest.shape)
```

```
pca = PCA(.98)  
# pca_train = pca.fit_transform(xtrain)  
# pca_test = pca.transform(xtest)  
pca_train = xtrain  
pca_test = xtest
```

```
import warnings  
warnings.filterwarnings('ignore')
```

```
lg = LogisticRegression(C=0.1)  
lg.fit(xtrain, ytrain)
```

```
sv = SVC()  
sv.fit(xtrain, ytrain)
```

- **Model Evaluation:** Using the testing data, the SVM classifier's performance is assessed following training. Various performance metrics, including accuracy, F1 score, precision, and sensitivity, are computed to evaluate the model's efficiency in classifying brain tumour images. Predictions are made on the test data using the trained model. The code makes predictions on the test data (xtest) using the trained SVM models predict method. Using functions from the Scikit-learn library (accuracy_score, f1_score, precision_score, and recall_score), these predictions are compared with the actual labels (ytest) to determine performance metrics like accuracy, F1 score, precision, and sensitivity.

```
accuracy = accuracy_score(ytest, pred)
f1 = f1_score(ytest, pred)
precision = precision_score(ytest, pred)
sensitivity = recall_score(ytest, pred)

print("Accuracy:", accuracy)
print("F1 Score:", f1)
print("Precision:", precision)
print("Sensitivity (Recall):", sensitivity)
```

In conclusion, pre-processing the data, dividing it into training and testing sets, training the SVM classifier, and assessing its performance on data comprise the implementation of the SVM algorithm. These procedures will help us create a useful model for MRI image-based brain tumour detection.

Convolutional Neural Network (CNN) Algorithm Implementation:

Among the deep learning models, convolutional neural networks (CNNs) are especially well-suited for image classification applications. CNNs can automatically identify hierarchical features from MRI images in the context of brain tumour detection, allowing for precise classification of tumour and non-tumor regions.

- **Data loading and Pre-processing:** Loading and pre-processing the MRI image data is the initial stage in implementing the CNN algorithm, much like the SVM algorithm. The photos are divided into training, validation, and testing sets after being scaled to a standard size and turned into grayscale. To improve the model's resilience, additional data augmentation methods like

rotation, shifting, and flipping are used. The following is how the pre-processing and data loading procedures are completed in the code that is provided:

To load and pre-process the MRI image data from the designated dataset path, a custom function called `load_data` is defined. The picture data and associated labels are returned as NumPy arrays by this function, which resizes the images to 200x200 pixels. The `load_data` function is then called to load and pre-process the data after the dataset path and class labels are specified. The `train_test_split` function from Scikit-learn next divides the pre-processed data into training, validation, and testing sets. In order to conform to the CNN input format, the image data is reshaped to include a channel dimension (grayscale images have one channel).

- **Model Architecture:** TensorFlow Keras API is used to design CNN model architecture. Max-pooling layers come after convolutional layers, which are responsible for extracting and capturing physical characteristics from the input images. Additionally, the architecture will have dense layers for classification and dropout layers to avoid overfitting. The CNN model architecture is defined in the code as follows: `Sequential` from Keras is used to create a sequential model.

The following layers make up the model:

Max-pooling layer with 2x2 pool size comes after a convolutional layer with 32 filters, 3x3 kernel size, and ReLU activation.

After a max-pooling layer with a 2x2 pool size, there is a convolutional layer with 64 filters, a 3x3 kernel size, and ReLU activation.

The max-pooling layer has a 2x2 pool size and is followed by a convolutional layer with 128 filters, a 3x3 kernel size, and ReLU activation.

Dense layer with ReLU activation and 128 units.

In order to lessen overfitting, use a 0.5 dropout layer rate.

For binary classification, output a dense layer with a single unit and sigmoid activation.

The accuracy metric, binary cross-entropy loss, and Adam optimizer are used in the compilation of the model.

```
dataset_path = r'D:\UOW SOFTWARE ENGINEERING Y3\FINAL YEAR PROJECT\FYP\Dataset\Training'
classes = {'no_tumour':0, 'yes_tumour':1}
```

```
X, Y = load_data(dataset_path, classes)
X = X.reshape(X.shape[0], X.shape[1], X.shape[2], 1)
```

```
xtrain, xtest, ytrain, ytest = train_test_split(X, Y, random_state=10, test_size=.20)
```

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state=10, test_size=.20)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, random_state=10, test_size=.15)
```

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(200, 200, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
```

```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
datagen = ImageDataGenerator(rotation_range=20,
                             width_shift_range=0.2,
                             height_shift_range=0.2,
                             shear_range=0.2,
                             zoom_range=0.2,
                             horizontal_flip=True,
                             fill_mode='nearest')
```

- Model Training:** The enhanced training data is used to train the CNN model, and the validation data is used to verify the model. The model gains the ability to identify appropriate characteristics from MRI pictures and divide them into classes for tumours and non-tumours during training. Validation loss and accuracy are used to track the training progress. The following actions are taken for model training in the code:

Rotation, shifting, shearing, zooming, and horizontal flipping are among the data augmentation parameters that are applied when creating an instance of the ImageDataGenerator class from Keras.

The augmented data generator is passed for the training and validation sets of data when the model's fit method is invoked. The number of training iterations is specified by the epoch's parameter, while the steps_per_epoch parameter guarantees that all training data is used in each epoch.

Matplotlib is used to plot the training and validation loss and accuracy curves, which show how well the model performs during training.

```
history = model.fit(datagen.flow(x_train, y_train, batch_size=32),
                    steps_per_epoch=len(x_train) / 32,
                    epochs=10,
                    validation_data=(x_val, y_val))

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

- **Model Evaluation:** The CNN model's performance is assessed using testing data following training. The model is trained on test data, and its classification performance is evaluated by computing a number of performance metrics, including accuracy, precision, F1 score, and sensitivity. The following actions are taken for model evaluation in the code below:

The predicted probabilities for the test data (x_{test}) are obtained using the trained CNN models predict method.

The predicted probabilities are transformed into binary predictions (0 for "no tumour" and 1 for "yes tumour") using a classification threshold of 0.6.

Using functions from the Scikit-learn library (`accuracy_score`, `f1_score`, `precision_score`, and `recall_score`), the binary predictions are compared with the actual labels (y_{test}) to calculate performance metrics like accuracy, F1 score, precision, and sensitivity.

```
y_pred_proba = model.predict(x_test)

threshold = 0.6

y_pred_binary = (y_pred_proba > threshold).astype(int)

accuracy = accuracy_score(y_test, y_pred_binary)
f1 = f1_score(y_test, y_pred_binary)
precision = precision_score(y_test, y_pred_binary)
sensitivity = recall_score(y_test, y_pred_binary)

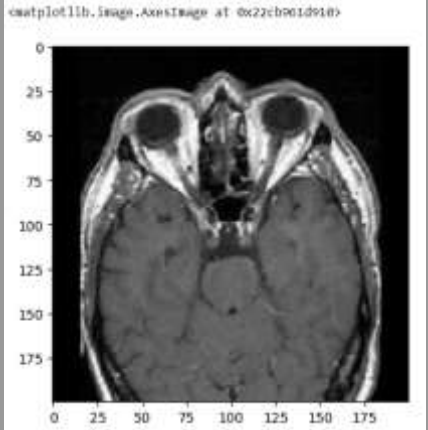
print("Accuracy:", accuracy)
print("F1 Score:", f1)
print("Precision:", precision)
print("Sensitivity (Recall):", sensitivity)
```

To summarise, the process of implementing the CNN algorithm entails loading and pre-processing the MRI image data, developing and perfecting a CNN model architecture, and assessing the model's performance using test data that hasn't been seen yet. These procedures will help us create a trustworthy and accurate CNN-based brain tumour detection model.

7. Testing

Support Vector Machine (SVM) Algorithm Testing:

TEST TYPE	TEST INFO	INPUT	OUTPUT	PASS/ FAIL
Generalisation test	Evaluating the SVM models performance on unseen data from the testing dataset to assess its ability to generalise.	<pre> pred = sv.predict(xtest) accuracy = accuracy_score(ytest, pred) f1 = f1_score(ytest, pred) precision = precision_score(ytest, pred) sensitivity = recall_score(ytest, pred) print("Accuracy:", accuracy) print("F1 Score:", f1) print("Precision:", precision) print("Sensitivity (Recall):", sensitivity) </pre>	<pre> Accuracy: 0.963265306122449 F1 Score: 0.9728096676737159 Precision: 0.9640718562874252 Sensitivity (Recall): 0.9817073170731707 </pre>	Pass
Misclassified analysis test	Analysing misclassified samples to identify any patterns that the model struggles to classify correctly.	<pre> print("Total Misclassified Samples: ",len(misclassified[0])) print(pred[36],ytest[36]) </pre>	<pre> Total Misclassified Samples: 9 0 1 </pre>	Pass

<p>Visualisation Test</p>	<p>Ensuring that the visualisations are correctly and match the expected outcome.</p>	<pre>plt.imshow(X[0], cmap='gray')</pre>		<p>Pass</p>
<p>Pre-processing Test</p>	<p>Testing the behaviour of the model using Normalisation.</p>	<pre>print(xtrain.max(), xtrain.min()) print(xtest.max(), xtest.min()) xtrain = xtrain/255 xtest = xtest/255 print(xtrain.max(), xtrain.min()) print(xtest.max(), xtest.min())</pre>	<pre>255 0 255 0 1.0 0.0 1.0 0.0</pre>	<p>Pass</p>
<p>Feature Selection Test</p>	<p>Implementing feature selection using PCA to identify the most relevant features for SVM model.</p>	<pre>print(xtrain.shape, xtest.shape) pca = PCA(.98) pca_train = pca.fit_transform(xtrain) pca_test = pca.transform(xtest) pca_train = xtrain pca_test = xtest</pre>	<pre>(977, 40000) (245, 40000)</pre>	<p>Pass</p>

Convolutional Neural Network (CNN) Algorithm Testing:

TEST TYPE	TEST INFO	INPUT	OUTPUT	PASS/ FAIL
Generalisation test	Evaluating the CNN models performance on unseen data from the testing dataset to assess its ability to generalise.	<pre>print("Accuracy:", accuracy) print("F1 Score:", f1) print("Precision:", precision) print("Sensitivity (Recall):", sensitivity)</pre>	<pre>Accuracy: 0.9224489795918367 F1 Score: 0.9446064139941691 Precision: 0.9050279329608939 Sensitivity (Recall): 0.9878048780487805</pre>	Pass
Validation Loss Analysis Test	Analysing the validation loss curve during training to identify any signs of overfitting.	<pre>history = model.fit(datagen.flow(x_train, y_train, batch_size=32), steps_per_epoch=len(x_train) / 32, epochs=10, validation_data=(x_val, y_val)) plt.plot(history.history['loss'], label='Training Loss') plt.plot(history.history['val_loss'], label='Validation Loss') plt.xlabel('Epoch') plt.ylabel('Loss') plt.legend() plt.show() plt.plot(history.history['accuracy'], label='Training Accuracy') plt.plot(history.history['val_accuracy'], label='Validation Accuracy') plt.xlabel('Epoch') plt.ylabel('Accuracy') plt.legend() plt.show()</pre>		Pass

<p>Visualisation Test</p>	<p>Ensuring that the visualisations are correctly and match the expected outcome.</p>	<pre>plt.figure(figsize=(12, 8)) for i in range(9): plt.subplot(3, 3, i + 1) plt.imshow(x_test[i].reshape(200, 200), cmap='gray') plt.title(f"Predicted: {y_pred_binary[i]}, Actual: {y_test[i]}") plt.axis('off') plt.show()</pre>		<p>Pass</p>
<p>Pre-processing Test</p>	<p>Testing the behaviour of the model using different data augmentation techniques.</p>	<pre>datagen = ImageDataGenerator(rotation_range=20, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest')</pre>	<p>No output needed</p>	<p>Pass</p>

Conclusion:

The Support Vector Machine (SVM) algorithm outperforms the Convolutional Neural Network (CNN) algorithm in brain tumour detection, according to the performance results. With an accuracy of 96.33%, the SVM algorithm outperforms the CNN algorithm, which comes in at 92.24%. Furthermore, the F1 score of the SVM algorithm is 97.28%, higher than the F1 score of the CNN algorithm, which is 94.46%. The SVM algorithm has a higher precision (96.41%) than the CNN algorithm (90.50%), which is a measure of correctly predicted positive cases. In contrast, the SVM algorithm achieves a sensitivity of 98.17%, while the CNN algorithm achieves a slightly higher sensitivity (or recall) of 98.78%. This implies that although both algorithms are effective in identifying brain tumours, the SVM algorithm performs better overall in terms of precision, F1 score, and accuracy, suggesting that it might be a better fit for the detection of Brain tumours. The superior performance of the SVM algorithm can be attributed to its ability to find an optimal hyperplane that maximises the margin between different classes and its resilience when processing high-dimensional data. High accuracy in brain tumour detection is largely dependent on SVM's ability to classify image data, which can be difficult to separate due to the non-linear nature of the features. Support vectors enable SVM to generalise well to new data, as evidenced by its high accuracy of 96.33%. However, even though the CNN algorithm is frequently employed for image classification tasks, it might not be able to fully take advantage of the complex patterns found in MRI images. CNNs usually use convolutional layers to learn structured representations of features, which may not pick up on minute details that are essential for telling tumour from non-tumour regions in brain images. Consequently, the accuracy of the CNN is 92.24%, which is marginally less than that of the SVM.

7.1 Test coverage

Two beneficial methods, each with its own goals and focus, are used to evaluate software systems: black box testing and white box testing. Both black box and white box testing are essential to ensuring that machine learning models that are used for brain tumour detection, such as the two algorithms I have developed which are support vector machine (SVM) and convolutional neural network (CNN), meet the requirements. Functional testing, sometimes referred to as black box testing, looks at how the system behaves without taking into account its internal operations or implementation details. As stated in the requirements, it instead concentrates on confirming whether the system operates as intended based on its inputs and outputs. White box testing, also known as structural testing, on the other hand, looks at the system's internal components, such as its code, algorithms, and data structures. The goal of this strategy is to find any implementation errors or problems that might have an impact on the behaviour of the system.

Generalisation Test (Performance Requirement):

- **Black Box Testing:** In this test, the generalisation capabilities of the CNN and SVM models are assessed using previously unseen data. Black box testing ignores the internal model parameters and training procedure in favour of concentrating on the models' performance metrics, such as accuracy, F1 score, precision, and sensitivity. Black box testing for the CNN model entails passing previously unseen MRI images through the trained model and assessing how well it can identify the images as tumours or non-tumours. In the same way, black box testing evaluates the SVM model's performance on the unknown data by calculating metrics such as accuracy and F1 score without exploring the finer details of its decision boundaries.
- **White Box Testing:** White box testing entails assessing the decision boundaries that the SVM model learns or looking at the layers and events in the CNN model to see how they affect generalisation. White box testing for the SVM model entails examining the hyperplane and support vectors to learn more about how the model distinguishes between the tumour and non-tumour classes.

Misclassified Analysis Test (Evaluation of Model Requirement):

- **Black Box Testing:** In black box testing, samples that have been incorrectly classified are analysed to find any patterns or traits that the models have trouble correctly classifying. Without going into the internal operations of the models, this analysis aids in pointing out potential improvement areas. Misclassified analysis, which applies to both the CNN and SVM models, involves analysing the traits of the misclassified samples in order to spot recurring themes or characteristics that might point to potential areas for model development.
- **White Box Testing:** In white box testing, the misclassified samples are thoroughly examined, and features or events in the SVM or CNN models that influence misclassifications would be examined. This more thorough examination offers insights into possible enhancements to training methods or model architecture. White box testing for the CNN model would entail examining the patterns in particular neurons or layers that are in charge of misclassifications. In the same manner, white box testing for the SVM model would involve examining the support vectors and how close they are to the decision boundary in order to determine why some samples are incorrectly classified.

Visualisation Test (User Interface requirement):

- **Black Box Testing:** Black box testing makes sure that any visualisations produced by the models correspond to what is anticipated. MRI images for the CNN model or decision boundaries for the SVM model may be examples of this.
- **White Box Testing:** Examining the internal calculations and representations that produce the visualisations is known as "white box" testing. White box testing for the CNN model, for instance, would entail examining the patterns in MRI images. Likewise, analysing the mathematical formulation of the decision boundaries and comprehending the various features' contributions to

the classification process would also be part of white box testing for the SVM model.

Pre-processing Test (Data Pre-processing Requirement):

- **Black Box Testing:** Black box testing assesses how pre-processing methods, like normalisation, affect the performance metrics of the models. Instead of analysing the internal operation of the models, it concentrates on evaluating the effects of various pre-processing techniques on the models' overall performance.
- **White Box Testing:** Analysing how pre-processing methods affect the internal representations and analysis of the models would be part of white box testing. White box testing, for example, would look at how feature scaling impacts decision boundaries and the optimisation process in the SVM model. White box testing for the CNN model would entail analysing the effects of various pre-processing methods on the patterns in the convolutional layers and the learning process as a whole.

7.2 Test methodology

A thorough methodology was used to test the output of the SVM and CNN algorithms for brain tumour detection in order to guarantee the system's dependability, accuracy, and usability. A variety of test types, including generalisation tests, misclassified analysis, visualisation tests, data pre-processing tests, and feedback gathering from non-expert users, were used in the testing process. Every kind of test had a distinct function in assessing various facets of the algorithms' usability and performance.

Generalisation Test:

The purpose of the generalisation test was to evaluate the SVM and CNN algorithms' performance on previously unseen data, indicating how well they could generalise to new cases. Test cases for both algorithms consisted of previously unpublished MRI images, and performance metrics like accuracy, F1 score, precision, and sensitivity were calculated. This test made sure the models could correctly identify brain tumours in situations outside of the training set. This test was crucial for determining how well the models could apply what they had learned to new, untested cases—a critical skill for real-world use.

Misclassified Analysis Test:

Misclassified analysis test entailed looking at samples that the SVM and CNN models misclassified in order to find any patterns or traits that the models had trouble correctly classifying. Potential flaws or areas where the algorithms could be strengthened could be found by examining samples that were incorrectly classified. The models were improved in terms of accuracy and robustness by this test. For instance the CNN models previous accuracy was 76%, however after improving the model a 92% accuracy was achieved. The models are continuously improved through this iterative refinement process, resulting in a brain tumour detection method that is more precise and trustworthy.

Data Pre-processing Test:

The test of data pre-processing assessed how various pre-processing techniques, including normalisation and data augmentation, affected the SVM and CNN algorithms' performance. The efficiency of pre-processing methods in enhancing model performance was evaluated by contrasting the performance metrics of models trained with and without pre-processing. The models are improved in their ability to detect brain tumours, for instance, by adding more samples to the training dataset through data augmentation and standardising input features through normalisation. The algorithms are prepared for optimal performance in real-world scenarios thanks to this careful assessment of pre-processing techniques, which improves their usability and effectiveness in clinical environments.

Visualisation Test:

The purpose of the visualisation test was to assess the SVM and CNN models' visual representations. MRI images and decision boundaries for the CNN model were visualised to make sure they matched the anticipated results. In the same way, decision boundaries, MRI images and support vectors for the SVM model were visually analysed. Visualisation tests offered insights into the models' decision-making processes and assisted in validating the internal operations of the models. The purpose of the visualisation test was not limited to evaluating visual representations instantly; it was also employed to improve the interpretability and transparency of the model. The test enabled a better comprehension of the behaviour of the models by offering user-friendly visualisations of MRI images, decision boundaries, and support vectors.

Feedback Collection:

In order to verify the SVM and CNN algorithms' practicality and efficacy in real-world scenarios, feedback gathering was essential. In this case, peers, mentors, and supervisors working in the fields of neuroscience and medical imaging were the main sources of feedback. These people offered insightful information about the

algorithms' technical accuracy, clinical applicability, and practical usefulness. Technical comments on the algorithms were provided by peers, mentors, and supervisors who were experienced in medical imaging and machine learning. Based on their knowledge of the subject, they evaluated the requirements of the algorithms. Their input was very helpful in pinpointing any technical issues or areas that needed work in the implementation, training, or assessment of the algorithms. All of these sources' feedback was gathered using a variety of techniques, such as questionnaires, interviews, and surveys. Subsequently, the input was examined, sorted, and utilised to guide iterative enhancements to the algorithms and general system performance. The SVM and CNN algorithms were continuously improved and optimised thanks to this iterative feedback loop, which made sure they met the standards for reliable brain tumour detection in clinical settings.

8. Conclusions and reflections

Through this project, I was able to effectively design and implement a system for the detection of brain tumours using cutting-edge machine learning algorithms, specifically Support Vector Machine (SVM) and Convolutional Neural Network (CNN). The main goal was to improve brain tumour diagnosis, which would increase the precision and effectiveness of medical assessments and meet the urgent need for early tumour detection in terms of predictions and patient outcomes.

With an astounding accuracy of 96.33% and an F1 score of 97.28%, the SVM algorithm proved to be the better performer than the CNN algorithm, which had an accuracy of 92.24% and an F1 score of 94.46%. The SVM is very good at classifying high-dimensional image data because of its capacity to identify an ideal hyperplane that maximises the margin between various classes. Its superior performance in brain tumour detection can be attributed to its resistance to the irregular features present in MRI images. Although CNN models are widely used for image classification applications, it's possible that the CNN algorithm in this project had trouble identifying the minute details and complex patterns needed to differentiate between tumorous and non-tumorous areas in brain images. However, the CNN algorithm demonstrated a respectable 92.24% accuracy and 98.78% sensitivity, suggesting room for further development and optimisation.

I used an agile methodology that allowed for constant delivery of value to stakeholders and allowed for flexibility and adaptability throughout the project lifecycle. Because the agile methodology is iterative, I was able to adapt to changing requirements, take feedback into account, and improve the system little by little to make sure it met user needs and project objectives.

Careful consideration of user interface (UI) and user experience (UX) design characterised the implementation phase. I developed a user-friendly platform that seamlessly integrated code, visualisations, and narrative text by utilising Jupyter Notebook's interactive computing environment. An essential component of the project was thorough testing, which guaranteed the accuracy, dependability, and usability of the system. A thorough testing approach was used, which included tests for

generalisation, misclassified analysis, data pre-processing, and visualisation. Peer, mentor, and supervisor feedback was also very helpful in terms of technical accuracy, clinical relevance, and practical functionality.

Upon analysing the project lifecycle, a number of strengths and weaknesses became apparent. The methodical approach to algorithm development, which included thorough pre-processing, feature extraction, and optimisation techniques, was one of the standout features. The potential of machine learning in medical imaging analysis was demonstrated by the robust and accurate brain tumour detection achieved through the use of sophisticated algorithms such as CNN and SVM. Nevertheless, the project encountered obstacles and constraints as well. One drawback was the lack of extensive and varied MRI image datasets, which would have enhanced the models' capacity for generalisation even more. Furthermore, scalability and real-time deployment may be affected in some situations by the computational resources needed for training and optimising deep learning models like CNN. It is possible to investigate a number of directions for future development and enhancement. Expanding and diversifying the datasets could improve the robustness and generalisability of the models to various tumour types and methods of imaging. Furthermore, investigating mixed methods that combine the advantages of SVM and CNN algorithms may produce even better results in terms of accuracy and performance.

From an individual perspective, this project gave me priceless chances to learn and grow my skills. I obtained practical experience with the application of sophisticated machine learning algorithms, methods for pre-processing data, and approaches for evaluating models. I also gained a greater understanding of the ethical issues, legal consequences, and impact on society related to brain tumour detection systems and medical imaging analysis. In addition, I gained a thorough understanding of model evaluation techniques and metrics, including F1 score, accuracy, precision, and recall. By evaluating the CNN and SVM models' performance using these metrics, I was able to pinpoint their advantages, disadvantages, and potential development areas, encouraging a data-driven approach to model improvement.

In summary, this project effectively created a brain tumour detection system that makes use of machine learning algorithms to improve medical assessments and support worldwide efforts to prevent and treat cancer. Although the SVM algorithm performed better, the CNN algorithm also produced encouraging results, suggesting that it could be further improved. Through constant iteration, integration of feedback, and investigation of unique methods, this system has the potential to prove a valuable tool in the battle against brain tumours, ultimately enhancing patient outcomes and quality of life. Furthermore, the ability of this brain tumour detection system to improve patient outcomes and aid in the worldwide fight against cancer is ultimately what will determine its true significance. The technology has the potential to save lives, lessen the overall impact of the illness, and enhance the quality of life for those who are afflicted and their families by facilitating early detection and precise diagnosis. This project serves as a testament to the beneficial impact that cross-disciplinary work and the relentless pursuit of innovation in healthcare as advances in computational methods and medical imaging technology continue.

9. References

1. Lotlikar VS, Satpute N, Gupta A. Brain Tumor Detection Using Machine Learning and Deep Learning: A Review. *Curr Med Imaging*. 2022;18(6):604-622. doi: 10.2174/1573405617666210923144739. PMID: 34561990.
2. Pandey, S., Goyal, A. and Vansuha, D. (2021) 'Brain Tumour Detection Using Cnn', *Turkish journal of computer and mathematics education*, 12(11), pp. 4597–4603.
3. www.iarc.who.int. (n.d.). *Cancer Topics – IARC*. [online] Available at: <https://www.iarc.who.int/cancer-topics/>.
4. Tulip Mazumdar (2023) Safer brain surgery using AI possible within two years Available at: <https://www.bbc.co.uk/news/health-66921926> Date Accessed: 2nd November 2023.
5. Cancer Research UK. (2015). Brain, other CNS and intracranial tumours statistics. [online] Available at: <https://www.cancerresearchuk.org/health-professional/cancer-statistics/statistics-by-cancer-type/brain-other-cns-and-intracranial-tumours#heading=Two>.
6. S. Damodharan and D. Raghavan, "Combining tissue segmentation and neural network for brain tumor detection," *International Arab Journal of Information Technology*, vol. 12, no. 1, pp. 42–52, 2015.
7. M. Alfonse and A.-B. M. Salem, "An automatic classification of brain tumors through MRI using support vector machine," *Egyptian Computer Science Journal*, vol. 40, pp. 11–21, 2016.
8. E. A. Zanaty, "Determination of gray matter (GM) and white matter (WM) volume in brain magnetic resonance images (MRI)," *International Journal of Computer Applications*, vol. 45, pp. 16–22, 2012.
9. admin (2021). Top Programming Languages 2021: By Type and Comparison. [online] ISHIR | Software Development India. Available at: <https://www.ishir.com/blog/36749/top-75-programming-languages-in-2021-comparison-and-by-type.htm>.
10. Kapoor, L. and Thakur, S., 2017, January. A survey on brain tumor detection using image processing techniques. In *2017 7th international conference on cloud computing, data science & engineering-confluence* (pp. 582-585). IEEE.

11. Sadad, T. *et al.* (2021) “Brain tumor detection and multi-classification using advanced deep learning techniques,” *Microscopy research and technique*, 84(6), pp. 1296–1308. Available at: <https://doi.org/10.1002/jemt.23688>.
12. Singh, A., 2016, March. Review of brain tumor detection from MRI images. In *2016 3rd International conference on computing for sustainable global development (INDIACom)* (pp. 3997-4000). IEEE.
13. Abdusalomov, A.B., Mukhiddinov, M. and Whangbo, T.K., 2023. Brain tumor detection based on deep learning approaches and magnetic resonance imaging. *Cancers*, 15(16), p.4172.
14. Deepak, S. and Ameer, P.M. (2021) “Automated Categorization of Brain Tumor from MRI Using CNN features and SVM,” *Journal of ambient intelligence and humanized computing*, 12(8), pp. 8357–8369. Available at: <https://doi.org/10.1007/s12652-020-02568-w>.

10. Bibliography

1. Lotlikar VS, Satpute N, Gupta A. Brain Tumor Detection Using Machine Learning and Deep Learning: A Review. *Curr Med Imaging*. 2022;18(6):604-622. doi: 10.2174/1573405617666210923144739. PMID: 34561990.
2. Pandey, S., Goyal, A. and Vansuha, D. (2021) 'Brain Tumour Detection Using Cnn', *Turkish journal of computer and mathematics education*, 12(11), pp. 4597–4603.
3. www.iarc.who.int. (n.d.). *Cancer Topics – IARC*. [online] Available at: <https://www.iarc.who.int/cancer-topics/>.
4. Tulip Mazumdar (2023) Safer brain surgery using AI possible within two years Available at: <https://www.bbc.co.uk/news/health-66921926> Date Accessed: 2nd November 2023.
5. Cancer Research UK. (2015). Brain, other CNS and intracranial tumours statistics. [online] Available at: <https://www.cancerresearchuk.org/health-professional/cancer-statistics/statistics-by-cancer-type/brain-other-cns-and-intracranial-tumours#heading-Two>.
6. S. Damodharan and D. Raghavan, "Combining tissue segmentation and neural network for brain tumor detection," *International Arab Journal of Information Technology*, vol. 12, no. 1, pp. 42–52, 2015.
7. M. Alfonse and A.-B. M. Salem, "An automatic classification of brain tumors through MRI using support vector machine," *Egyptian Computer Science Journal*, vol. 40, pp. 11–21, 2016.
8. E. A. Zanaty, "Determination of gray matter (GM) and white matter (WM) volume in brain magnetic resonance images (MRI)," *International Journal of Computer Applications*, vol. 45, pp. 16–22, 2012.
9. admin (2021). Top Programming Languages 2021: By Type and Comparison. [online] ISHIR | Software Development India. Available at: <https://www.ishir.com/blog/36749/top-75-programming-languages-in-2021-comparison-and-by-type.htm>.
10. Sadad, T. *et al.* (2021) "Brain tumor detection and multi-classification using advanced deep learning techniques," *Microscopy research and technique*, 84(6), pp. 1296–1308. Available at: <https://doi.org/10.1002/jemt.23688>.

11. Singh, A., 2016, March. Review of brain tumor detection from MRI images. In *2016 3rd International conference on computing for sustainable global development (INDIACom)* (pp. 3997-4000). IEEE.
12. Abdusalomov, A.B., Mukhiddinov, M. and Whangbo, T.K., 2023. Brain tumor detection based on deep learning approaches and magnetic resonance imaging. *Cancers*, 15(16), p.4172.
13. Deepak, S. and Ameer, P.M. (2021) “Automated Categorization of Brain Tumor from MRI Using CNN features and SVM,” *Journal of ambient intelligence and humanized computing*, 12(8), pp. 8357–8369. Available at: <https://doi.org/10.1007/s12652-020-02568-w>.
14. Khairandish, M.O. *et al.* (2022) “A Hybrid CNN-SVM Threshold Segmentation Approach for Tumor Detection and Classification of MRI Brain Images,” *Ingénierie et recherche biomédicale*, 43(4), pp. 290–299. Available at: <https://doi.org/10.1016/j.irbm.2021.06.003>.
15. Ganesan, P. *et al.* (2020) “Brain Tumour Segmentation and Measurement Based on Threshold and Support Vector Machine Classifier,” *Research journal of pharmacy and technology*, 13(6), pp. 2573–2577. Available at: <https://doi.org/10.5958/0974-360X.2020.00458.8>.

Appendix I

(BRAIN TUMOUR DETECTION SVM ALGORITHM DEMO VIDEO):

<https://drive.google.com/file/d/1P7nos2wluUrO0f9NMZt3tsoV7dPaQejU/view?usp=sharing>

(BRAIN TUMOUR DETECTION CNN ALGORITHM DEMO VIDEO):

<https://drive.google.com/file/d/1KgNsZNN80vLMNQ6LtTSnp5951QODRzID/view?usp=sharing>