A MINOR PROJECT REPORT

ON

BRAIN STROKE DETECTION USING DEEP LEARNING: A COMPREHENSIVE STUDY

Submitted in partial fulfillment of the requirement

for the award of degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

by

Tappa Rohith Kumar	*21P61A05N3*
Samba Vipul Raj	*21P61A05K9*
Suhas Reddy Gandhi	*21P61A05M4*

Under the esteemed guidance of

Mrs. K. Swetha

Assistant Professor, Dept of CSE

Department of Computer Science and Engineering

Counselling Code : VBIT

VIGNANA BHARATHI Institute of Technology ®

(A UGC Autonomous Institution, Approved by AICTE, Accredited by NBA & NAAC-A Grade, Affiliated to JNTUH)

Aushapur Village, Ghatkesar mandal, Medchal Malkajgiri (District) Telangana-501301

May – 2024

DECLARATION

We, **Tappa Rohith Kumar, Samba Vipul Raj, Suhas Reddy Gandhi,** bearing hall ticket numbers **21P61A05N3, 21P61A05K9, 21P61A05M4** here by declare that the major project report entitled "**Brain Stroke Detection using Deep Learning**" under the guidance of **Mrs. K. Swetha**, Assistant Professor, Department of Computer Science and Engineering, **Vignana Bharathi Institute of Technology**, **Hyderabad,** have submitted to Jawaharlal Nehru Technological University Hyderabad, Kukatpally, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

This is a record of bonafide work carried out by us and the design embodied for this project have not been reproduced or copied from any source. The design embodied for this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

Tappa Rohith Kumar	21P61A05N3
Samba Vipul Raj	21P61A05K9
Suhas Reddy Gandhi	21P61A05M4

Counselling Code : VBIT

VIGNANA BHARATHI Institute of Technology

(A UGC Autonomous Institution, Approved by AICTE, Accredited by NBA & NAAC-A Grade, Affiliated to JNTUH)

Aushapur (V), Ghatkesar (M), Hyderabad, Medchal –Dist, Telangana – 501 301.

DEPARTMENT

OF

COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the major project titled **"Brain Stroke Detection using Deep Learning: A Comprehensive Study"** Submitted by **Tappa Rohith Kumar(21P61A05N3), Samba Vipul Raj(21P61A05K9), Suhas Reddy Gandhi(21P61A05M4)** in B.Tech IV-I semester Computer Science & Engineering is a record of the bonafide work carried out by them.

The Design embodied in this report have not been submitted to any other University for the award of any degree.

INTERNAL GUIDE	**HEAD OF DEPARTMENT**
Mrs. K. Swetha	Dr. Raju Dara
[Assistant Professor, Dept of CSE]	[Professor & HoD, Dept of CSE]

[External Examiner]

ACKNOWLEDGEMENTS

We are extremely thankful to our beloved Chairman, **Dr. N. Goutham Rao** and Secretar**y, Dr. G. Manohar Reddy** who took keen interest to provide us the infrastructural facilities for carrying out the project work.

We whole-heartedly thank **Dr. P. V. S. Srinivas** Professor & Principal, and **Dr. Raju Dara**, Professor & Head of the Department, Computer Science and Engineering for their encouragement and support and guidance in carrying out the major project phase II.

We would like to express our indebtedness to the Overall Project Coordinator, **Dr. Praveen Talari** Associate Professor, and Section coordinators, **Mrs. Kumari Jelli, Dr. A. L. Sreenivasulu**, Department of CSE for their valuable guidance during the course of project work.

We thank our Project Guide, **Mrs. Swetha,** Assistant Professor, for providing us with an excellent project and guiding us in completing our major project phase II successfully.

We would like to express our sincere thanks to all the staff of Computer Science and Engineering, VBIT, for their kind cooperation and timely help during the course of our project.

Finally, we would like to thank our parents and friends who have always stood by us whenever we were in need of them.

ABSTRACT

Brain strokes, or cerebrovascular accidents (CVAs), occur when the brain's blood supply is interrupted or reduced, causing brain tissue to lose oxygen and nutrients. This can lead to cell death, resulting in lasting brain damage, long-term disability, or even death. There are two main types of strokes: ischemic strokes, caused by blockages in blood vessels, and hemorrhagic strokes, caused by bleeding in or around the brain. Key risk factors include hypertension, smoking, diabetes, high cholesterol, and a sedentary lifestyle.

The project aims to develop a robust and accurate image classification model to aid in the early detection and diagnosis of brain strokes using medical imaging data. Leveraging Convolutional Neural Networks (CNNs) and advanced data augmentation techniques, the model is trained to recognize patterns and anomalies indicative of strokes. CNNs are particularly effective in analyzing visual data, as they automatically and adaptively learn spatial hierarchies of features. This project employs three distinct codes to study the model's development comprehensively: the first focuses on data preprocessing and augmentation, the second uses a pre-trained VGG16 model with additional layers and regularization techniques, and the third integrates advanced visualization techniques and evaluation metrics.

Various callbacks, such as TensorBoard for real-time tracking, ModelCheckpoint for saving the best-performing model, EarlyStopping to prevent overfitting, and ReduceLROnPlateau to adjust learning rates, are implemented to ensure the model's robustness and generalization. The expected outcome is a highly accurate and reliable tool to assist medical professionals in promptly diagnosing brain strokes, thus improving patient outcomes through timely intervention. Early detection enabled by such systems can significantly reduce the long-term consequences of strokes by allowing for faster treatment and preventive measures. This project highlights the transformative potential of artificial intelligence in healthcare, providing advanced diagnostic capabilities and supporting clinical decision-making.

***Keywords*: Transfer Learning, Data Augmentation, Machine Learning in Medicine, VGG16 Model**

VISION

To become, a Center for Excellence in Computer Science and Engineering with a focused Research, Innovation through Skill Development and Social Responsibility.

MISSION

DM-1: Provide a rigorous theoretical and practical framework across ***State-of-the-art*** infrastructure with an emphasis on ***software development***.

DM-2: Impact the skills necessary to amplify the pedagogy to grow technically and to meet ***interdisciplinary needs*** with collaborations.

DM-3: Inculcate the habit of attaining the professional knowledge, firm ethical values, ***innovative research*** abilities and societal needs.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO-01: Domain Knowledge: Synthesize mathematics, science, engineering fundamentals, pragmatic programming concepts to formulate and solve engineering problems using prevalent and prominent software.

PEO-02: Professional Employment: Succeed at entry- level engineering positions in the software industries and government agencies.

PEO-03: Higher Degree: Succeed in the pursuit of higher degree in engineering or other by applying mathematics, science, and engineering fundamentals.

PEO-04: Engineering Citizenship: Communicate and work effectively on team-based engineering projects and practice the ethics of the profession, consistent with a sense of social responsibility.

PEO-05: Lifelong Learning: Recognize the significance of independent learning to become experts in chosen fields and broaden professional knowledge.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO-01: Ability to explore emerging technologies in the field of computer science and engineering.

PSO-02: Ability to apply different algorithms indifferent domains to create innovative products.

PSO-03: Ability to gain knowledge to work on various platforms to develop useful and secured applications to the society.

PSO-04: Ability to apply the intelligence of system architecture and organization in designing the new era of computing environment.

PROGRAM OUTCOMES (POs)

Engineering graduates will be able to:

PO-01: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO-02: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO-03: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and cultural, societal, and environmental considerations.

PO-04: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO-05: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO-06: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities

relevant to the professional engineering practice.

PO-07: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO-08: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO-09: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO-10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO-11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO-12: Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Project Mapping Table:

Topic	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
Brain Stroke Prediction using Deep Learning: A Comprehensive Study	✓	✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

TABLE OF CONTENTS

CHAPTER – 1
INTRODUCTION

1. INTRODUCTION

Brain strokes, or cerebrovascular accidents (CVAs), are severe medical emergencies resulting from interrupted or reduced blood supply to the brain, leading to cell death, brain damage, disability, or death. They are categorized into ischemic strokes (caused by blockages) and haemorrhagic strokes (caused by bleeding). Risk factors include hypertension, smoking, diabetes, high cholesterol, and a sedentary lifestyle. Early detection and prompt medical intervention are crucial for improving patient outcomes and reducing long-term consequences.

This comprehensive study aims to develop a precise image classification model to aid in the early detection of brain strokes using medical imaging data. By leveraging Convolutional Neural Networks (CNNs) and advanced data augmentation techniques, the model is trained to recognize stroke-related patterns and anomalies. The project utilizes three codes: the first focuses on data preprocessing and augmentation, the second employs a pre-trained VGG16 model with additional layers and regularization techniques, and the third integrates advanced visualization techniques and evaluation metrics.

The primary objective is to develop a highly accurate and reliable model to assist medical professionals in diagnosing brain strokes early. By identifying stroke patterns in medical imaging data promptly, the system aims to improve patient outcomes through timely medical intervention, highlighting the potential of artificial intelligence to transform healthcare and support clinical decision-making..

1.1 MOTIVATION

The motivation behind this project stems from the pressing need to improve early detection and diagnosis of brain strokes, a leading cause of long-term disability and mortality worldwide. By harnessing the power of advanced artificial intelligence and deep learning techniques, we aim to create a reliable and precise tool that can aid medical professionals in identifying strokes quickly and accurately. This project aspires to make a significant impact on patient outcomes by enabling timely medical intervention, ultimately reducing the devastating effects of strokes and transforming healthcare through innovative technology. With early detection, we can save lives, enhance recovery, and provide a better quality of life for stroke survivors.

1.2 OVERVIEW OF EXISTING SYSTEM

Existing systems for brain stroke detection have made significant strides in leveraging advanced technologies to improve early diagnosis and treatment. One notable approach involves the use of machine learning techniques, such as support vector machines (SVM), decision trees, and deep learning models, to analyse medical imaging data. These models are trained on extensive datasets of labelled brain scans, allowing them to accurately identify and categorize stroke cases.

Another promising area is the development of portable stroke detection devices that can be used in prehospital settings. These devices utilize various diagnostic technologies, including near-infrared spectroscopy, ultrasound, electroencephalography, and microwave technology, to quickly determine stroke type and severity.

Additionally, artificial intelligence (AI) and deep learning have been applied to cerebral stroke image classification and segmentation. These methods automate the analysis of medical images, reducing human error and increasing accuracy. State-of-the-art studies have explored various computer-aided diagnosis techniques, comparing their performance and identifying challenges and future directions for research.

Overall, these existing systems demonstrate the potential of advanced technologies to enhance stroke detection and diagnosis, ultimately leading to better patient care and outcomes

1.3 OVERVIEW OF PROPOSED SYSTEM

Our proposed system aims to develop an accurate and reliable image classification model to assist in the early detection and diagnosis of brain strokes using medical imaging data. We leverage Convolutional Neural Networks (CNNs) to analyze and recognize patterns indicative of strokes. CNNs are particularly effective for image analysis as they learn spatial hierarchies of features. To enhance the model's generalization capabilities, we preprocess the data and apply augmentation techniques, introducing variability through random flips, rotations, and zooms.

We use a pre-trained VGG16 model for transfer learning, which helps improve accuracy by building on the established feature extraction capabilities of VGG16. This model's layers are frozen, and additional fully connected layers are added to fine-tune it specifically for stroke detection. Alongside this, we design and test two custom CNN architectures, each with different configurations of convolutional, pooling, dropout, and fully connected layers. The

first custom CNN includes a series of convolutional and max-pooling layers with dropout to prevent overfitting, while the second custom CNN explores batch normalization and an increased number of convolutional layers.

To ensure a fair comparison, we train and evaluate all three models—VGG16 and the two custom CNNs—using the same dataset. We consistently apply data augmentation techniques across all models and use the same evaluation metrics, such as accuracy, loss, and confusion matrices, to assess performance. This comprehensive evaluation helps us identify the most effective approach for early stroke detection.

Our primary objective is to create a highly accurate and reliable tool for early detection and diagnosis of brain strokes. By quickly identifying stroke patterns in medical imaging data, the system aims to support medical professionals in making timely decisions, thus improving patient outcomes through faster treatment and preventive measures. This project highlights the potential of artificial intelligence in healthcare, enhancing diagnostic capabilities and supporting clinical decision-making.

1.4 PROBLEM DEFINITION

Brain strokes cause significant disability and mortality worldwide. Prompt, accurate diagnosis is crucial for timely medical intervention, improving patient outcomes, and reducing severe complications. Traditional stroke diagnosis relies on manual medical imaging analysis, which is time-consuming and prone to error.

This project addresses the need for an efficient, accurate system to detect and diagnose brain strokes using medical imaging data. By leveraging Convolutional Neural Networks (CNNs), we aim to develop an automated image classification model to assist medical professionals in quickly identifying stroke-related anomalies.

Using transfer learning with a pre-trained VGG16 model, the project overcomes challenges related to limited datasets and computational resources. It evaluates various CNN architectures to determine the optimal approach for stroke detection, highlighting AI's potential to revolutionize healthcare by enhancing diagnostic capabilities and reducing the burden on medical professionals.

1.5 SYSTEM FEATURES

 Data Augmentation: The system employs data augmentation techniques, such as random flips, rotations, and zooms, to enhance the variability of the training dataset.

 Convolutional Neural Networks (CNNs): At the core of the system are CNNs, which are highly effective for image analysis. These networks automatically learn spatial hierarchies of features, making them ideal for detecting patterns in medical imaging data.

 Transfer Learning with VGG16: The system utilizes a pre-trained VGG16 model for transfer learning, leveraging its established feature extraction capabilities. This allows us to build on the knowledge of an already effective model, improving accuracy and reducing the need for extensive computational resources.

 Custom CNN Architectures: In addition to VGG16, the system includes two custom CNN architectures with different configurations of convolutional, pooling, dropout, and fully connected layers. This enables experimentation with various layer combinations to identify the most effective model for stroke detection.

 Regularization Techniques: To prevent overfitting, the system incorporates regularization techniques such as L2 regularization and dropout layers. These techniques help ensure the model performs well on unseen data by promoting simpler, more generalizable models.

 Training Callbacks: Several callbacks are integrated into the training process to enhance model performance and efficiency. TensorBoard provides real-time tracking and visualization of training metrics. ModelCheckpoint saves the best-performing model based on validation accuracy. EarlyStopping halts training when improvement plateaus, preventing overfitting, and ReduceLROnPlateau adjusts the learning rate based on validation loss to facilitate better convergence.

 Evaluation and Visualization: The system employs advanced evaluation metrics and visualization techniques to monitor model performance. This includes plotting accuracy and loss curves, visualizing predictions on test data, and generating confusion matrices to provide detailed insights into the model's predictive accuracy.

 Scalability: The architecture of the system allows for scalability, making it adaptable for larger datasets and more complex medical imaging tasks in the future.

CHAPTER – 2
LITERATURE SURVEY

2. LITERATURE SURVEY

Brain stroke is a critical health issue requiring timely diagnosis and treatment. Recent advancements in machine learning (ML) and deep learning (DL) have facilitated the development of automated systems for stroke detection and prediction. The following works highlight significant contributions in this domain:

1. **An Efficient Detection Model for Brain Stroke Based on Transfer Learning**

   A study by Mohammed et al. (2024) proposed a transfer learning approach for stroke classification using CNN architectures such as EfficientNet, ResNet, and VGG16. The models achieved an impressive accuracy of 99.95% using a dataset of 2,515 CT images. Preprocessing steps included resizing images to 224×224 pixels, and the models were fine-tuned with additional layers for enhanced performance. The research demonstrates the feasibility of transfer learning for accurate and efficient stroke diagnosis.

2. **Brain Stroke Prediction Using Deep Learning: A CNN Approach**

   Madhavi et al. (2022) utilized CNN architectures like ResNet, MobileNet, and VGG16 to classify stroke images from CT scans. The dataset, obtained from Virinchi Hospital, was preprocessed using gray-scaling and resizing. While the MobileNet model achieved a training accuracy of 98.96%, validation accuracy for certain models remained below 90%, indicating possible overfitting. The study emphasizes the importance of architecture selection and data augmentation in DL-based stroke detection.

3. **Machine Learning Models for Early Brain Stroke Prediction: A Performance Analogy**

   Srivastav et al. (2023) explored multiple ML algorithms for stroke prediction, including Logistic Regression (LR), Decision Trees (DT), Naïve Bayes (NB), and K-Star. Using a dataset of 4,982 records from Kaggle, the LR model achieved the highest accuracy of 95.02%. The research highlighted the critical role of feature engineering and dataset balancing in achieving robust prediction performance.

4. **Brain Stroke Prediction using Convolutional Neural Network and Deep Learning Models**

Gaidhani et al. (2019) developed a dual-system approach combining LeNet for classification and SegNet for segmentation of MRI images. The classification model achieved 97% accuracy, while segmentation reached 85%-87% accuracy in delineating abnormal regions. The dataset comprised 420 MRI scans from the ATLAS repository, preprocessed for normalization and noise reduction. This work emphasizes the utility of segmentation in providing detailed diagnostic insights.

Summary:

The studies reviewed demonstrate the effectiveness of ML and DL techniques in improving the accuracy and efficiency of brain stroke detection. Transfer learning and CNN-based architectures excel in handling image data, while traditional ML models like Logistic Regression are effective for tabular data. Future advancements may focus on integrating multi-modal datasets, hybrid architectures, and enhanced preprocessing techniques to improve prediction accuracy and clinical applicability.

CHAPTER – 3
REQUIREMENT ANALYSIS

3. REQUIREMENT ANALYSIS

The requirement analysis for our brain stroke detection project identifies the key needs essential for its successful development. It involves acquiring a substantial dataset of labelled brain scan images for training the Convolutional Neural Network (CNN) models, along with implementing data augmentation techniques to enhance dataset variability. The project necessitates the use of deep learning frameworks such as TensorFlow, high-performance GPUs for efficient model training, and sufficient storage for datasets and model checkpoints. Additionally, we utilize a pre-trained VGG16 model for transfer learning, coupled with custom CNN architectures and regularization techniques to optimize performance. The integration of training callbacks and advanced evaluation metrics ensures robust model evaluation, while thorough documentation and version control facilitate effective project management. These requirements collectively aim to create a highly accurate and reliable system for early stroke detection, supporting timely medical interventions and improving patient outcomes.

3.1 OPERATING ENVIRONMENT

Programming Language:

Python: The primary programming language for developing the system is Python, due to its extensive libraries and frameworks that support machine learning, deep learning, and data processing.

Deep Learning Frameworks:

TensorFlow and Keras: The system utilizes TensorFlow, an open-source deep learning framework, along with its high-level API, Keras.

Software Libraries:

Data Manipulation and Analysis: NumPy and Pandas are used for efficient data manipulation and analysis.

Data Visualization: Matplotlib and Seaborn are employed for creating informative data visualizations that help in understanding the data distribution and model performance.

Machine Learning Tools: Scikit-learn provides additional machine learning tools and metrics for model evaluation and performance assessment.

Medical Imaging Compatibility:

MRI and CT scans: The system is designed to be compatible with various imaging modalities, including MRI and CT scans (.jpg, .jpeg, .bmp, .png).

Security and Compliance:

Data Privacy: The system must adhere to strict data privacy and security regulations, to ensure the privacy and protection of sensitive medical data.

These features collectively ensure that our system operates efficiently within the healthcare environment, providing a reliable and effective tool for early stroke detection and diagnosis. This comprehensive operating environment supports the system's deployment, integration, and utilization in clinical settings, ultimately enhancing patient outcomes through advanced AI-driven diagnostics.

3.2. FUNCTIONAL REQUIREMENTS

Data Preprocessing:

- The system must preprocess medical imaging data by normalizing, resizing, and augmenting the images.
- It should apply data augmentation techniques like random flips, rotations, and zooms to enhance dataset variability.

Model Training:

- The system must implement Convolutional Neural Networks (CNNs) for image classification.
- It should support transfer learning using a pre-trained VGG16 model with additional custom layers.
- The system must include two custom CNN architectures with different configurations of convolutional, pooling, dropout, and fully connected layers.
- It should incorporate regularization techniques such as regularization and dropout to prevent overfitting.
- The system must perform hyperparameter tuning to optimize the model's performance.

Training Management:

- The system must include training callbacks such as TensorBoard for real-time tracking and visualization of training metrics.
- It should implement ModelCheckpoint to save the best-performing model based on validation accuracy.
- The system must use EarlyStopping to halt training when improvement plateaus, preventing overfitting.
- It should incorporate ReduceLROnPlateau to adjust the learning rate based on validation loss for better convergence.

Model Evaluation:

- The system must evaluate the trained models using accuracy, loss, and confusion matrices to assess performance.
- The system must provide a comprehensive comparison of the VGG16 model and the two custom CNN architectures using the same dataset.

Security and Compliance:

- The system must adhere to data privacy and security regulations.

3.3 NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements for our brain stroke detection system ensure it operates effectively, securely, and efficiently. The system must process and analyse medical imaging data with high accuracy, scalability, and minimal latency. The system needs to be reliable, maintainable, and compatible with existing hospital IT infrastructure, including PACS. Additionally, it should support modular design for easy updates, provide secure data storage, ensure accessibility, and maintain compliance with healthcare standards to aid medical professionals in timely and accurate stroke diagnosis and treatment.

3.4 HARDWARE & SOFTWARE REQUIREMENTS

- **Processor** – i5, M1 ARM
- **RAM** – 4 GB (Minimum)
- **Hard Disk** – 256GB SSD
- **Operating System** – Windows, MacOS
- **IDE** - Jupiter

3.5 FEASIBILITY STUDY

A feasibility study assesses the practicality and potential success of a project by examining various critical factors. It typically includes an analysis of technical feasibility, evaluating if the technology and tools required for the project are available and capable of delivering the desired outcomes. Economic feasibility focuses on the cost-effectiveness and financial benefits, weighing the project's costs against its potential financial gains and savings. Social feasibility considers the project's impact on society, including potential benefits to the community and alignment with social values and needs. Legal and ethical feasibility ensures compliance with relevant laws, regulations, and ethical standards. Operational feasibility assesses whether the project's implementation and integration into existing systems are achievable and sustainable. By examining these factors, a feasibility study provides a comprehensive overview of a project's viability.

3.5.1 ECONOMIC FEASIBILITY

The economic feasibility of implementing AI in healthcare, particularly for brain stroke detection, is promising. AI has the potential to significantly reduce costs by improving diagnostic accuracy and efficiency, thereby reducing the need for expensive and time-consuming tests. Additionally, AI can help in early detection of strokes, which can lead to better patient outcomes and lower long-term healthcare costs. However, there are challenges to consider, such as the initial investment in AI infrastructure and the need for ongoing maintenance and updates. Despite these challenges, the potential cost savings and improvements in patient care make AI a viable and economically feasible option for healthcare systems.

3.5.2 TECHNICAL FEASIBILITY

The technical feasibility of our brain stroke detection system is highly promising, given the advancements in artificial intelligence and deep learning. Modern deep learning frameworks like TensorFlow and PyTorch, combined with powerful computational hardware (such as high-performance GPUs), enable the efficient training and deployment of Convolutional Neural Networks (CNNs) for medical image analysis. The availability of pre-trained models like VGG16 for transfer learning further accelerates development, allowing us to leverage established feature extraction capabilities. Additionally, robust data preprocessing and augmentation techniques ensure that the system can handle the variability in medical imaging data, making the project technically viable and effective in improving stroke diagnosis.

CHAPTER – 4
SYSTEM DESIGN

4. SYSTEM DESIGN

System design entails conceptualizing, planning, and creating a blueprint, pattern, or sketch. This process translates a logical representation of the system's requirements into a physical reality during development. The goal of system design is to take the system's description and align it with specific facilities, machines (computing and otherwise), and accommodations, to provide detailed specifications of a functional system. The design must adhere to constraints and improve upon the existing system.

SOFTWARE DESIGN

Software Design is a subset of system design focused on the software components. It includes planning and creating software solutions to meet the requirements specified during system analysis.

INPUT DESIGN

Taking into account the requirements, procedures are established to gather the necessary input data in the most efficient format. The input design ensures that user interaction with the system is both effective and straightforward. It aims to control the volume of input and prevent unauthorized access. During this stage, input forms and screens are designed.

- **Medical Imaging Data**: MRI or CT scans of the brain, which need to be processed and analyzed by the system.
- **Preprocessing Parameters**: Settings for normalizing, resizing, and augmenting the medical images.
- **Model Configuration**: Specifications for the architecture of the Convolutional Neural Networks (CNNs), including the use of pre-trained models like VGG16 and custom CNN architectures.
- **Training Data**: A dataset of labeled medical images used to train the CNN models.
- **Hyperparameters**: Values such as learning rate, batch size, and the number of layers/neurons that need to be optimized during model training.

- **Evaluation Criteria**: Metrics and thresholds for assessing model performance, such as accuracy, loss, and confusion matrices.

OUTPUT DESIGN

Every screen in the system is crafted to offer users a straightforward and efficient experience, minimizing the number of keystrokes required. Key information is highlighted to draw attention. Nearly all screens are designed to be error-free, including essential messages and options for selection to facilitate smooth operations.

- **Trained Model**: The final CNN model trained and fine-tuned to detect brain strokes with high accuracy.
- **Diagnostic Results**: Predictions generated by the model, indicating the presence or absence of a stroke in the input medical images.
- **Visualizations**: Graphs and plots showing model performance, such as accuracy and loss curves, confusion matrices, and heatmaps of model predictions.
- **Reports**: Summaries of model evaluation and performance metrics, which can be used by medical professionals to understand the model's effectiveness.

UML DIAGRAMS

4.1 SYSTEM ARCHITECTURE MODEL

System architecture, illustrated in Fig 4.1, is a high-level framework detailing the structure and interconnections of a complex system. It encompasses the design of software, hardware, networks, and interfaces to facilitate smooth interactions and achieve objectives. This architecture outlines functionality distribution, communication protocols, and data flow, ensuring optimal performance, scalability, and reliability. It serves as a foundation for development, integration, and maintenance, supporting system evolution and adaptation to changing requirements and technological advancements.

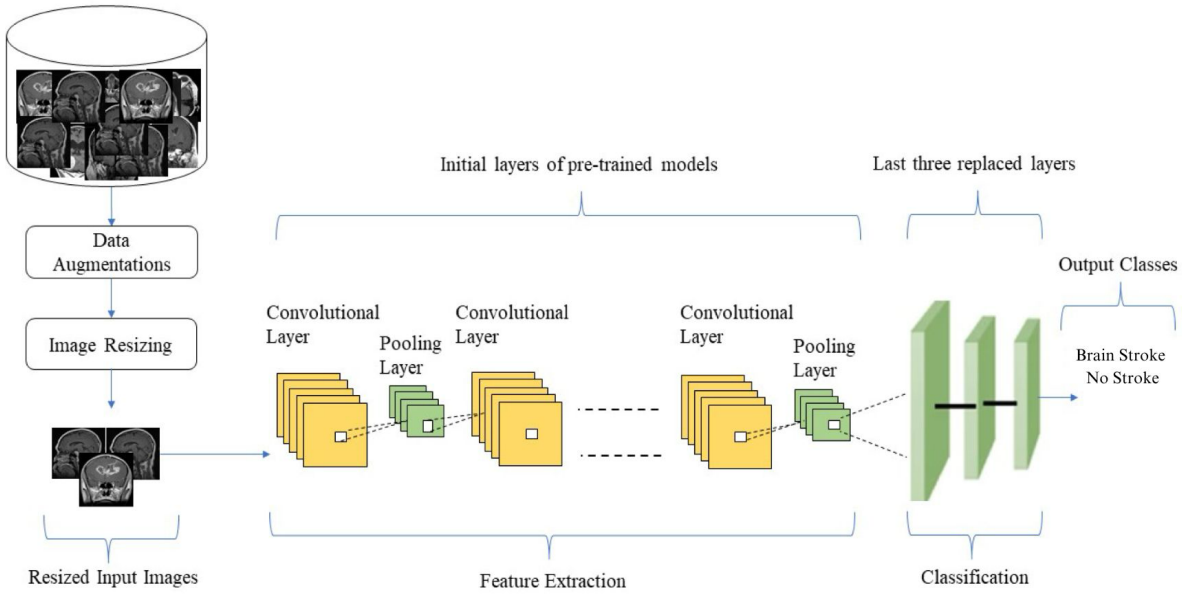


Fig 4.1

4.2 CLASS DIAGRAM

A class diagram is a type of static structure diagram in Unified Modeling Language (UML) that describes the structure of a system by showing its classes, attributes, operations, and the relationships among objects. It's a blueprint for creating an object-oriented system.

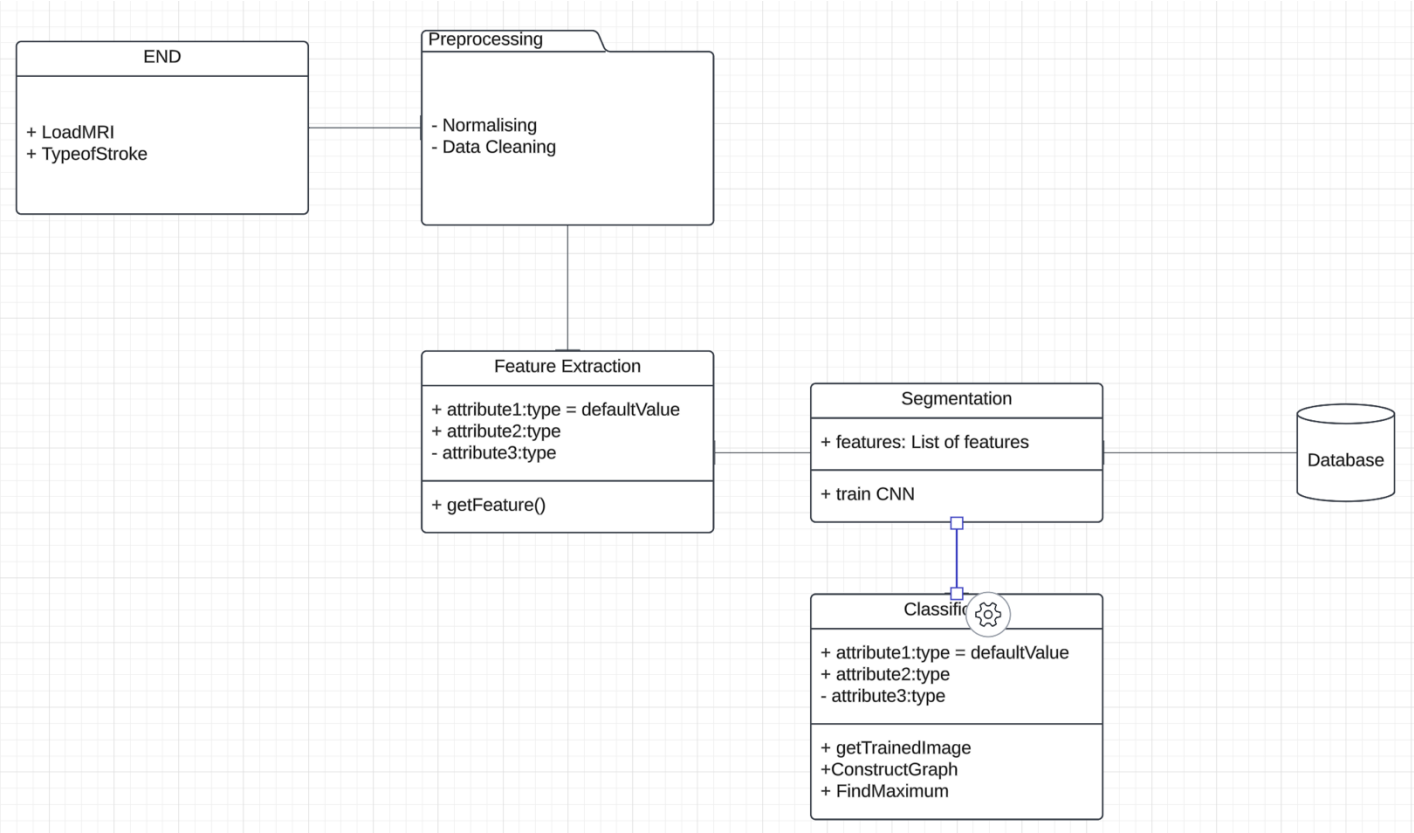


Fig 4.2

4.3 BLOCK DIAGRAM

A block diagram is a graphical representation used to visualize the structure and components of a system. It uses labeled blocks to represent distinct components or functions and arrows to show the relationships and data flow between these blocks, providing a high-level overview of the system's operation.

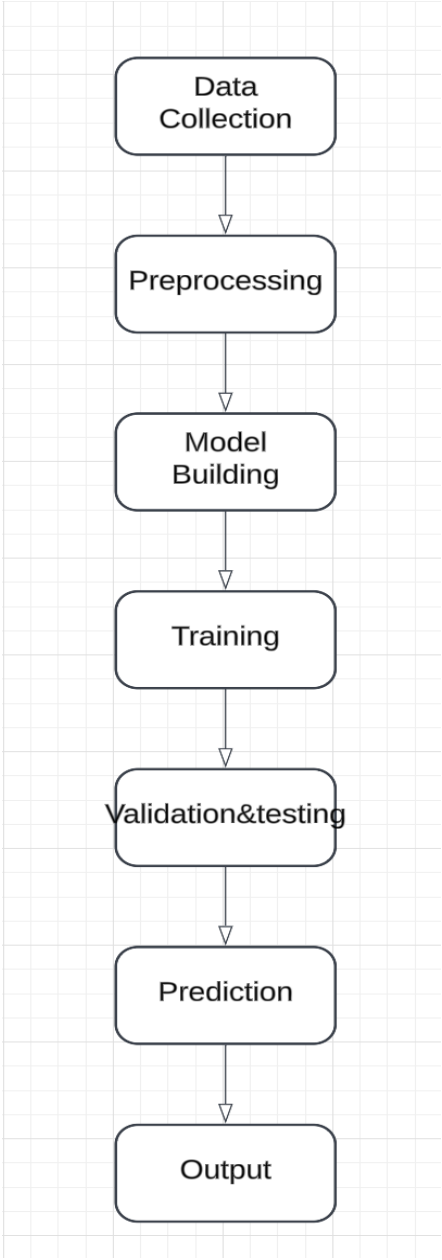


Fig 4.3

4.4 USE CASE

Use case diagrams are a fundamental aspect of the Unified Modeling Language (UML) used to visualize the functional requirements of a system. They provide a high-level overview of how users (actors) interact with the system to achieve specific goals (use cases). These diagrams capture the system's intended behavior and identify the primary functionalities from the user's perspective.

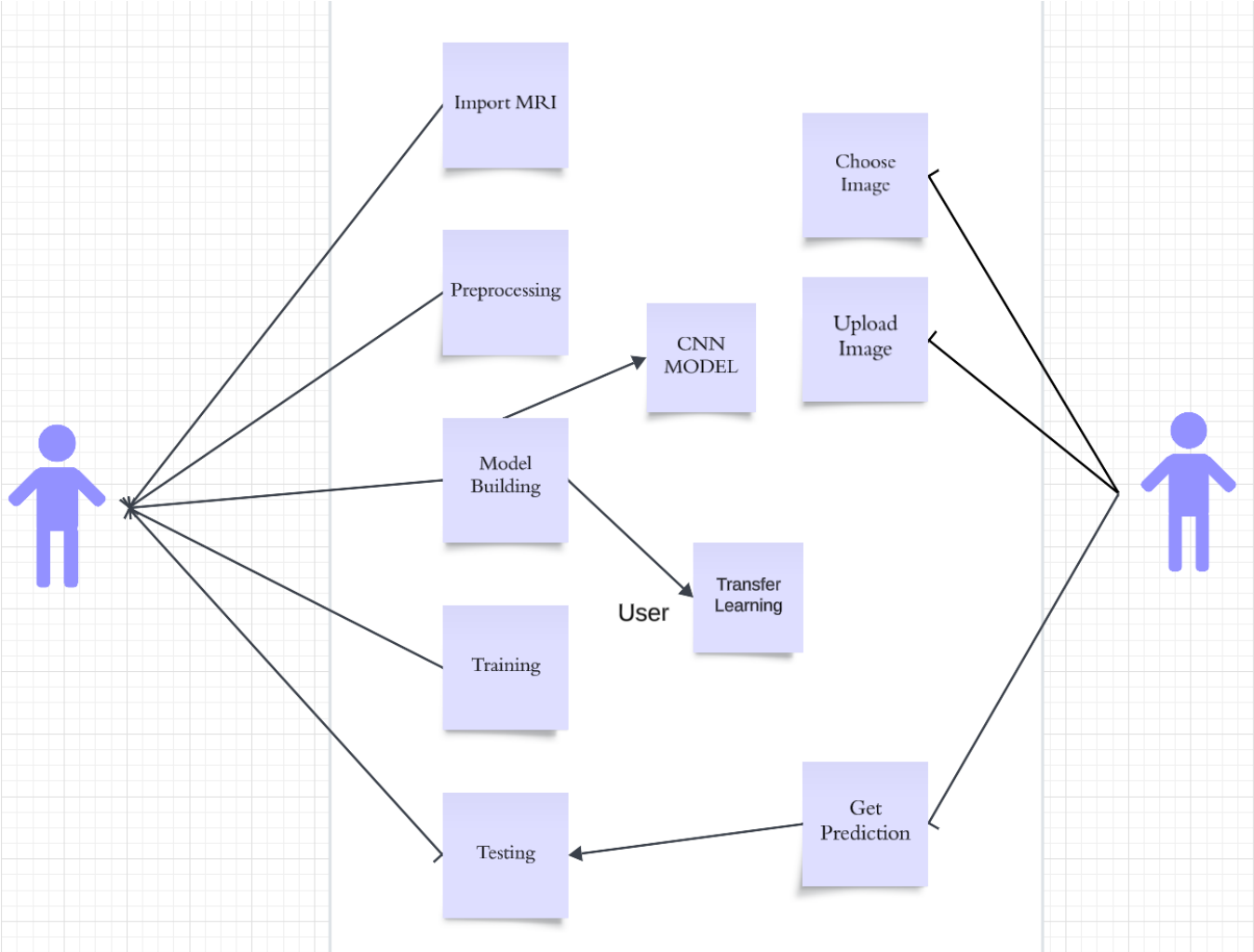


Fig 4.4

4.5 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram in the Unified Modeling Language (UML) that shows how objects interact in a particular scenario of a use case. It illustrates the sequence of messages exchanged between objects to carry out a specific process or function, detailing the time order of the interactions.

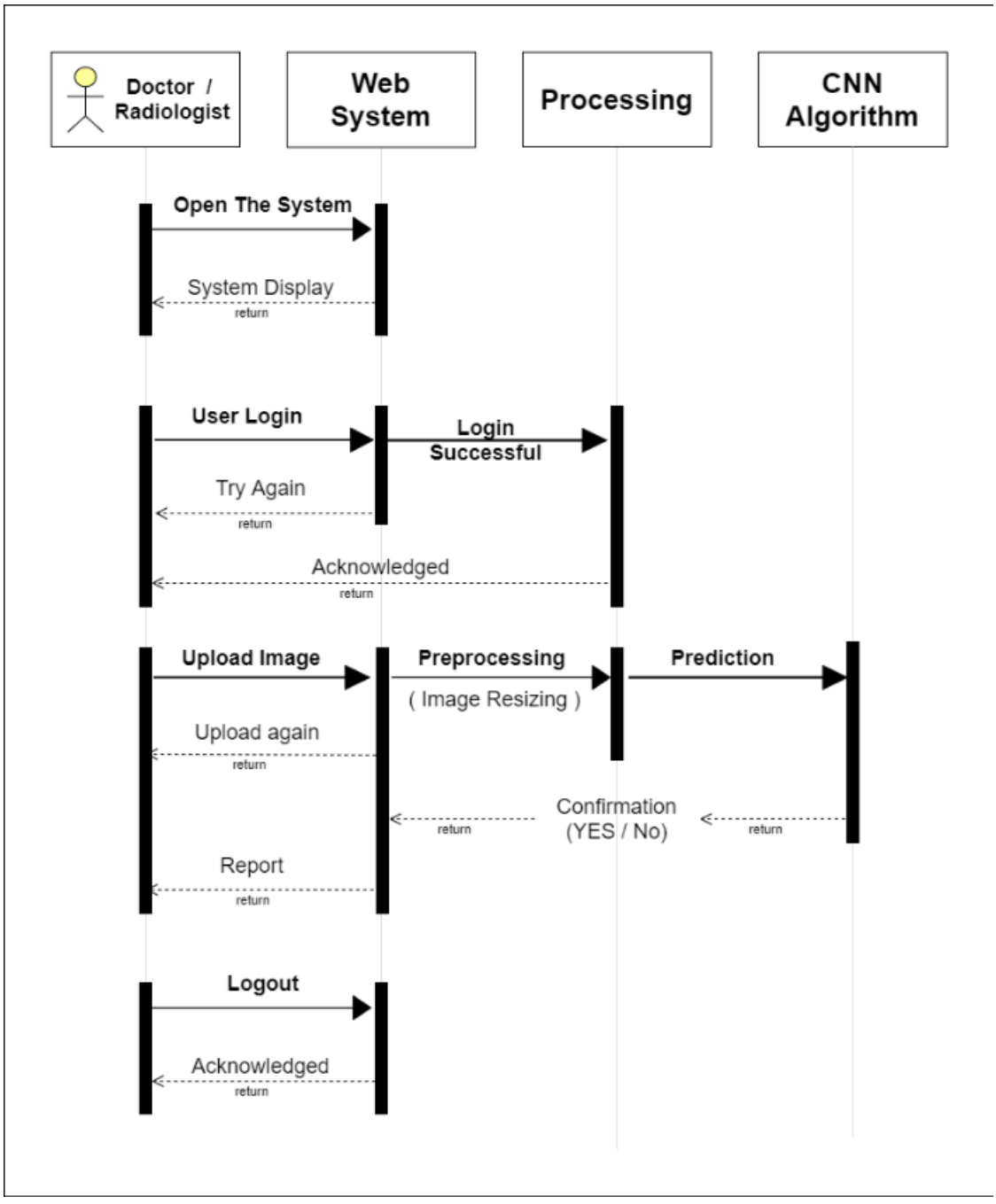


Fig. 4.5

CHAPTER – 5
IMPLEMENTATION

5. IMPLEMENTATION

Implementation refers to the process of executing and integrating the planned system design into a functional and operational system. This phase involves transforming design specifications into actual software or hardware components, ensuring that each part works correctly and cohesively. Implementation is crucial as it brings the theoretical design to life, allowing for real-world application and usage.

KEY FUNCTIONS

- **Code Development**:
  - Writing the source code based on design documents.
  - Ensuring the code adheres to specified requirements and standards.
- **Debugging**:
  - Identifying and resolving defects or issues found during testing.
  - Ensuring the system performs as expected under various conditions.
- **Evaluation**:
  - Assessing the system's performance and effectiveness post-deployment.
  - Collecting feedback and making necessary adjustments or improvements.

CONFIGURATION

In this project, several essential libraries and frameworks were used to build, train, and evaluate the Convolutional Neural Networks (CNNs) for brain stroke detection. Here is an overview of the libraries used and their purposes:

1. **NumPy**:
   - **Purpose**: Provides support for large multi-dimensional arrays and matrices with a collection of mathematical functions to operate on these arrays.
2. **TensorFlow**:
   - **Purpose**: An open-source machine learning framework developed by Google, used for training and deploying machine learning models.

3. **Seaborn**:

    - **Purpose**: A data visualization library based on Matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics.

4. **Scikit-learn**:

    - **Purpose**: A machine learning library that provides simple and efficient tools for data mining and data analysis.

5. **TensorFlow Addons**:

    - **Purpose**: A repository of contributions that follow the TensorFlow extended standards, providing additional functionality.

5.2 METHOD OF IMPLMENTATION

This project harnesses the power of Python, chosen for its effectiveness, versatility, and intuitive design. As an open-source language, Python fosters accessibility and collaboration, making it a top choice for developers. Its emphasis on code readability, achieved through significant indentation, ensures clarity and maintainability. Additionally, Python's support for modules and packages promotes code reuse and modularity, significantly enhancing productivity and efficiency throughout the development process.

CHAPTER – 6
MODULES

6. MODULES

Cleaning Dataset:

```
import tensorflow as tf
import os
import cv2
import imghdr

DatasetDir = 'Dataset'

ImageExts = ['jpeg', 'jpg', 'bmp', 'png']

for ImageClass in os.listdir(DatasetDir):
    for Image in os.listdir(os.path.join(DatasetDir, ImageClass)):
        ImagePath = os.path.join(DatasetDir, ImageClass, Image)
        try:
            Img = cv2.imread(ImagePath)
            ImgExt = imghdr.what(ImagePath)
            if ImgExt not in ImageExts:
                print(f'This image is going to be deleted because it\'s extension is not compatible : {ImagePath}')
                os.remove(ImagePath)
        except Exception as e:
            print(f'Issue with image {ImagePath}')
```

Code 1: Basic CNN

Loading Dataset

```
import numpy as np
from matplotlib import pyplot as plt

TrainingData = tf.keras.utils.image_dataset_from_directory('Dataset/Train').map(lambda x,y: (x/255, y))
TestingData = tf.keras.utils.image_dataset_from_directory('Dataset/Test').map(lambda x,y: (x/255, y))
ValidationData = tf.keras.utils.image_dataset_from_directory('Dataset/Validation').map(lambda x,y: (x/255, y))

DataIterator = TrainingData.as_numpy_iterator()

Batch = DataIterator.next()

Fig, ax = plt.subplots(ncols=4, figsize=(20,20))

for idx, img in enumerate(Batch[0][:4]):
    ax[idx].imshow(img.astype(float))
    ax[idx].title.set_text(Batch[1][idx])
```

Training Model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten

Model = Sequential()

Model.add(Conv2D(16, (3,3), 1, activation='relu', input_shape=(256,256,3)))
Model.add(MaxPooling2D())
Model.add(Conv2D(32, (3,3), 1, activation='relu'))
Model.add(MaxPooling2D())
Model.add(Conv2D(16, (3,3), 1, activation='relu'))
Model.add(MaxPooling2D())
Model.add(Flatten())
Model.add(Dense(256, activation='relu'))
Model.add(Dense(1, activation='sigmoid'))

Model.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy'])

Model.summary()
```

```
LogDir='logs'
Tensorboard_Callback = tf.keras.callbacks.TensorBoard(log_dir=LogDir)
History = Model.fit(TrainingData, epochs=20, validation_data=ValidationData, callbacks=[Tensorboard_Callback])
```

Model Performance

```
Fig = plt.figure()
plt.plot(History.history['loss'], color='red', label='Loss')
plt.plot(History.history['val_loss'], color='black', label='Validation_Loss')
Fig.suptitle('Loss', fontsize=20)
plt.legend(loc="upper left")
plt.show()
```

```
Fig = plt.figure()
plt.plot(History.history['accuracy'], color='red', label='Accuracy')
plt.plot(History.history['val_accuracy'], color='black', label='Validation_Accuracy')
Fig.suptitle('Accuracy', fontsize=20)
plt.legend(loc="upper left")
plt.show()
```

Code 2: Enhanced CNN with VGG16

Loading Dataset

```
import numpy as np
import tensorflow as tf
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Data augmentation and normalization
from tensorflow.keras.layers import Rescaling, RandomFlip, RandomRotation, RandomZoom, Flatten, Dense, Dropout,
from tensorflow.keras.applications import VGG16

DataAugmentation = tf.keras.Sequential([
    Rescaling(1./255),
    RandomFlip("horizontal_and_vertical"),
    RandomRotation(0.3),  # Increased rotation
    RandomZoom(0.3)       # Increased zoom
])

TrainingData = tf.keras.utils.image_dataset_from_directory(
    'Dataset/Train'
).map(lambda x, y: (DataAugmentation(x), y))

ValidationData = tf.keras.utils.image_dataset_from_directory(
    'Dataset/Validation'
).map(lambda x, y: (DataAugmentation(x), y))

TestingData = tf.keras.utils.image_dataset_from_directory(
    'Dataset/Test', shuffle=False
).map(lambda x, y: (Rescaling(1./255)(x), y))
```

CNN model with VGG16

```
# Define the enhanced CNN model with VGG16
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(256, 256, 3))
base_model.trainable = False  # Freeze base model layers

Model = tf.keras.Sequential([
    base_model,
    Flatten(),
    Dense(256, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001)),
    Dropout(0.5),
    BatchNormalization(),
    Dense(128, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001)),
    Dense(1, activation='sigmoid')
])

# Compile the model
Model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
              loss=tf.keras.losses.BinaryCrossentropy(),
              metrics=['accuracy'])

# Set up callbacks
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau, TensorBoard

LogDir = 'logs'
Tensorboard_Callback = TensorBoard(log_dir=LogDir)
Checkpoint_Callback = ModelCheckpoint('best_model.keras', monitor='val_accuracy', save_best_only=True, mode='max
EarlyStopping_Callback = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
ReduceLROnPlateau_Callback = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=3, min_lr=0.00001)

class CustomCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs=None):
        print(f"Epoch {epoch+1}: Training Accuracy: {logs['accuracy']:.2f}, Validation Accuracy: {logs['val_accu

# Train the model
History = Model.fit(
    TrainingData,
    epochs=20,
    validation_data=ValidationData,
    callbacks=[Tensorboard_Callback, Checkpoint_Callback, EarlyStopping_Callback, ReduceLROnPlateau_Callback, Cu
)
```

Evaluation

```
# Evaluate the model on the testing data
test_loss, test_accuracy = Model.evaluate(TestingData)
print(f"Test Accuracy: {test_accuracy:.2f}, Test Loss: {test_loss:.2f}")

# Visualize predictions on a batch of testing data
DataIterator = TestingData.as_numpy_iterator()
Batch = DataIterator.next()
```

Prediction and Plotting

```
# Make predictions
predictions = Model.predict(Batch[0])

# Display images with predictions
fig, ax = plt.subplots(1, 4, figsize=(20, 20))
for idx in range(min(4, len(Batch[0]))):  # Ensure batch size is handled dynamically
    ax[idx].imshow(Batch[0][idx].astype(float))
    true_label = Batch[1][idx]
    predicted_label = (predictions[idx] > 0.5).astype(int)  # Convert probabilities to binary labels
    ax[idx].set_title(f'True: {true_label}, Pred: {predicted_label}')
    ax[idx].axis('off')

plt.show()

# Plotting accuracy and loss
plt.figure(figsize=(12, 4))

# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(History.history['accuracy'], label='Training Accuracy')
plt.plot(History.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title('Accuracy')

# Loss plot
plt.subplot(1, 2, 2)
plt.plot(History.history['loss'], label='Training Loss')
plt.plot(History.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Loss')

plt.show()

# Confusion matrix for testing data
y_true = np.concatenate([y for x, y in TestingData], axis=0)
y_pred = np.concatenate([Model.predict(x) > 0.5 for x, y in TestingData], axis=0)

conf_matrix = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```

Code 3: Enhanced CNN with Custom Layers

Loading Dataset

```
import numpy as np
import tensorflow as tf
from matplotlib import pyplot as plt

# Load and normalize datasets
TrainingData = tf.keras.utils.image_dataset_from_directory(
    'Dataset/Train'
).map(lambda x, y: (x / 255, y))

TestingData = tf.keras.utils.image_dataset_from_directory(
    'Dataset/Test', shuffle=False  # Disable shuffling for testing data
).map(lambda x, y: (x / 255, y))

ValidationData = tf.keras.utils.image_dataset_from_directory(
    'Dataset/Validation'
).map(lambda x, y: (x / 255, y))
```

File display

Custom CNN Model

```
# Define the enhanced CNN model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout, BatchNormalization

Model = Sequential([
    Conv2D(16, (3, 3), activation='relu', input_shape=(256, 256, 3)),
    BatchNormalization(),
    MaxPooling2D(),
    Conv2D(32, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(),
    Dropout(0.25),
    Conv2D(64, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(),
    Conv2D(32, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(),
    Dropout(0.25),
    Conv2D(16, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(),
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    BatchNormalization(),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

Compilation

```
# Compile the model
Model.compile(optimizer='adam',
              loss=tf.losses.BinaryCrossentropy(),
              metrics=['accuracy'])
# Set up TensorBoard logging
LogDir = 'logs'
Tensorboard_Callback = tf.keras.callbacks.TensorBoard(log_dir=LogDir)
# Train the model
History = Model.fit(
    TrainingData,
    epochs=20,
    validation_data=ValidationData,
    callbacks=[Tensorboard_Callback]
)
```

Evaluate

```
# Evaluate the model on the testing data
test_loss, test_accuracy = Model.evaluate(TestingData)
print(f"Test Accuracy: {test_accuracy:.2f}, Test Loss: {test_loss:.2f}")
# Visualize predictions on a batch of testing data
DataIterator = TestingData.as_numpy_iterator()
Batch = DataIterator.next()
```

Predictions

```
# Make predictions
predictions = Model.predict(Batch[0])

# Display images with predictions
fig, ax = plt.subplots(1, 4, figsize=(20, 20))
for idx in range(min(4, len(Batch[0]))):  # Ensure batch size is handled dynamically
    ax[idx].imshow(Batch[0][idx].astype(float))
    true_label = Batch[1][idx]
    predicted_label = (predictions[idx] > 0.5).astype(int)  # Convert probabilities to binary labels
    ax[idx].set_title(f'True: {true_label}, Pred: {predicted_label}')
    ax[idx].axis('off')

plt.show()
```

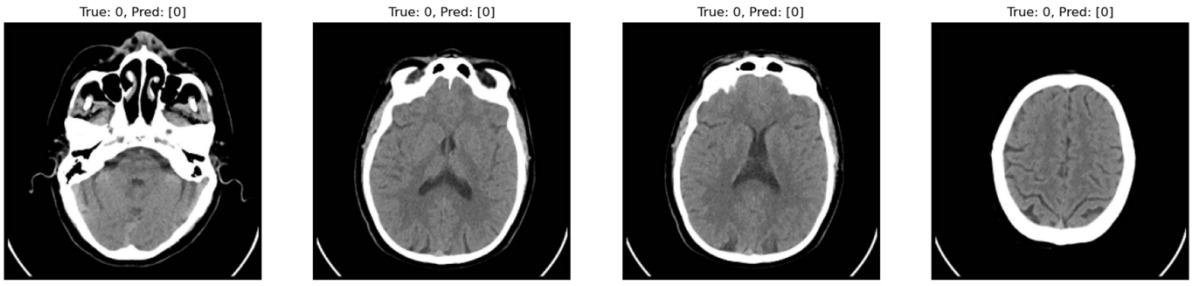
Plotting Accuracy and loss

```
# Plotting accuracy and loss
plt.figure(figsize=(12, 4))

# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(History.history['accuracy'], label='Training Accuracy')
plt.plot(History.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title('Accuracy')

# Loss plot
plt.subplot(1, 2, 2)
plt.plot(History.history['loss'], label='Training Loss')
plt.plot(History.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Loss')

plt.show()
```

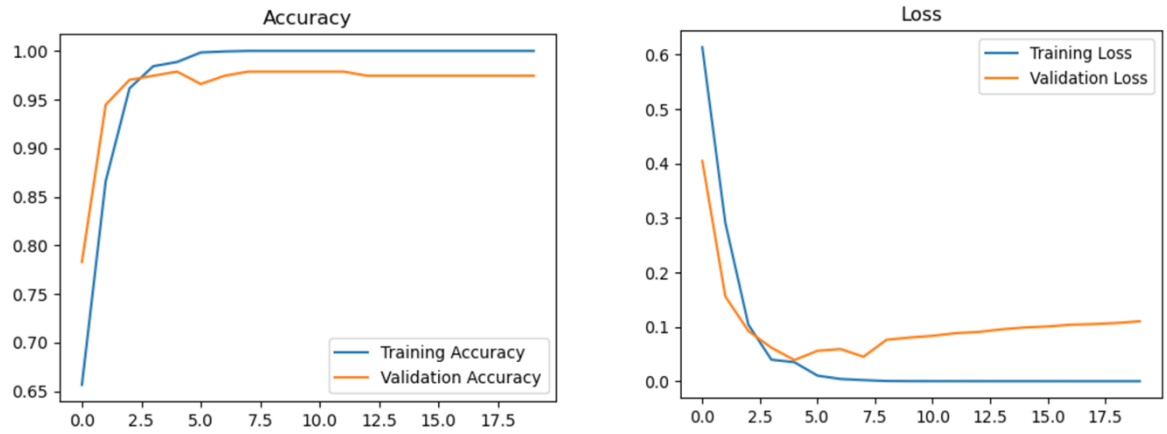
CHAPTER – 7
OUTPUT SCREENS

7. OUTPUT SCREENS

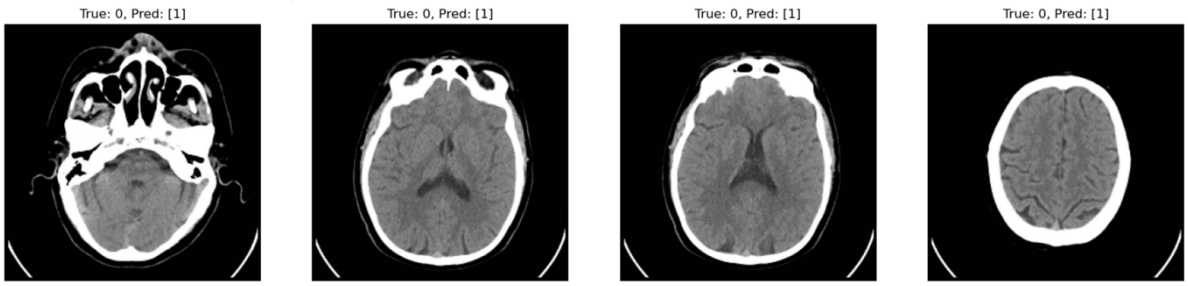
Code 1: Basic CNN

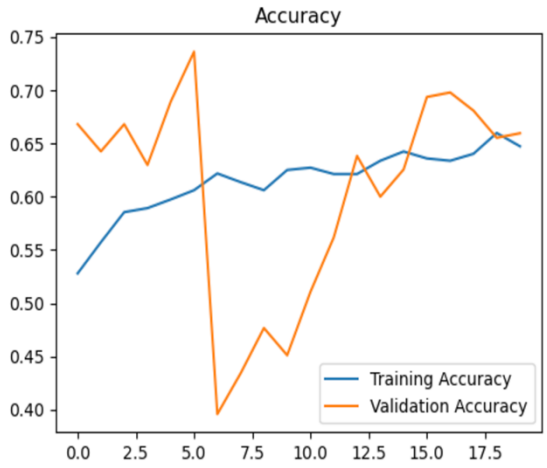


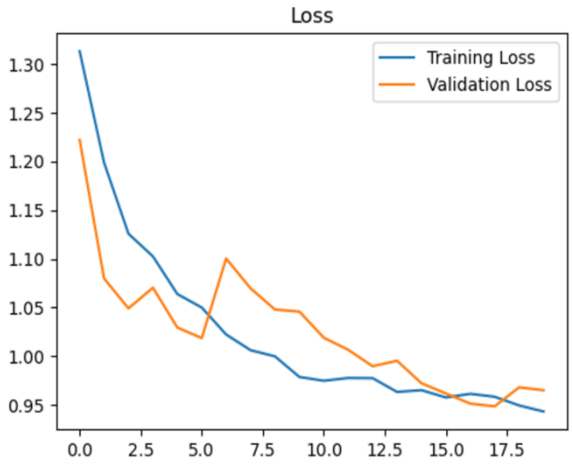
This code randomly picks up few samples out from the dataset and performs the model testing, we can see the true[0] and pred[0]. In case both of them are true, then the prediction is true, else it's false.



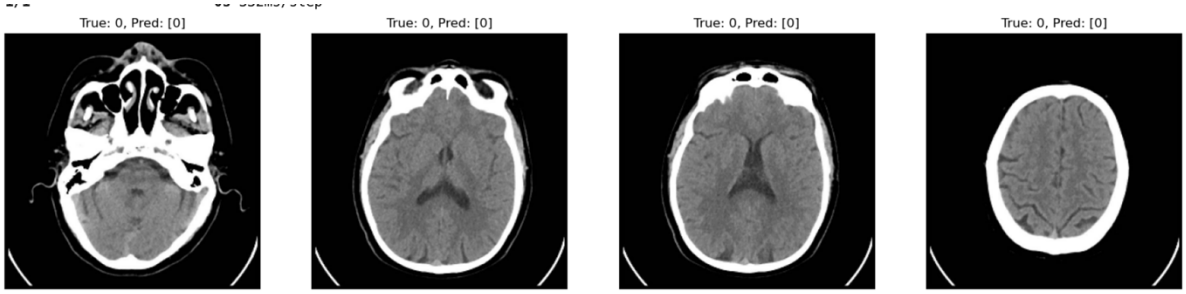
Code 2: Enhanced CNN with VGG16

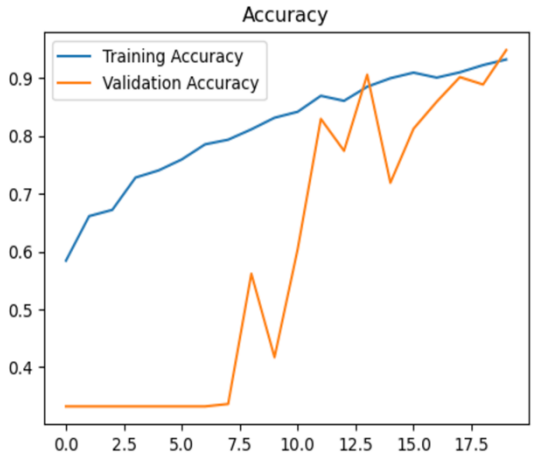






Code 3: Enhanced CNN with Custom Layers





CHAPTER – 8
CONCLUSION & FUTURE SCOPE

8.1 CONCLUSION

To conclude, our project embodies a trailblazing effort in AI-driven healthcare, leveraging cutting-edge advancements to enhance brain stroke diagnostics. By meticulously preprocessing medical imaging data and integrating advanced CNN architectures, including basic models, custom layers, and the pre-trained VGG16 network, our system ensures high accuracy and reliability. The use of data augmentation techniques, hyperparameter fine-tuning, and innovative training strategies such as early stopping, model checkpointing, and learning rate adjustments further bolsters the model's performance. Comprehensive evaluation through metrics and visualization tools, including confusion matrices and accuracy and loss plots, allows for detailed performance assessment and continuous improvement.

This project stands out for its ability to seamlessly integrate advanced AI techniques with practical healthcare applications, providing medical professionals with a reliable, efficient, and user-friendly diagnostic tool. The system's robustness across varying conditions and its high performance ensure it meets the stringent requirements of clinical practice. By harnessing the power of AI, we are able to significantly enhance diagnostic accuracy and speed, offering a transformative solution that transcends traditional diagnostic methods.

Ultimately, our project represents a significant step forward in the field of medical diagnostics, empowering medical professionals to make timely and accurate diagnoses. This not only improves patient outcomes by facilitating early intervention but also sets the stage for continued innovation and impactful contributions to global healthcare. By leveraging AI, we aspire to support clinical decision-making, reduce the burden on medical professionals, and ultimately save lives through enhanced diagnostic capabilities.

8.2 FUTURE SCOPE

The success of this project opens many future opportunities in AI-driven medical diagnostics. Expanding the dataset to include more diverse medical images from various institutions and demographics will enhance the model's robustness and generalizability, addressing biases and ensuring reliable performance in different clinical settings.

Exploring advanced architectures like ResNet, DenseNet, and EfficientNet, known for superior image classification, can further improve diagnostic accuracy. Integrating these with CNNs and Recurrent Neural Networks (RNNs) for time-sequence data may capture temporal dynamics in medical imaging. Implementing the model in real-time clinical environments by integrating with hospital IT infrastructure like PACS and EHR systems will provide instant diagnostic support, enhancing the efficiency and accuracy of medical decision-making.

Additionally, combining the CNN-based stroke detection system with other diagnostic methods, such as genomic data analysis or patient history, will create a comprehensive diagnostic tool. Enhancing the user experience with intuitive controls, advanced visualization tools, and possibly augmented reality (AR) for better image interpretation will ensure widespread adoption. Leveraging cloud computing technologies will address computational demands, facilitating large-scale deployment. Ensuring regulatory compliance and addressing ethical considerations will safeguard patient data privacy and enhance AI trustworthiness in healthcare. Continuous learning, regular updates, and interdisciplinary collaboration with medical professionals will refine and validate the models, ensuring they meet clinical standards and deliver practical benefits. Evaluating the system's impact on patient outcomes through early detection rates, treatment effectiveness, and patient satisfaction will provide valuable insights, guiding further improvements.

In summary, the future scope of this project is vast, with the potential to significantly advance medical diagnostics. Integrating advanced AI technologies with healthcare practices can lead to early, accurate diagnosis, ultimately improving global patient care and outcomes.

REFERENCES

[1]. S. Prasher *et al.*, "Brain Stroke Prediction Using EfficientNet-B0," *2024 INCET*, pp. 1-4, doi: 10.1109/INCET61516.2024.10593220.

[2]. F. Aboudi *et al.*, "Efficient U-Net CNN for MRI Stroke Segmentation," *2022 CoDIT*, pp. 724-728, doi: 10.1109/CoDIT55151.2022.9804030.

[3]. M. K. Reddy *et al.*, "Brain Stroke Prediction with CNN," *2022 ICIRCA*, pp. 775-780, doi: 10.1109/ICIRCA54612.2022.9985596.

[4]. V. Bandi *et al.*, "Prediction of Stroke Severity Using ML," *Revue d'Intelligence Artificielle*, 34(6), pp. 753-761, doi: 10.18280/ria.340609.

[5]. I. Schiopu *et al.*, "CNN-Based Lossless Image Coding," *2018 PCS*, pp. 16-20, doi: 10.1109/PCS.2018.8456311.

[6]. M. S. Sirsat *et al.*, "ML for Brain Stroke: A Review," *Journal of Stroke and Cerebrovascular Diseases*, 29(10), doi: 10.1016/j.jstrokecerebrovasdis.2020.105162.

[7]. M. U. Emon *et al.*, "ML Approaches in Stroke Prediction," *2020 ICECA*, pp. 1464-1469, doi: 10.1109/ICECA49313.2020.9297525.

[8]. E. Dritsas & M. Trigka, "Stroke Risk Prediction with ML," *Sensors*, 22(13), doi: 10.3390/s22134670.

[9]. A. N. Tusher *et al.*, "Early Stroke Prediction Using ML," *2022 SMART*, pp. 1280-1284, doi: 10.1109/SMART55829.2022.10046889.

[10]. V.S. Telu *et al.*, "Optimizing Stroke Prediction with ML," *JNFS*, 2(2), pp. 31-43.

[11]. P. Bentley *et al.*, "Stroke Thrombolysis Outcome Prediction," *NeuroImage: Clinical*, 4, pp. 635-640.