



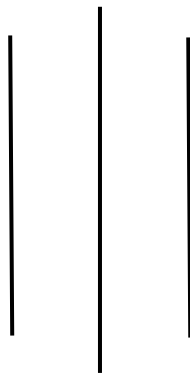
DURSIKSHYA
Transforming Education

CANCER DETECTION SYSTEM USING CNN

A PROJECT REPORT

Submitted to

SCOTTISH QUALIFICATION AUTHORITY



Submitted by

Sourav Rajbanshi

February 2025

ABSTRACT

This project presents an innovative cancer detection system that integrates machine learning, deep learning, and transfer learning techniques to enhance diagnostic accuracy. The system employs Convolutional Neural Networks (CNN) for robust image classification, leveraging their capability to automatically and efficiently extract relevant features from medical images. Advanced image preprocessing techniques, including normalization, augmentation, and noise reduction, are applied to improve image quality and ensure consistent input data. Transfer learning with pre-trained models such as VGG19 and ResNet50 accelerates the training process and enhances the system's performance by utilizing existing knowledge from large-scale datasets. The proposed system is designed to detect various types of cancer from medical images, providing early and accurate diagnosis, which is crucial for effective treatment planning. Experimental results demonstrate the system's high accuracy and reliability in identifying cancerous tissues, showcasing its potential as a valuable tool in medical diagnostics. This project aims to contribute to the field of medical imaging by providing a scalable and efficient solution for cancer detection, ultimately improving patient outcomes through timely and precise diagnosis. The integration of these advanced techniques ensures that the system remains at the forefront of medical technology, offering a promising approach to cancer detection and treatment.

Keywords: Autoencoders (AEs); cancer detection; convolutional neural networks (CNNs); deep learning; generative adversarial models (GANs); machine learning, Image Processing.

TABLE OF CONTENT

ABSTRACT.....	i
TABLE OF CONTENT	ii
TABLE OF FIGURES	iv
Chapter 1: Introduction.....	1
1.1 Introduction.....	1
1.2 Problem Statement	1
1.3 Objectives	2
1.4 Scope and limitation	2
Chapter 2: Background Study and Literature Review.....	3
2.1 Background Study.....	3
2.2 Literature Review.....	3
Chapter 3: System Analysis and Design.....	6
3.1 System Analysis	6
3.2 CNN (Convolution Neural Network) Model	7
3.2.1 Requirement Analysis	9
3.2.2 Feasibility Study	13
3.3 System Design	14
3.3.1 Architecture Design	14
3.3.2 Use Case Diagram.....	15
3.3.3 Flow Chart	16
3.3.4 Interface Design	17
Chapter 4: Implementation and Testing.....	18
4.1 Implementation	18
4.1.1 Tools Used	18
4.1.2 Implementation Detail of Modules	21

4.2	Testing.....	21
4.2.1	Balancing the Data to remove bias	22
4.2.2	Image Resizing.....	23
	23
4.2.3	Accuracy Graph	24
4.2.4	Loss Graph	25
4.2.5	Model Training	26
Chapter 5:	Conclusion and Future Recommendation.....	27
5.1	Lesson Learnt / Outcome	27
5.2	Conclusion	27
5.3	Future Recommendation.....	27
References.....		28
Appendix.....		29

TABLE OF FIGURES

Figure 3. 1 Agile Methodology	6
Figure 3. 2 Architecture of Convolution Neural Network.....	8
Figure 3. 3 Image Uploading.....	9
Figure 3. 4 Preprocessing of image	9
Figure 3. 5 Base model.....	10
Figure 3. 6 Result Display	10
Figure 3. 7 Balanced Dataset.....	11
Figure 3. 8 Accuracy score	12
Figure 3. 9 Gantt Chart of Fitness Mobile Application Error! Bookmark not defined.	
Figure 3. 10 Use Case Diagram of Cancer Detection System using CNN.....	15
Figure 3. 11 Flow chart of Cancer Detection System using CNN	16
Figure 3. 12 UI Page.....	17
Figure 3. 13 Implementation of Detail of module	21

LIST OF ABBREVIATIONS

BCA	Bachelor of Computer Application
CNN	Convolution Neural Network
CPU	Central Processing Unit
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
JS	JavaScript
ML	Machine Learning
TF	TensorFlow
TL	Transfer Learning

Chapter 1: Introduction

1.1 Introduction

Cancer is one of the most critical diseases that has caused several deaths in today's world. In most cases, doctors and practitioners are only able to diagnose cancer in its later stages. In the later stages, planning cancer treatment and increasing the patient's survival rate becomes a very challenging task. Therefore, it becomes the need of the hour to detect cancer in the early stages for appropriate treatment and surgery planning. Analysis and interpretation of medical images such as MRI and CT scans help doctors and practitioners diagnose many diseases, including cancer disease. However, manual interpretation of medical images is costly, time-consuming and biased. Nowadays, deep learning, a subset of artificial intelligence, is gaining increasing attention from practitioners in automatically analyzing and interpreting medical images without their intervention. Deep learning methods have reported extraordinary results in different fields due to their ability to automatically extract intrinsic features from images without any dependence on manually extracted features. This project provides a comprehensive review of deep learning methods in cancer detection and diagnosis, mainly focusing on cancer detection. This study describes various deep learning models and steps for applying deep learning models in detecting cancer. Recent developments in cancer detection based on deep learning methods have been critically analyzed and summarized to identify critical challenges in applying them for detecting cancer accurately in the early stages. Based on the identified challenges. The outcome of this project provides many clues for developing practical and accurate cancer detection systems for its early diagnosis and treatment planning.

1.2 Problem Statement

The problem statement of Cancer Detection System is given below:

- To early detection of cancer significantly increases the chances of successful treatment and survival.
- To reduce traditional, diagnose that are often time-consuming, expensive, and sometimes inaccurate.
- To develop an accurate and efficient early-stage cancer detection system is a critical challenge.

- To aim for system to enhance diagnostic accuracy, reducing false positives and negatives.
- To speed up diagnosis that allows for intervention and better patient outcomes.

1.3 Objectives

The objectives of a cancer detection system using machine learning include improving early diagnosis accuracy, reducing false positives and negatives, accelerating detection speed, enhancing image analysis capabilities, providing cost-effective screening, enabling personalized treatment plans, integrating with existing healthcare systems, facilitating large-scale data analysis, and supporting continuous learning and improvement.

The objective of Cancer Detection System are as follows:

- To developing a machine learning model to accurately detect early-stage cancer.
- To collect and preprocess diverse medical data for model training.
- To support oncologists and pathologists in making informed diagnostic decisions.
- To validate the model using independent datasets to ensure reliability.

1.4 Scope and limitation

The cancer detection system using machine learning aims to revolutionize the early diagnosis of cancer by leveraging advanced algorithms to analyze complex medical data. This system will incorporate diverse data sources, including medical imaging, histopathological slides, and genomic information, to provide a comprehensive and accurate diagnosis. The scope includes developing a robust model capable of detecting various types of cancer with high precision and speed, reducing the incidence of false positives and negatives. It will also feature a user-friendly interface designed to assist medical professionals, enhancing their diagnostic capabilities and supporting more informed decision-making. The system will be validated on independent datasets to ensure its reliability and generalizability. By integrating this technology into clinical practice, the project aims to improve patient outcomes through early detection and timely intervention, ultimately reducing cancer-related mortality and healthcare costs.

Chapter 2: Background Study and Literature Review

2.1 Background Study

Cancer detection has been a major area of research in medical sciences due to the high mortality rate associated with late diagnoses. Traditional diagnostic methods such as biopsy, MRI, and CT scans, though effective, are often time-consuming, expensive, and prone to human error. In recent years, advancements in artificial intelligence (AI) and machine learning, specifically deep learning, have revolutionized medical imaging and cancer detection. Convolutional Neural Networks (CNNs), a class of deep learning models, have shown immense promise in this domain due to their ability to automatically extract relevant features from medical images without the need for manual intervention.

CNNs are particularly effective in image recognition tasks, making them suitable for detecting cancerous lesions in various imaging modalities such as mammograms, histopathological images, and MRIs. By learning from large datasets of labeled images, CNNs can identify patterns and anomalies associated with cancer, enabling early detection and improving patient outcomes.

In the context of a cancer detection system, CNN models such as VGG, ResNet, and InceptionNet have been widely used to classify images into cancerous and non-cancerous categories. These models can significantly reduce the workload of radiologists and pathologists, offering more accurate and faster diagnoses. Additionally, CNN-based systems are continually improving with access to larger datasets and advanced architectures, further increasing their reliability in real-world clinical settings. This technological advancement holds great potential to make cancer screening more accessible, especially in regions with limited healthcare resources.

2.2 Literature Review

Cancer detection is a critical field in medical diagnostics, where early detection significantly improves patient survival rates. Traditional methods, while effective, are often limited by their manual nature, time consumption, and potential for human error. Advances in machine learning (ML), image processing, and deep learning (DL) have opened new avenues for enhancing the accuracy and efficiency of cancer detection systems.

Machine learning techniques, including support vector machines (SVM), decision trees, and ensemble methods, have been extensively studied for cancer detection. Studies have demonstrated that ML algorithms can classify cancerous and non-cancerous tissues based on features extracted from various medical data sources. For instance, SVMs have shown high accuracy in classifying breast cancer from mammographic images, and random forests have been used effectively in identifying lung cancer nodules in CT scans.

Image processing plays a pivotal role in enhancing the quality and interpretability of medical images. Techniques such as segmentation, feature extraction, and enhancement are critical for preparing data for subsequent ML and DL analyses. Research has shown that using techniques like histogram equalization, edge detection, and morphological operations can significantly improve the visibility of tumors in medical images. For example, segmentation algorithms like watershed and active contour models have been applied to accurately delineate tumor boundaries in MRI scans.

Deep learning, particularly convolutional neural networks (CNNs), has revolutionized cancer detection through its ability to automatically learn and extract features from raw medical images. CNNs have been successfully employed in various studies for detecting different types of cancer, including breast, lung, and skin cancers. Notable works include the use of deep CNNs for mammogram analysis, achieving accuracy comparable to expert radiologists, and the development of DL models for detecting melanoma from dermoscopic images with high sensitivity and specificity.

Recent literature highlights the integration of ML, image processing, and DL techniques to build comprehensive cancer detection systems. This approach leverages the strengths of each domain to enhance overall system performance. For instance, a study combined traditional image processing methods with DL to preprocess histopathological images before feeding them into a CNN, resulting in improved detection accuracy of colorectal cancer. Another research integrated ML algorithms with DL-extracted features to classify lung nodules, demonstrating enhanced performance compared to using either method alone.

Despite significant advancements, challenges remain in developing robust and generalizable cancer detection systems. Issues such as data variability, limited labeled datasets, and the need for explainable AI in clinical settings are areas of ongoing research. Future directions include the development of more sophisticated data augmentation

techniques, the creation of large, annotated datasets, and the integration of multi-modal data to improve diagnostic accuracy and robustness.

The integration of machine learning, image processing, and deep learning has shown great promise in enhancing cancer detection systems. By leveraging these technologies, researchers and clinicians can achieve more accurate, efficient, and early detection of cancer, ultimately improving patient outcomes. Continued research and development in this interdisciplinary field are essential to overcome existing challenges and realize the full potential of these advanced technologies in clinical practice.

Chapter 3: System Analysis and Design

3.1 System Analysis

The cancer detection system requires a robust architecture integrating data acquisition, preprocessing, feature extraction, and classification. It leverages advanced image processing techniques and deep learning models like CNNs for high accuracy and efficiency. Key components include secure data storage, a preprocessing module, feature extraction, and model training. A user-friendly interface supports medical professionals in making informed decisions. The system must ensure performance, scalability, reliability, security, and usability. Evaluation metrics such as accuracy, sensitivity, specificity, and computational efficiency are essential for validating the system's effectiveness in clinical practice.

Steps Essential for System Analysis:

1. Research and define essential components.
2. Analyze current processes and identify gaps in the model and implementation.
3. Write a requirements document.
4. Define standards, policies, and procedures for user data management and privacy.
5. Review the draft requirements document with users and other personnel.
6. Update and expand the project plan based on feedback.

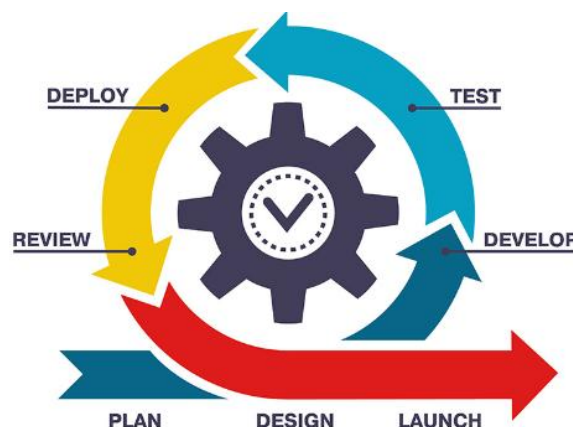


Figure 3. 1 Agile Methodology

3.2 CNN (Convolution Neural Network) Model

A Convolutional Neural Network (CNN) is a type of deep learning model, primarily used for analyzing visual data, such as images or videos. CNNs are particularly effective at recognizing patterns, shapes, and features in images, making them essential for tasks like image classification, object detection, and facial recognition. There are five types of layers in CNN as given below:

1. Convolutional Layer

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size $M \times M$. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter ($M \times M$).

2. Pooling Layer

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations. It basically summarises the features generated by a convolution layer.

3. Fully Connected Layer

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

4. Dropout

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on a new data

5. Activation Functions

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the

model should fire in the forward direction and which ones should not at the end of the network.

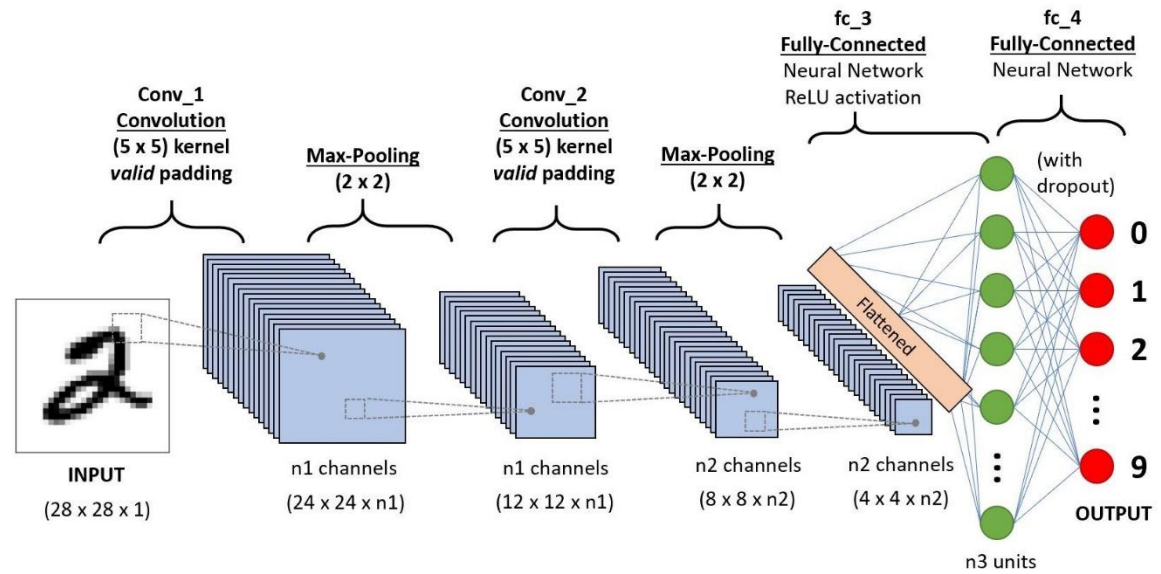


Figure 3. 2 Architecture of Convolution Neural Network

3.2.1 Requirement Analysis

Requirement analysis holds the process of reviewing and determining the system need, functional requirements, and non-functional requirements that a system must meet. For requirement analysis. Requirements are generally split into two types:

- i. Functional Requirements
- ii. Non-Functional Requirements

i. Functional Requirements

Image Uploading

The system provides an interface for users to upload medical images (e.g., mammograms, MRIs, CT scans, or histopathology images).

The system must support multiple image formats (e.g., PNG, JPEG, DICOM).

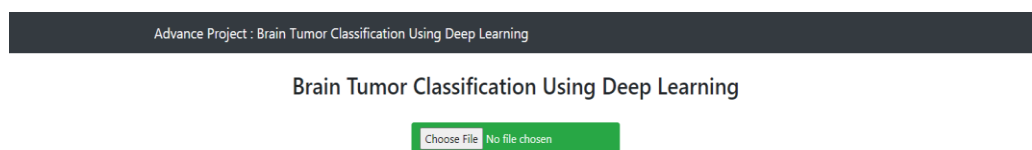


Figure 3. 3 Image Uploading

Preprocessing of Medical Images

The system preprocess images to ensure they are correctly formatted, resized, and normalized for input into the CNN model. This system handles grayscale and color images and adjust brightness or contrast if necessary.

```
[ ] def preprocessingImages1(path):  
    """  
    input : Path  
    output : ImageDataGenerator  
    """  
  
    image_data = ImageDataGenerator(zoom_range=0.2, shear_range=0.2, rescale=1/225, horizontal_flip=True)  
    image = image_data.flow_from_directory(directory = path, target_size=(224, 224), batch_size=32, class_mode='binary')  
  
    return image
```

Figure 3. 4 Preprocessing of image

Cancer Detection Using CNN Model

The system utilizes a CNN model (such as VGG19 or ResNet) to analyze uploaded images and classify them into cancerous and non-cancerous categories.

```
base_model = VGG19(input_shape=(240,240,3), include_top=False, weights='imagenet')

for layer in base_model.layers:
    layer.trainable=False

x=base_model.output
flat= Flatten()(x)

class_1 = Dense(4608, activation='relu')(flat)
drop_out = Dropout(0.2)(class_1)
class_2 = Dense(1152, activation='relu')(drop_out)
output = Dense(2, activation='softmax')(class_2)

model_01 = Model(base_model.input, output)
model_01.summary()
```

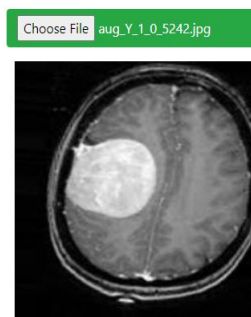
Figure 3. 5 Base model

Prediction Results Display

The system displays the classification result of the cancer detection, in a clear, user-friendly format. The system should offer additional visualization tools like heatmaps to highlight the areas of concern in the image.

Advance Project : Brain Tumor Classification Using Deep Learning

Brain Tumor Classification Using Deep Learning



Result: Yes Brain Tumor

Figure 3. 6 Result Display

ii. Non-Functional Requirements

Performance:

The system delivers high performance with minimal latency, providing quick predictions for cancer detection from medical images. Inference time should be optimized to ensure real-time or near real-time analysis, especially when processing large datasets.

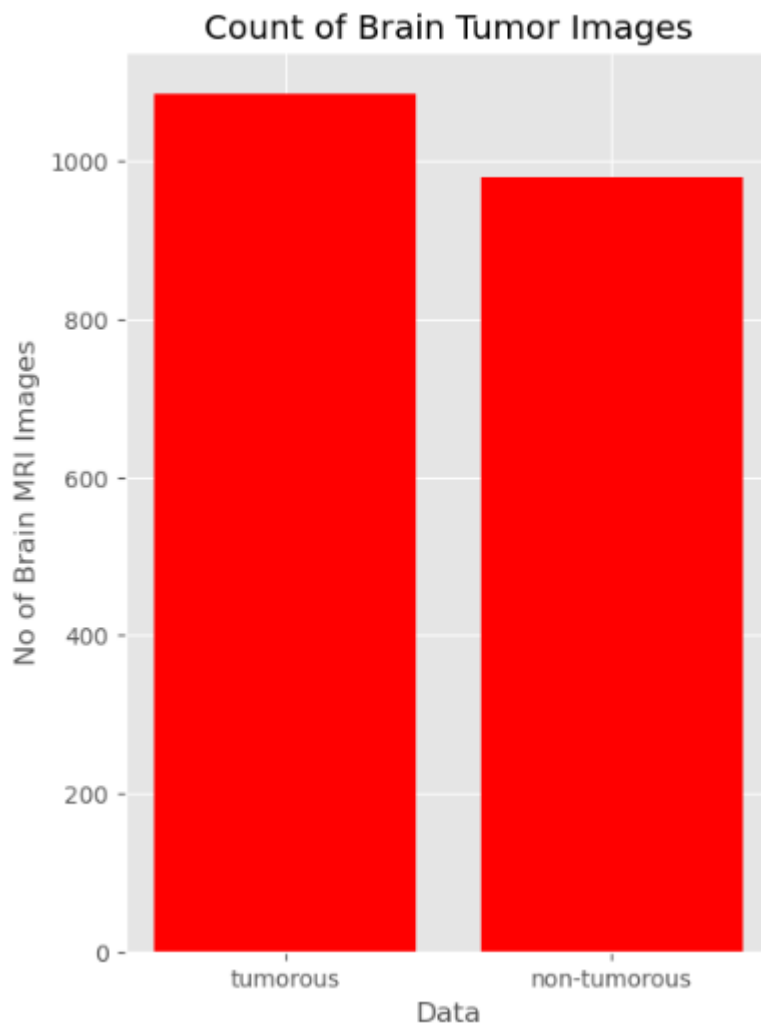


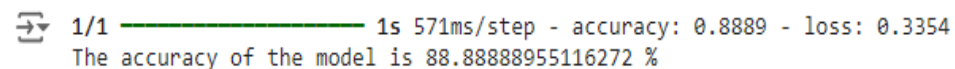
Figure 3. 7 Balanced Dataset

Accuracy:

The system must achieve a high level of accuracy (e.g., sensitivity and specificity above 90%) to ensure reliable cancer detection. False positives and false negatives should be minimized to avoid misdiagnosis.

```
[ ] acc = model.evaluate(test_data)[1]

print(f"The accuracy of the model is {acc*100} %")
```



1/1 ————— 1s 571ms/step - accuracy: 0.8889 - loss: 0.3354
The accuracy of the model is 88.8888955116272 %

Figure 3. 8 Accuracy score

Scalability:

The system should be scalable, capable of handling a large volume of medical image data. As the dataset grows, the system should efficiently accommodate the increase in input without degrading performance.

Security:

Patient data and medical images must be securely stored and transmitted. The system should implement encryption, secure authentication mechanisms, and compliance with data privacy regulations such as HIPAA or GDPR.

Usability:

The system should provide a user-friendly interface for medical professionals, making it easy to upload images, review results, and integrate with existing workflows in hospitals or clinics. Clear visualizations of predictions should be presented.

3.2.2 Feasibility Study

The feasibility study is the important step in any software development process. This is because it makes analysis of different aspects like cost required for developing and executing the system, the time required for each phase of the system and so on. If these important factors are not analyzed then definitely it would have impact on the organization and the development and the system would be a total failure. So, for running the project and the organization successfully this step is a very important step in a software development life cycle process.

i. Technical Feasibility

A technical feasibility study for a cancer detection system using machine learning, deep learning, and CNNs in Python evaluates the system's capability to accurately identify tumors from medical images. It assesses the integration of libraries like TensorFlow and Keras, data preprocessing, model training, and performance metrics, ensuring the system's reliability and efficiency in clinical settings.

ii. Operational Feasibility

An operational feasibility study for a brain tumor detection system using machine learning, deep learning, and CNNs in Python examines the system's integration into clinical workflows. It evaluates user training, data handling, and system maintenance, ensuring the solution is practical, user-friendly, and can be seamlessly adopted by healthcare professionals for accurate and efficient tumor detection.

iii. Schedule Feasibility

The development timeline for cancer detection system using cnn will be planned to accommodate key milestones, including initial design, development, testing, and deployment phases. The iterative development process allows for regular updates and feature additions, with each phase being scheduled to ensure timely delivery of the app. By setting realistic deadlines and allocating resources efficiently, the project can be completed within the planned timeframe while allowing for adjustments based on user feedback and emerging requirements.

3.3 System Design

The system design of Cancer Detection System using CNN that consists of architectural design shown as follows:

3.3.1 Architecture Design

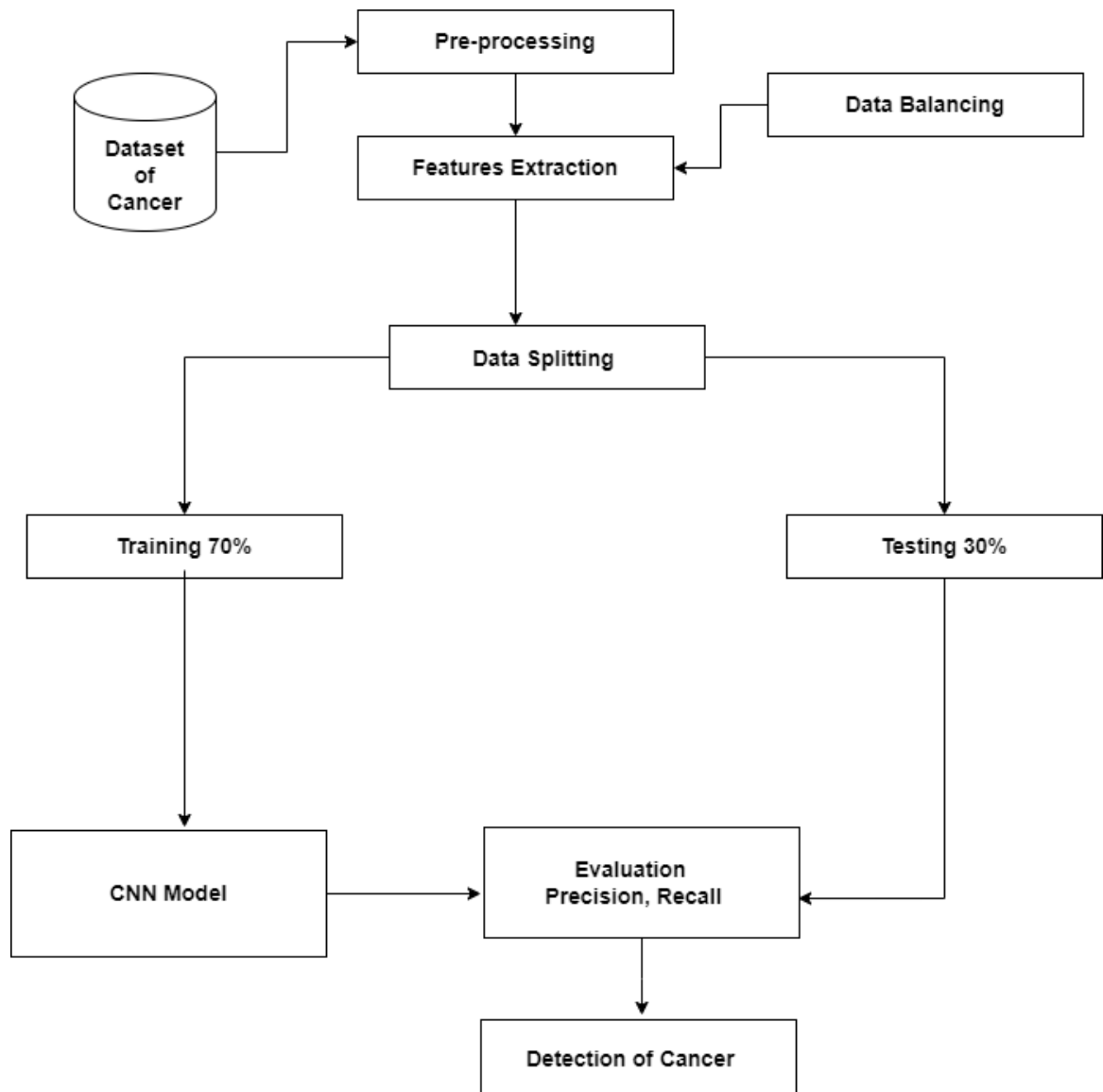


Figure 3.2- 1 Block Diagram of Cancer Detection using CNN

3.3.2 Use Case Diagram

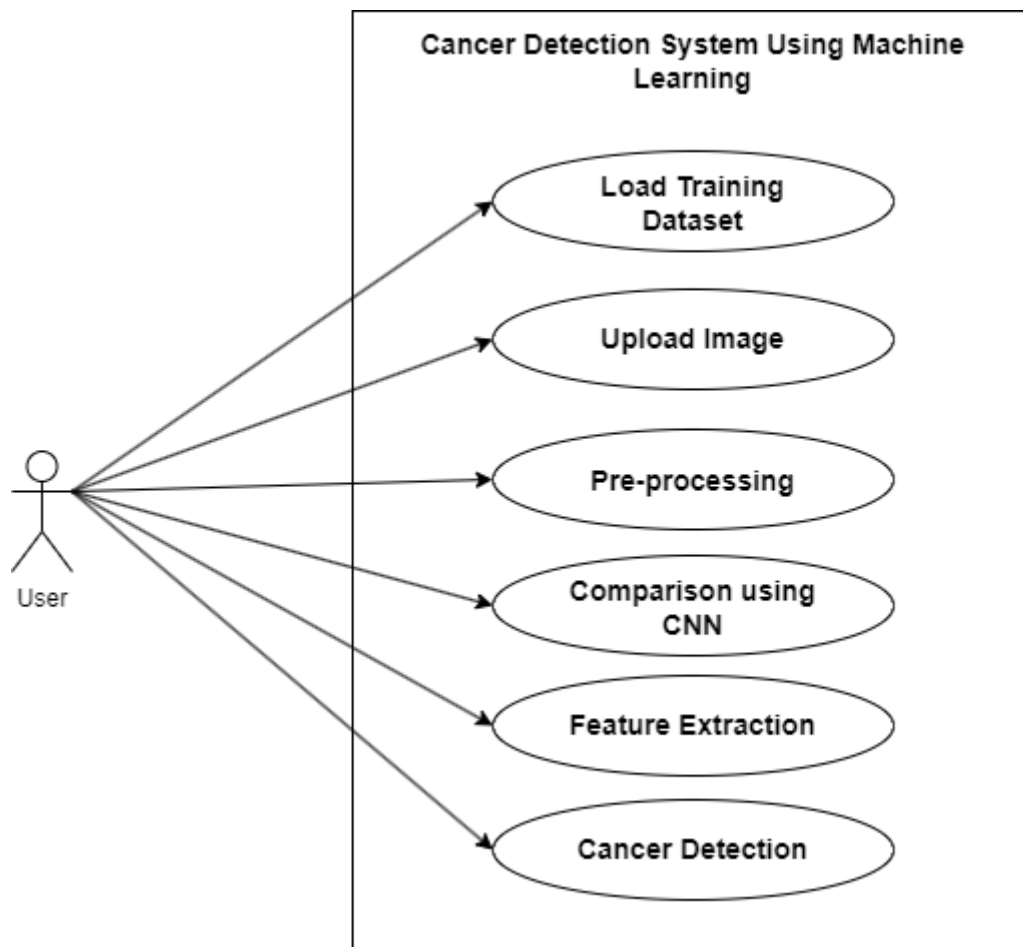


Figure 3. 9 Use Case Diagram of Cancer Detection System using CNN

The use case diagram of a cancer detection system using machine learning involves several key actions and interactions. The process begins with the "Load Training Dataset" step, where historical medical images are inputted to train the system. Users then "Upload Image" for analysis. The "Pre-processing" step involves preparing the image by normalizing and resizing it. This image is then subjected to "Comparison using CNN" (Convolutional Neural Networks) to identify patterns and anomalies. "Feature Extraction" is the next critical step, where specific characteristics indicative of cancer is extracted from the image. Finally, the system performs "Cancer Detection," providing a diagnostic output to the user.

3.3.3 Flow Chart

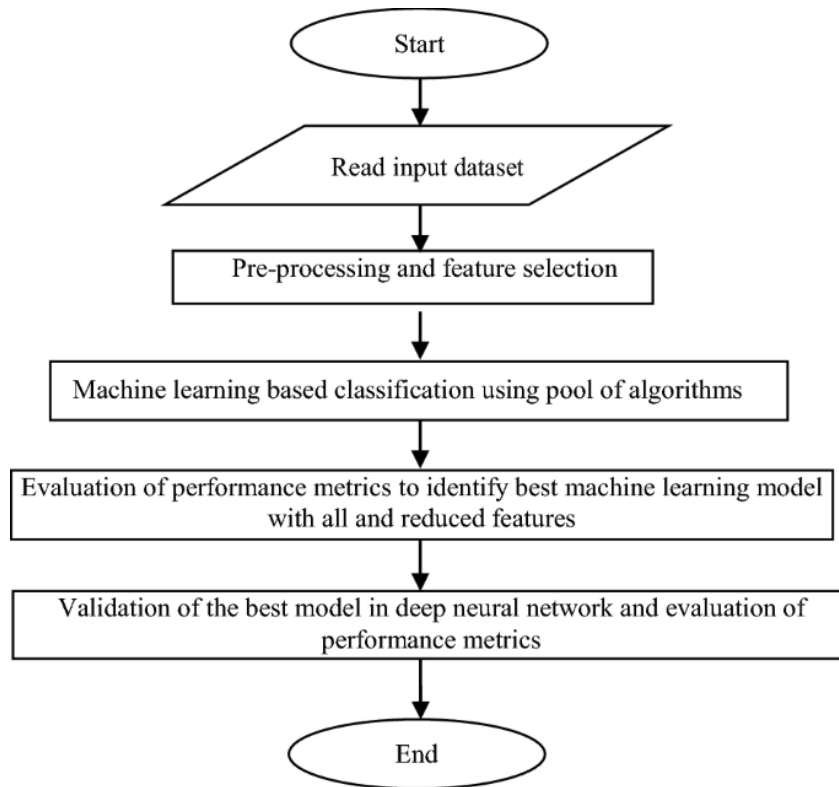


Figure 3. 10 Flow chart of Cancer Detection System using CNN

3.3.4 Interface Design

The interface design for Cancer Detection System using CNN is shown as follows:

1. Start Page UI

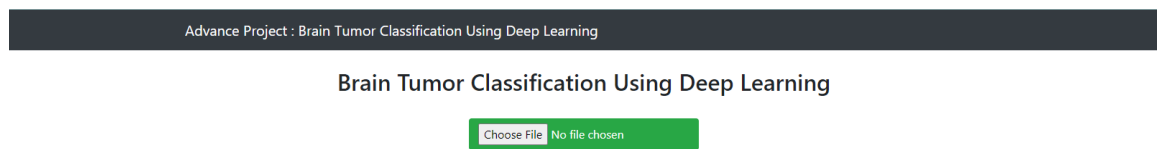


Figure 3. 11 UI Page

Chapter 4: Implementation and Testing

4.1 Implementation

For the development of the project, Agile model suits perfectly. The Agile model is a Iterative design process, used in software development processes. The Agile model is a flexible and iterative approach to software development. It emphasizes continuous planning, analysis, design, implementation, and testing. It adapts to changing requirements and customer feedback throughout the process, delivering small increments of the system in frequent iterations. This customer-centric approach ensures a high-quality and error-free product.

The planning phase is used for understanding why a system should be built and determining how the project team will go about building it. The analysis phase answers the questions of who will use the system, what the system will do, and where and when it will be used. During this analysis phase, the project team investigates any current system, identifies improvement opportunities, and develops a concept for the new system. The design phase decides how the system will operate in terms of the hardware, software, and network infrastructure that will be in place. In implementation phase, system is actually created and this phase usually gets the most attention, because for most systems it is the longest and most expensive single part of the development process. The final phase is the testing phase, were system after implementation it needs to be tested to make it error free.

4.1.1 Tools Used

The tools that are used for designing and development of the Fitness app are listed below:

Draw.io

It was used for designing the system designs such as system flowchart, ER diagram, architectural design, use case diagram and DFDs.

Programming Languages:

Python: The primary language for building machine learning models, particularly for implementing CNNs and performing data processing tasks.

Frameworks & Libraries:

TensorFlow/Keras: Deep learning frameworks used to build and train the CNN model. Keras provides an easy-to-use API for defining neural network architectures, while TensorFlow handles the computational graph and optimization processes.

NumPy: Used for numerical computations, including handling image arrays and matrix operations.

Pandas: Useful for managing datasets, handling data manipulation, and preprocessing.

OpenCV: A library for image processing, used to preprocess medical images before feeding them into the model.

Matplotlib/Seaborn: Visualization libraries used to generate plots for data exploration and model performance (e.g., accuracy, loss curves, confusion matrices).

Development Environment:

Jupyter Notebooks: An interactive environment for experimenting with model design, data preprocessing, and visualization.

PyCharm/VSCode: IDEs for code development, debugging, and managing larger codebases outside Jupyter Notebooks.

Data Management:

Kaggle: Often used to source publicly available datasets like medical image datasets (e.g., histopathological images, MRI scans) and for model benchmarking.

Google Colab: A cloud-based platform for running machine learning experiments without requiring local GPU resources.

Hardware/Cloud Platforms:

GPUs (NVIDIA): Essential for accelerating the training of CNNs, which require large-scale matrix operations that are computationally expensive.

Version Control:

Git/GitHub: Version control tools for collaborative development, tracking code changes, and ensuring the project is well-documented.

Deployment Tools:

Flask/Django: Python-based web frameworks for deploying the trained CNN model into a web-based interface for real-time image classification.

4.1.2 Implementation Detail of Modules

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 240, 240, 3)]	0
block1_conv1 (Conv2D)	(None, 240, 240, 64)	1792
block1_conv2 (Conv2D)	(None, 240, 240, 64)	36928
block1_pool (MaxPooling2D)	(None, 120, 120, 64)	0
block2_conv1 (Conv2D)	(None, 120, 120, 128)	73856
block2_conv2 (Conv2D)	(None, 120, 120, 128)	147584
block2_pool (MaxPooling2D)	(None, 60, 60, 128)	0
block3_conv1 (Conv2D)	(None, 60, 60, 256)	295168
block3_conv2 (Conv2D)	(None, 60, 60, 256)	590080
block3_conv3 (Conv2D)	(None, 60, 60, 256)	590080
block3_conv4 (Conv2D)	(None, 60, 60, 256)	590080
...		
Total params: 140,946,370		
Trainable params: 120,921,986		
Non-trainable params: 20,024,384		

Figure 3. 12 Implementation of Detail of module

4.2 Testing

The testing for Cancer Detection System using CNN is done in following ways:

4.2.1 Balancing the Data to remove bias

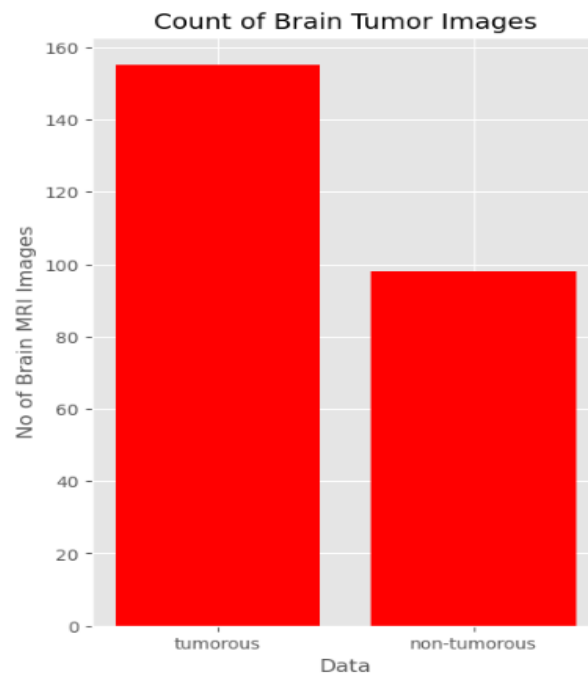


Figure 4 1 Unbalanced Dataset

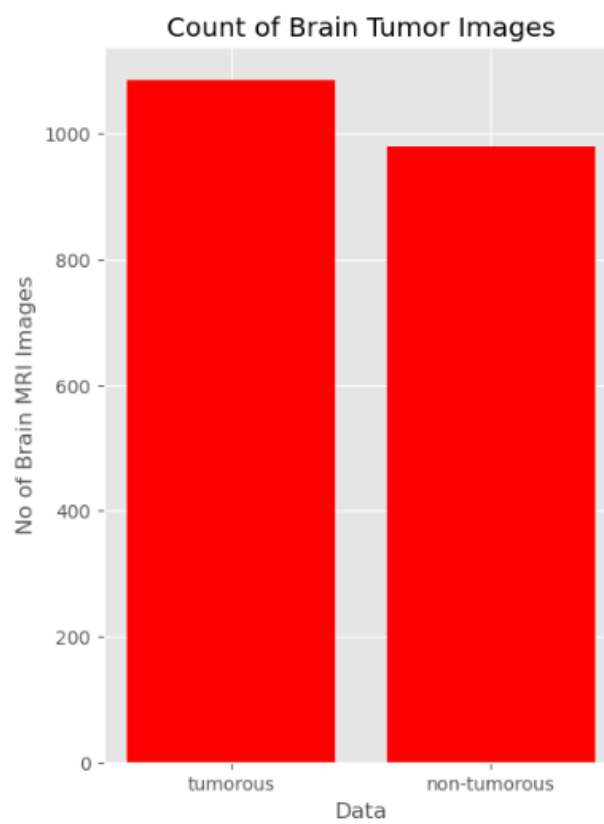


Figure 4 2 Balanced Dataset

4.2.2 Image Resizing

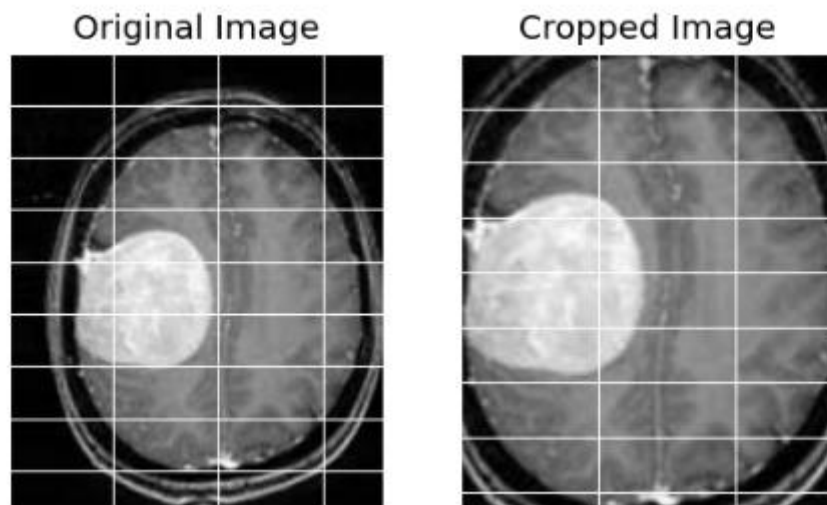


Figure 4 3 Image resizing of unhealthy image

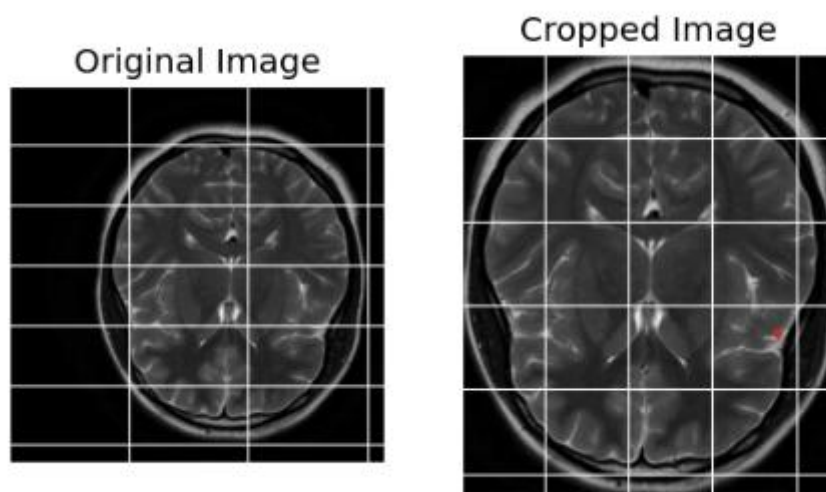


Figure 4 4 Image resizing of healthy image

4.2.3 Accuracy Graph

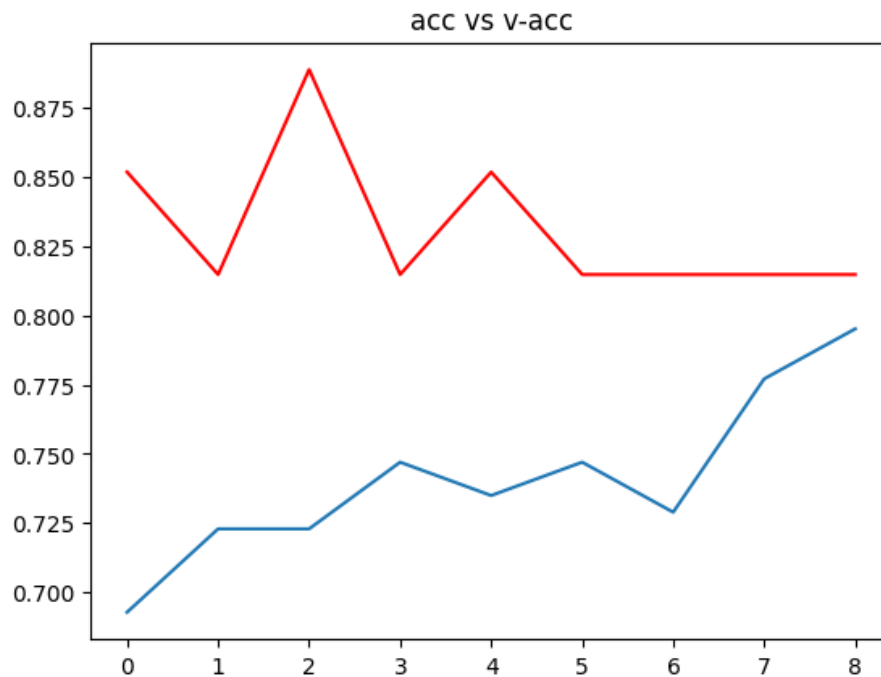


Figure 4 5 Accuracy Graph of CNN model

This graph compares training accuracy (blue line) and validation accuracy (red line) over 9 epochs in a machine learning model.

Training Accuracy: Starts at approximately 0.70 and steadily improves, reaching around 0.80 by the 8th epoch. This consistent increase indicates that the model is learning effectively from the training data.

Validation Accuracy: Begins at around 0.85 but fluctuates significantly across epochs, peaking at around 0.88. From the 5th epoch onward, the validation accuracy stabilizes at around 0.80.

4.2.4 Loss Graph



Figure 4 6 Accuracy Graph of CNN model

This graph shows the comparison of training loss (blue line) and validation loss (red line) over 9 epochs in a machine learning model.

Training Loss: Begins at approximately 0.60 and decreases steadily as the number of epochs increases. By the 8th epoch, it drops to around 0.45, indicating that the model is progressively minimizing the error on the training data.

Validation Loss: Starts at around 0.45 and fluctuates across epochs. It decreases sharply in the first few epochs but then fluctuates, showing several peaks and troughs. By the 7th epoch, the validation loss stabilizes around 0.35.

4.2.5 Model Training

```
Epoch 1/30
6/8 ----- 0s 68ms/step - accuracy: 0.6850 - loss: 0.6371
Epoch 1: val_accuracy improved from -inf to 0.85185, saving model to ./bestmodel.keras
8/8 ----- 3s 139ms/step - accuracy: 0.6870 - loss: 0.6321 - val_accuracy: 0.8519 - val_loss: 0.4831
Epoch 2/30
5/8 ----- 0s 82ms/step - accuracy: 0.7365 - loss: 0.5516
Epoch 2: val_accuracy did not improve from 0.85185
8/8 ----- 4s 61ms/step - accuracy: 0.7314 - loss: 0.5524 - val_accuracy: 0.8148 - val_loss: 0.3669
Epoch 3/30
6/8 ----- 0s 73ms/step - accuracy: 0.6719 - loss: 0.6570
Epoch 3: val_accuracy improved from 0.85185 to 0.88889, saving model to ./bestmodel.keras
8/8 ----- 3s 118ms/step - accuracy: 0.6846 - loss: 0.6410 - val_accuracy: 0.8889 - val_loss: 0.4019
Epoch 4/30
6/8 ----- 0s 87ms/step - accuracy: 0.7396 - loss: 0.5779
Epoch 4: val_accuracy did not improve from 0.88889
8/8 ----- 5s 86ms/step - accuracy: 0.7415 - loss: 0.5704 - val_accuracy: 0.8148 - val_loss: 0.4489
Epoch 5/30
6/8 ----- 0s 72ms/step - accuracy: 0.7668 - loss: 0.5057
Epoch 5: val_accuracy did not improve from 0.88889
8/8 ----- 5s 63ms/step - accuracy: 0.7588 - loss: 0.5133 - val_accuracy: 0.8519 - val_loss: 0.3582
Epoch 6/30
6/8 ----- 0s 74ms/step - accuracy: 0.6936 - loss: 0.5766
Epoch 6: val_accuracy did not improve from 0.88889
8/8 ----- 2s 68ms/step - accuracy: 0.7069 - loss: 0.5647 - val_accuracy: 0.8148 - val_loss: 0.4601
Epoch 7/30
6/8 ----- 0s 71ms/step - accuracy: 0.7044 - loss: 0.5401
Epoch 7: val_accuracy did not improve from 0.88889
8/8 ----- 3s 68ms/step - accuracy: 0.7105 - loss: 0.5379 - val_accuracy: 0.8148 - val_loss: 0.4208
Epoch 8/30
6/8 ----- 0s 68ms/step - accuracy: 0.7748 - loss: 0.4488
Epoch 8: val_accuracy did not improve from 0.88889
8/8 ----- 4s 61ms/step - accuracy: 0.7754 - loss: 0.4564 - val_accuracy: 0.8148 - val_loss: 0.3524
Epoch 9/30
```

Figure 4 7 Model Training

Chapter 5: Conclusion and Future Recommendation

5.1 Lesson Learnt / Outcome

The cancer detection project using machine learning, deep learning, and CNNs in Python revealed the critical role of high-quality data and robust preprocessing techniques. It emphasized the importance of model tuning and validation for achieving high accuracy. The project showcased the potential for early diagnosis, leading to better patient outcomes and cost savings. Collaboration between data scientists and medical professionals was key, highlighting the need for interdisciplinary approaches in developing effective healthcare solutions. Continuous learning and adaptation to new technologies were essential.

5.2 Conclusion

The cancer detection project using machine learning, deep learning, and CNNs in Python demonstrated the transformative potential of AI in healthcare. By leveraging advanced algorithms and interdisciplinary collaboration, the project achieved significant improvements in diagnostic accuracy and efficiency. The lessons learned underscore the importance of high-quality data, continuous model optimization, and the integration of AI solutions into clinical practice. This project paves the way for future innovations, promising better patient outcomes and cost-effective healthcare solutions.

5.3 Future Recommendation

Future recommendations for the cancer detection system using machine learning, deep learning, and CNNs in Python include enhancing data collection with diverse, high-quality medical images to improve model robustness. Exploring advanced architectures like transfer learning and ensemble methods can boost accuracy. Developing user-friendly interfaces will facilitate clinical integration. Ensuring regulatory compliance is crucial for safe use. Ongoing interdisciplinary collaboration between data scientists, medical experts, and regulatory bodies will address emerging challenges and drive innovation, ensuring the system remains effective and up-to-date.

References

- [1] J. S. S. M. S. A. Utpol Kanti Das, "Intelligent Cancer Detection System," *Intelligent Cancer Detection System*, no. 25-27 June 2021, 2021.
- [2] F. S. S. A. A. A. N. F. M. S. Tonmoy Hossain, "Brain Tumor Detection," *Brain Tumor Detection Using Convolutional Neural*, no. 2019-12-22, p. 6, 2019.
- [3] M. B. G. S. M. D. K. A. P. Pushkar Sathe, "Cancer Detection using Machine Learning," *Cancer Detection using Machine Learning*, no. 09 | Sep 2020, p. 5, 2020.
- [4] A. R. P. S. M. R. D. G. S. Avigyan Sinha, "Brain Tumour Detection Using Deep Learning," no. 2021, p. 5, 2021.
- [5] "Deep Learning-Based Cancer Detection-Recent Developments, Trend," *Deep Learning-Based Cancer Detection-Recent Developments, Trend*, no. 26 September 2021, p. 37, 2021.

Appendix

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os, shutil
import cv2
import matplotlib.image as mpimg
import seaborn as sns
%matplotlib inline
plt.style.use('ggplot')
```

```
# Dataset
# import zipfile

# z = zipfile.ZipFile('archive.zip')
# z.extractall()
```

```
folder = 'brain_tumor_dataset/yes/'
count = 1

for filename in os.listdir(folder):
    source = folder + filename
    destination = folder + "Y_" +str(count)+".jpg"
    os.rename(source, destination)
    count+=1
print("All files are rename in the yes dir")
```

```
folder = 'brain_tumor_dataset/no/'
count = 1

for filename in os.listdir(folder):
    source = folder + filename
    destination = folder + "N_" +str(count)+".jpg"
    os.rename(source, destination)
    count+=1
```

Figure 6 1 Image Unzipping and loading

EDA (Exploratory data Analysis)

```
listyes = os.listdir("brain_tumor_dataset/yes/")
number_files_yes = len(listyes)
print(number_files_yes)

listno = os.listdir("brain_tumor_dataset/no/")
number_files_no = len(listno)
print(number_files_no)
```

Plot

```
data = {'tumorous': number_files_yes, 'non-tumorous': number_files_no}
typex = data.keys()
values = data.values()

fig = plt.figure(figsize=(5,7))
plt.bar(typex, values, color="red")
plt.xlabel("Data")
plt.ylabel("No of Brain MRI Images")
plt.title("Count of Brain Tumor Images")
```

Figure 6 2 Performing EDA

Data Preprocessing

```
# Convert BGR to Gray
# GaussianBlur
# Threshold
# Erode
# Dilate
# Finding Contours
```

```
import cv2
import imutils
import matplotlib.pyplot as plt

def crop_brain_tumor(image, plot=False):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (5, 5), 0)

    thres = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
    thres = cv2.erode(thres, None, iterations=2)
    thres = cv2.dilate(thres, None, iterations=2)

    cnts = cv2.findContours(thres.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    c = max(cnts, key=cv2.contourArea)

    extLeft = tuple(c[c[:, :, 0].argmin()][0])
    extRight = tuple(c[c[:, :, 0].argmax()][0])
    extTop = tuple(c[c[:, :, 1].argmin()][1])
    extBot = tuple(c[c[:, :, 1].argmax()][1])

    new_image = image[extTop[1]:extBot[1], extLeft[0]:extRight[0]]

    if plot:
        plt.figure()
        plt.subplot(1, 2, 1)
        plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
        plt.tick_params(axis='both', which='both', top=False, bottom=False, left=False, right=False, labelbottom=False, labeltop=False, labelleft=False, labelright=False)
```

Figure 6 3 Data Preprocessing

Data Splitting

```
# Train
# Test
# Validation
```

```
if not os.path.isdir('tumorous_and_nontumorous'):
    base_dir = 'tumorous_and_nontumorous'
    os.mkdir(base_dir)
```

```
if not os.path.isdir('tumorous_and_nontumorous/train'):
    train_dir = os.path.join(base_dir, 'train')
    os.mkdir(train_dir)

if not os.path.isdir('tumorous_and_nontumorous/test'):
    test_dir = os.path.join(base_dir, 'test')
    os.mkdir(test_dir)

if not os.path.isdir('tumorous_and_nontumorous/valid'):
    valid_dir = os.path.join(base_dir, 'valid')
    os.mkdir(valid_dir)
```

Figure 6 4 Performing Data Splitting

Model Building

```
train_datagen = ImageDataGenerator(rescale = 1.0/255,
    horizontal_flip=0.4,
    vertical_flip=0.4,
    rotation_range=40,
    shear_range=0.2,
    width_shift_range=0.4,
    height_shift_range=0.4,
    fill_mode='nearest')

test_data_gen = ImageDataGenerator(rescale=1.0/255)
valid_data_gen = ImageDataGenerator(rescale=1.0/255)
```

```
train_generator = train_datagen.flow_from_directory('tumorous_and_nontumorous/train', batch_size=32, target_size=(240, 240), class_mode='categorical', shuffle=True, seed = 42, color_mode='rgb')
```

```
test_generator = test_data_gen.flow_from_directory('tumorous_and_nontumorous/test', batch_size=32, target_size=(240, 240), class_mode='categorical', shuffle=True, seed = 42, color_mode='rgb')
```

```
valid_generator = valid_data_gen.flow_from_directory('tumorous_and_nontumorous/valid', batch_size=32, target_size=(240, 240), class_mode='categorical', shuffle=True, seed = 42, color_mode='rgb')
```

[+ Code](#) [+ Markdown](#)

```
class_labels = train_generator.class_indices
class_name = {value: key for (key,value) in class_labels.items()}
```

```
class_name
```

Figure 6 5 Model Building

Incremental unfreezing and fine tuning

```
base_model = VGG19(include_top=False, input_shape=(240,240,3))
base_model_layer_names = [layer.name for layer in base_model.layers]
base_model_layer_names
```

```
base_model = VGG19(include_top=False, input_shape=(240,240,3))
base_model_layer_names = [layer.name for layer in base_model.layers]
base_model_layer_names

x=base_model.output
flat = Flatten()(x)

class_1 = Dense(4608, activation = 'relu')(flat)
drop_out = Dropout(0.2)(class_1)
class_2 = Dense(1152, activation = 'relu')(drop_out)
output = Dense(2, activation='softmax')(class_2)

model_02 = Model(base_model.inputs, output)
model_02.load_weights('model_weights/vgg19_model_01.weights.h5')

set_trainable = False
for layer in base_model.layers:
    if layer.name in ['block5_conv4', 'block5_conv3']:
        set_trainable=True
    if set_trainable:
        layer.trainable=True
    else:
        layer.trainable=False

print(model_02.summary())
```

Figure 6 6 Incremental unfreezing and fine tuning