RSET
RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)

Project Report on

Brain Hemorrhage Detection using Deep Learning and Explainable AI

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

By

Ranjit Shine (U2103170)

Reuben Simon George (U2103173)

Rohan Thundil Rajeev (U2103182)

Ron Thomas Vijy (U2103184)

Under the guidance of

Ms. Sreejadevi P

Department of Computer Science and Engineering
Rajagiri School of Engineering & Technology (Autonomous)
(Parent University: APJ Abdul Kalam Technological University)
Rajagiri Valley, Kakkanad, Kochi, 682039
April 2025

CERTIFICATE

This is to certify that the project report entitled ***"Brain Hemorrhage Detection using Deep Learning and Explainable AI"*** *is a bonafide record of the work done by* ***Ranjit Shine (U2103170), Reuben Simon George (U2103173), Rohan Thundil Rajeev (U2103182), Ron Thomas Vijy (U2103184)*** *submitted to the Rajagiri School of Engineering & Technology (RSET) (Autonomous) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in Computer Science and Engineering during the academic year 2024-2025.*

Ms. Sreejadevi P
Project Guide
Assistant Professor
Dept. of CSE
RSET

Ms. Sangeetha Jamal
Project Co-ordinator
Assistant Professor
Dept. of CSE
RSET

Dr. Preetha K G
Professor & HOD
Dept. of CSE
RSET

ACKNOWLEDGMENT

We wish to express our sincere gratitude towards **Rev. Dr. Jaison Paul Mulerikkal CMI** , Principal of RSET, and **Dr. Preetha K G**, Head of the Department of Computer Science and Engineering for providing us with the opportunity to undertake our project, "Brain Hemorrhage Detection using Deep Learning and Explainable AI".

We are highly indebted to our project coordinators **Ms. Sangeetha Jamal**, Assistant Professor, Department of Computer Science and Engineering and **Dr. Preetha K G**, Professor & HOD, Department of Computer Science and Engineering for their valuable support.

It is indeed our pleasure and a moment of satisfaction for us to express our sincere gratitude to our project guide **Ms. Sreejadevi P**, Assistant Professor, Department of Computer Science and Engineering for her patience and all the priceless advice and wisdom she has shared with us.

Last but not the least, We would like to express our sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

Ranjit Shine

Reuben Simon George

Rohan Thundil Rajeev

Ron Thomas Vijy

Abstract

Brain hemorrhage is a life-threatening medical condition that demands quick and correct diagnosis to avoid grave complications. Conventional diagnosis uses radiologists to read CT brain scans, a time-consuming, expertise-based, and vulnerable-to-human-mistakes process. This research project aims at a comparative evaluation of deep learning models—VGG-16, ResNet-50, EfficientNet-B0, and DenseNet-121—to classify brain hemorrhages into four groups: lobar, subtentorial, deep, and no hemorrhage.
indent The project includes data preprocessing, model training, and performance metrics based on accuracy, precision, recall, and F1-score. For increased interpretability, Grad-CAM is incorporated into the system, an Explainable AI method that identifies regions impacting model predictions. A user-friendly interface is also created where users can upload CT scans and visualize model predictions as well as highlighted hemorrhage regions.

Through rigorous comparison of several deep learning models and combining explainability, this research will determine the best model for brain hemorrhage classification while promoting transparency and trust in AI-based medical diagnosis.

Contents

List of Abbreviations

CAM: Class Activation Mapping

CNN: Convolutional Neural Network

CT Scan: Computed Tomography Scan

DL: Deep Learning

AI: Artificial Intelligence

MRI Scan: Magnetic Resonance Imaging Scan

XAI: Explainable Artificial Intelligence

BO: Bayesian Optimization

Grad-CAM: Gradient-weighted Class Activation Mapping

VGG: Visual Geometry Group

List of Figures

Chapter 1

Introduction

A brain hemorrhage is a life-threatening medical condition if not treated quickly. Doctors depend on brain scans like CT or MRI images to identify such issues. However, interpreting such scans can be slow and sometimes inaccurate, particularly during emergency cases where every minute matters. This project aims to develop an automated system that can quickly and accurately classify hemorrhages using deep learning techniques. By leveraging CNN-based models such as EfficientNet-B0, VGG-16, ResNet-50, and DenseNet-121, the system analyzes brain scans to identify hemorrhage types, helping doctors make faster and more accurate decisions.

1.1 Background

Brain hemorrhage is a serious medical condition caused by bleeding within the brain. It can result from trauma, high blood pressure, aneurysms, or other medical conditions. If it is undiagnosed or untreated, it can lead to permanent brain damage or even death.

Traditionally, brain hemorrhage diagnosis has depended on radiologists interpreting brain scans, such as CT or MRI scans. However, interpreting these images requires high expertise and experience. Even skilled radiologists could miss subtle hemorrhages, leading to delayed treatment and poor patient outcomes. Additionally, the process is time-consuming, which is very important in emergency situations where every second matters.

With the rise of AI and DL, automated medical image analysis has gained attention to assist radiologists in detecting brain hemorrhages. CNN-based models, such as EfficientNet, VGG-16, ResNet, and DenseNet, have shown impressive performance in recognizing patterns in medical images that might be difficult for humans to detect. By training these models on large datasets of brain scans, they can classify hemorrhages into different

types—lobar, subtentorial, deep, or no hemorrhage—helping in faster and more reliable diagnosis.

However, one of the biggest challenges of deep learning in healthcare is the "black-box" nature of AI models. These models provide predictions but tend to lack interpretability, making it difficult for clinicians to trust their decisions. To address this issue, Grad-CAM has been implemented in this project. Grad-CAM helps visualize which regions of the brain scan influenced the model's prediction, making the AI system more transparent and increasing trust in its suggestions.

1.2 Problem Definition

Traditional methods of diagnosing brain hemorrhages are time-consuming and heavily reliant on radiologists, which can lead to delays in critical treatment and negatively impact patient outcomes.

With the increasing volume of medical imaging data, there is an urgent need for reliable, automated systems that is capable of quickly and accurately identify and classify brain hemorrhages, supporting healthcare professionals in providing timely treatment

1.3 Scope and Motivation

Scope

This project focuses on developing and evaluating an automatic brain hemorrhage detection and its classification using deep learning techniques. The project specifically compares the performance of four CNN-based models—EfficientNet-B0, VGG-16, ResNet-50, and DenseNet-121—on classifying hemorrhages into lobar, subtentorial, deep, or no hemorrhage. The study involves image preprocessing, model training, and performance evaluation based on key metrics such as accuracy, sensitivity, and specificity.

To improve the interpretability of predictions, Grad-CAM is integrated into the system, allowing visualization of important regions in the CT scans that influenced the model's decisions. By comparing the models, this research aims to identify the most effective architecture for brain hemorrhage classification and obtain insights into their strengths and weaknesses in medical image analysis.

Motivation

The need for this research arises from the critical need to have quick and reliable detection of brain hemorrhages. Conventional methods rely on radiologists manually interpreting brain scans, a process that is time-consuming and prone to human errors. With the increasing demand for automated diagnostic tools, deep learning offers promising solutions. However, different CNN architectures perform differently in medical image classification, necessitating a detailed comparison to determine the most suitable model.

Moreover, the "black-box" nature of deep learning models restricts their acceptance in healthcare. By integrating Grad-CAM, this research ensures model predictions are explainable, increasing trust and usability among medical professionals. Ultimately, this study aims to improve patient outcomes by identifying the most efficient and interpretable AI model for brain hemorrhage classification.

1.4 Objectives

1. Implement and Compare CNN Architectures: Train and evaluate four deep learning models—EfficientNet-B0, VGG-16, ResNet-50, and DenseNet-121—on a dataset of brain CT scans to classify hemorrhages into lobar, subtentorial, deep, and no hemorrhage.

2. Assess Model Performance Using Key Metrics: Evaluate the classification performance of each model using metrics such as accuracy, recall, precision and F1-score to determine the most effective architecture.

3. Enhance Explainability Using Grad-CAM: Integrate Grad-CAM to visualize important regions in CT images that influence model predictions, improving transparency and interpretability in medical decision-making.

1.5 Challenges

1. Data Availability and Quality: Obtaining a large, high-quality dataset of brain CT or MRI scans with accurate labels (hemorrhage or non-hemorrhage) can be difficult, which can affect the performance and generalization of the model.

2. Model Overfitting: There is a risk of overfitting, learning noise and irrelevant details, which could lead to poor performance on unseen data.

3. Computational Resources and Time: Deep learning models, particularly CNNs, require substantial computational power and time for training, especially when working with large medical imaging datasets.

1.6 Societal / Industrial Relevance

The development of an automated brain hemorrhage detection system has significant societal and industrial relevance, particularly in the healthcare sector. Brain hemorrhages are life-threatening and require immediate diagnosis, but traditional methods can be slow and resource-intensive, especially in emergency situations or in regions with limited access to experienced radiologists. By providing fast and accurate hemorrhage detection, this system has the potential to save lives and improve patient outcomes by enabling quicker treatment decisions.

Industrially, this project aligns with the growing demand for AI-based solutions in healthcare, as AI has shown promise in reducing diagnostic errors and enhancing workflow efficiency. Implementing automated detection systems in hospitals and diagnostic centers which helps in reducing the workload for practitioners, this allows them to focus on more complex cases and thus increasing the productivity of medical staff. Moreover, with the integration of explainable AI, this project addresses an essential aspect of trust and transparency in AI, making it more likely that such tools will be adopted widely in clinical practice. This technology can also support remote healthcare services, improving diagnostic capabilities in underserved or rural areas and contributing to overall healthcare equity.

1.7 Assumptions

This research assumes that the dataset used for training and evaluation is diverse and representative of real-world brain hemorrhage cases. It is also assumed that the CT images are of sufficient quality for accurate classification by deep learning models. Furthermore, it is expected that the selected CNN architectures—EfficientNet-B0, VGG-16, ResNet-50, and DenseNet-121—are well-suited for medical image analysis. Lastly, the effectiveness of Grad-CAM in providing interpretability is assumed to be sufficient for increased model transparency as well as assisting medical professionals in understanding

AI-based predictions.

1.8 Organization of the Report

The report is structured into several chapters, each focusing on a specific aspect of the project. Chapter 1 introduces the project by discussing its background, problem definition, scope, motivation, objectives, challenges, and societal relevance. Chapter 2 presents a literature survey, reviewing existing works on intracranial hemorrhage classification, deep learning in medical imaging, and explainability techniques. It highlights methodologies, results, and gaps in prior research. Chapter 3 outlines the hardware and software requirements, detailing the computational resources, tools, and frameworks used for model training and deployment. Chapter 4 describes the system architecture, explaining the CNN models—EfficientNet-B0, ResNet-50, DenseNet-121, and VGG-16—along with system flow, module design, and evaluation metrics. Chapter 5 details the implementation process, including dataset preprocessing, model training, and Grad-CAM integration for explainability. It also discusses the deployment strategy. Chapter 6 presents the results, analyzing model performance using accuracy, precision, recall, and F1-score. It includes confusion matrices, Grad-CAM visualizations, and insights into generalization and overfitting. Chapter 7 concludes the report by summarizing key findings, identifying the best-performing model, and discussing potential improvements and future research directions. Additionally, the report includes a list of abbreviations, figures, tables, references, and appendices with supplementary details. This structured approach ensures clarity and logical progression throughout the study.

Chapter 2

Literature Survey

This chapter delves into various research papers and studies that were reviewed for the development of our project. A comprehensive literature review helped to identify existing approaches for brain hemorrhage detection, understand advancements in deep learning and explainable AI (XAI) techniques, and highlight gaps in current methodologies. The insights gathered from these studies guided the design of our model and the selection of techniques best suited for accurate and interpretable hemorrhage detection. The following sections summarize key findings and methodologies from relevant research papers.

2.1 Intracranial Hemorrhage Classification From CT Scan Using Deep Learning And Bayesian Optimization [1]

2.1.1 Introduction

Intracranial hemorrhage (ICH), characterized by bleeding within the skull, is a critical medical emergency requiring immediate diagnosis and intervention. Commonly caused by traumatic brain injury, aneurysms, high blood pressure, or vascular malformations, ICH can lead to severe complications or even death if not promptly and accurately diagnosed. Medical imaging, particularly computed tomography (CT), is the preferred modality for detecting and classifying ICH due to its speed and reliability in visualizing internal bleeding. However, relying solely on human interpretation of CT scans is not only time-consuming but also subject to variability, especially in emergency settings where radiologists may be unavailable or under significant pressure.[1]

Recent developments in deep learning and artificial intelligence have created new opportunities to assist radiologists with precise, automated medical picture analysis. The capacity of deep learning models, especially CNNs, to identify increasingly intricate patterns in image data has made them effective instruments for image classification tasks.

In this regard, classification of ICH with CNNs presents a promising approach toward the automatic detection and differentiation of epidural, subdural, subarachnoid, intraparenchymal, and intraventricular hemorrhages, so improving the speed and consistency of diagnoses. This is important for the betterment of patient outcomes, especially in resource-poor or isolated communities where the availability of radiology expertise may be low.

The challenge of training a CNN model to accurately classify ICH types is to obtain optimal hyperparameter settings, as different settings can drastically affect model performance. Traditional methods of tuning hyperparameters are expensive and may not be effectively the best solution. This report explores the use of BO, A strategy using probability models, to efficiently identify the best suitable hyperparameters for the models. By using Bayesian Optimization, the DL model will be able to achieve higher classification accuracy, reducing training time and computational resources. It allows a intended search for more suitable parameters that balances exploration and exploitation to upgrade performance.

Apart from the technological breakthroughs, this study can further emphasize AI-assisted diagnostics for transforming health care delivery; this will rapidly enable high accuracy in diagnostics accessible to different types of clinical setups. The integrating CNNs with Bayesian optimization aspires to produce a reliable tool that supports and supplements the medical expert's experience, including those of radiologists. Such systems could both enhance accuracy and speed up diagnostics while unloading the health-care provider with much workload, and the patient can receive better care as a result. This project stands as an extraordinary contribution to the domain of computer-aided diagnosis, demonstrating the role of AI in addressing severe medical imaging needs and patient management requirements.

2.1.2 Methodology

CNN is a specialized neural network architecturethat is used to process data with grid-like structures such as images. CNNs are particularly efficient in processing image data because they can adapt and automatically learn spatial orders through convolutional layers. In the context of this project, CNNs are used for classifying different types of intracranial hemorrhages from CT scans. The important feature of CNNs is their extraction capability

in images. The network will be able to identify the pattern of edges, textures, and other important details that might be helpful in identify the type of hemorrhage.

Convolution Operation

At the heart of the CNN architecture is the **convolution operation**, which processes the input image using a tiny matrix of numbers known as a **filter** or **kernel**. This is achieved through sliding the filter across the input image pixel by pixel and performing element-wise multiplications with each pixel in the image. Products are summed up to produce a single output value for that position. They are added together to generate one output value for the position. It is done across the entire image all the way through, which gives a resulting output image known as a **feature map**. Mathematically, the convolution operation on an image is expressed as:

$$X'(i,j) = \sum_m \sum_n X(i+m, j+n)K(m,n) \tag{2.1}$$

This convolution process allows the network to identify features like edges and other patterns that are important for differentiating between ICH types.

Pooling Operation

The next convolutional layer, includes **pooling layers** to downsample and summarize the extracted features. This reduces the size of the feature maps, thereby reducing the load and hence improves model efficiency. Additionally, pooling improves ability of the network to detect features irrespective of minimal positional shift. **Max pooling** and **Average pooling** are the most commonly used operations. Max pooling assigns the highest value within a specified region (e.g., 2x2 pixels) as the output, while average pooling takes the average of all values within the region. In this project, max pooling is used, as it effectively reduces the feature map's dimensions, while retaining the most prominent features. For instance, max pooling on a 4x4 region results in a 2x2 downsampled map by taking the maximum value in each 2x2 sub-region.

Flattening and Fully Connected Layers

Once the feature maps are generated through convolution and pooling layers, they are passed through a **flattening layer** to convert them into a one-dimensional vector. This vector is then led into a **FCN**, where every neuron in a layer is connected to neurons in the next layer. It serves as the classifier by using the features taken out from the convolutional and pooling layers. A visualization of the flattening operation demonstrates how the multi-dimensional feature map is transformed into a single vector that can be processed by the fully connected layers. It consists of many layers. In the last layer, it consists of one node for every class representing possible types of hemorrhage. In this project, there are five nodes that correspond to five subtypes of intracranial hemorrhage as an output in FCN.

2.1.3 Result

The document discusses various ML and DL techniques for detecting brain hemorrhage, emphasizing the use of imaging data, particularly computed tomography (CT) scans. It outlines the main stages of the brain hemorrhage detection process, including data fetching, preprocessing, segmentation, feature extraction, and classification. The preprocessing methods highlighted include windowing, grayscale conversion, and edge detection, which optimize the images for analysis. DL techniques, such CNNs and hybrids like CNN-LSTM, are applied to capture complex features for accurate diagnosis. The paper also reviews numerous algorithms, discussing their main contributions, limitations, and optimization parameters. A general framework for brain hemorrhage detection is presented, which integrates AI approaches with a pipeline for improved prediction and classification. Finally, the document compares the performance of different models, datasets, and parameters, giving insights into the advantages and disadvantages of each method, and suggests future research directions in enhancing brain hemorrhage detection systems.

2.2 Visualizing Deep Network Outputs via Gradient-Based Localization[2]

2.2.1 Introduction

CNN and other DL models achieved great advancements across different computer vision tasks. Despite their success, these models often lack interpretability due to their complex,

layered structures, making it difficult to know how and why they make specific predictions. As a result, when these systems fail, they may do so in unexpected and unexplainable ways, leaving users without insights into the underlying causes. [2]

Interpretability is crucial for building sureness in AI systems and for the relevent implementation in our daily lives. Transparent models, which can elucidate their decisions, benefit AI development in three main ways: identifying failure modes during the early stages of AI, establishing user trust when AI is reliable enough for deployment, and providing insights when AI surpasses human performance in certain tasks, like chess or Go. Traditional rule-based systems offer interpretability but often fall short in accuracy compared to deep models. On the other hand, deep models sacrifice interpretability for higher performance, interchangeing between accuracy and simplicity.

This is addressed by Grad-CAM, a technique that creates visual representations for CNN models without necessitating retraining or architectural modifications. It highlights areas of the image that are crucial for class prediction by employing gradients to produce a coarse localization map. This approach is broadly applicable to CNNs across different tasks and enables high-resolution, class-discriminative visualizations when combined with other pixel-space gradient techniques, forming Guided Grad-CAM. This combination allows for more interpretable and reliable AI systems, helping users to better understand model decisions and establish appropriate levels of trust.

2.2.2 Methodology

The methodology suggested in this paper is the Grad-CAM, which creates visual explanations for decisions taken by CNN-based models.This is achieved through Grad-CAM, which generates a localization map using gradients flowing into the last convolutional layer. The regions of an image are highlighted, indicating what parts the model uses for its prediction. Unlike the earlier approaches, Grad-CAM can be widely applied to a variety of CNN architectures without having to make modifications or even retrain them. This is inclusive of those with fully connected layers and other models used for tasks such as captioning, visual question answering, and reinforcement learning.

$$L^c_{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k \alpha^c_k A^k\right) \tag{2.2}$$

We first calculate the gradient of the class score (before to the softmax) in relation to the feature maps of a convolutional layer in order to produce the localization map. These gradients flowing back are subsequently global-average-pooled to obtain the neuron significance weights.

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y_c}{\partial A_{k,ij}} \tag{2.3}$$

Advantages

Class-Discriminative Localization: Grad-CAM focuses on atghe areas in the image that are important for predicting a specific class, helping to visualize where the model is "looking" for its decision. This makes it class-discriminative.

Efficient Computation: Grad-CAM requires only a single forward pass and a partial backward pass through the model, making it computationally efficient compared to methods that need multiple passes or substantial perturbation.

Disadvantages

Coarse Localization: The heatmaps generated by Grad-CAM are often coarse and may not capture fine-grained details. For instance, it may highlight broad areas rather than specific parts of an object.

Dependence on Convolutional Layers: Grad-CAM is limited to models with convolutional layers, making it less applicable to architectures without convolution, such as some types of transformers or dense networks.

2.2.3 Results

To check whether Grad-CAM aids in classification by distinguishing classes, images from the VOC 2007 validation set with exactly two annotated categories were selected and visualizations were generated for each. Category-based visualizations were created using both VGG-16 and AlexNet CNNs, four methods being Deconvolution, Guided Backpropagation, and two versions of Grad-CAM: Deconvolution Grad-CAM and Guided Grad-CAM. Interestingly, results are showing that Deconvolution is more class discriminative than Guided Backpropagation, though it generates images, which are much more visually attractive. To our knowledge, no previous study exists that quantifies such differences in subtle classes between the different methods.

2.3 Interpretable Deep Learning Approaches for Medical Image Analysis [3]

2.3.1 Introduction

CAD using AI presents a assuring solution for enhancing the efficiency and accessibility of medical diagnoses, particularly through the analysis of medical imaging. DL, a powerful subdivision of AI, has demonstrated exceptional performance in functions such as lung cancer detection, and retinal disease identification. However, despite these advancements, the implementation of AI into clinical settings has been limited because of the black-box nature of DL models. Regardless of their statistical strength, these models are opaque, which makes it challenging for patients, regulators, and doctors to understand the rationale behind the model's predictions. [3]

This opacity reduces trust and raises ethical distress. XAI seeks to address these challenges in addressing methods that can explain the decision-making process of complex models. XAI techniques, such as attribution-based methods like Grad-CAM, Score-CAM, and Increment-CAM, offer insights into the features of an image or data point that influence a model's decision, making the model's behavior more transparent and interpretable. This explainability is essential not only for gaining the trust of medical professionals but also for ensuring ethical and reliable AI applications in critical fields like healthcare, where understanding the model's reasoning is vital for informed decision-making.

This paper provides studies that deals with explainability of DL models in the medical imaging fields.

2.3.2 Classification of Explainability Approaches

Model Specific and Model Agnostic: While model-agnostic approaches can be applied to any model, the major focus is still on post-hoc analysis without accessing the internal parameters of the model, whereas model-specific interpretation methods are specialised to a particular model, meaning they are suited to its internal workings.

Global Methods and Local Methods: The intention of global methods focuses to depict the behavior of the model by analyzing the whole model such as through feature importance to determine which features generally contribute to the model's performance whereas Local interpretable methods focus on explaining a single prediction by identifying specific

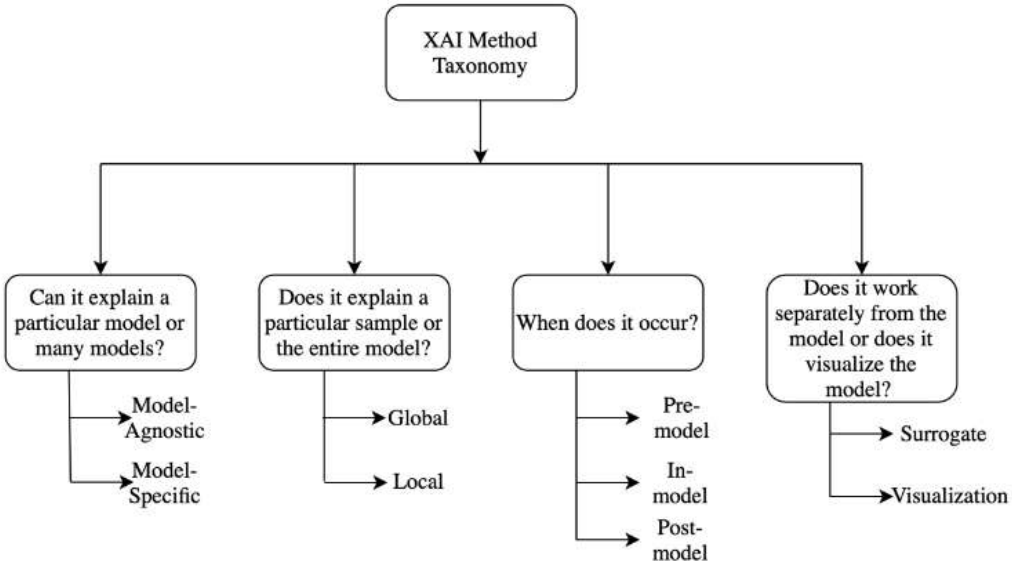


Figure 2.1: Classification of XAI methods

features and their influence on that outcome

Pre-Model, In-Model and Post-Model: Here, Pre-model methods are used for dimensionality reduction or visualizing data. Post-model methods, applied after model development, help interpret and understand what the model has learned during training, offering valuable insights into its decision-making process.

Surrogate Methods vs. Visualization Methods: Surrogate methods involve using simpler, interpretable models, like decision trees, to approximate and analyze complex black-box models by comparing their decisions. Visualization methods, on the other hand, do not involve separate models but help explain parts of a model through visual representations, such as activation maps, to enhance understanding of the model's behavior.

2.4 Integrated Deep Learning System for Intracranial Hemorrhage Detection[4]

2.4.1 Introduction

Intracranial hemorrhage (ICH) is a critical medical condition involving bleeding within the skull, which poses a significant risk to patients' health and often requires immediate diagnosis and treatment. The complexity of accurately identifying ICH from cranial CT scans makes this process both time-consuming and challenging for medical professionals.Delays in diagnosis or incorrect diagnoses can have serious consequences, such as

death or irreversible brain damage.[4]

Recently, deep learning and artificial intelligence advancements have opened the way for innovative solutions to medical imaging challenges. Neural network-based models, particularly CNNs, have showcased remarkable success in automating segmentation and image classification tasks. However, most existing models face limitations in handling complex, multi-label classifications required for identifying different types of ICH.

This project proposes a hybrid DL framework that implements CNNs with LSTM networks. The proposed approach leverages systematic windowing techniques and spatiotemporal data extraction to enhance the detection and classification of six types of intracranial hemorrhage. This framework is based on a dataset obtained from the RSNA competition aims to achieve high accuracy, sensitivity, and specificity, outperforming traditional methods. By automating and improving the accuracy of ICH detection, this work not only aids radiologists in making prompt decisions but also contributes to advancing AI applications in critical healthcare situations.

2.4.2 Proposed Methodology

The proposed methodology suggests a hybrid DL model that involves the combination of CNNs and LSTM networks. In this approach, systematic windowing of CT scans and spatiotemporal feature extraction are used for improved ICH detection and its classification.

2.4.3 Systematic Windowing Technique

The systematic windowing technique mimics the practice of radiologists viewing CT scans under multiple window settings to enhance the visibility of different types of hemorrhages. The pixel value transformations are defined as follows:

$$\text{Min} = \text{Window Center} - \frac{\text{Window Width}}{2} \tag{2.4}$$

$$\text{Max} = \text{Window Center} + \frac{\text{Window Width}}{2} \tag{2.5}$$

$$\text{New Pixel Value} = \begin{cases} \text{Min}, & \text{if Pixel Value} < \text{Min} \\ \text{Max}, & \text{if Pixel Value} > \text{Max} \\ \text{Pixel Value}, & \text{otherwise} \end{cases} \tag{2.6}$$

The pixel values are then normalized to the range $[0, 1]$ as:

$$\text{Normalized Value} = \frac{\text{Pixel Value} - \text{Min}}{\text{Max} - \text{Min}} \tag{2.7}$$

2.4.4 Feature Extraction with CNN

CNN extracts spatial features from the sequence of windowed images. The architecture includes convolutional, max-pooling, and dense layers. The activation function is:

$$\text{ReLU}(x) = \begin{cases} 0, & \text{if } x \leq 0 \\ x, & \text{if } x > 0 \end{cases} \tag{2.8}$$

The output layer uses a sigmoid activation function to compute probabilities:

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{2.9}$$

2.4.5 Sequence Modeling with LSTM

The features extracted by the CNN are passed to the LSTM network, which captures temporal dependencies across the sequence of windowed images. LSTM networks are best for handling long-term dependencies, which makes them suitable for processing medical data.

2.4.6 Loss Function

Binary cross-entropy is the loss function used to handle the multi-label classification problem. The loss is described as follows for each label:

$$\text{Loss} = -\left(y \cdot \log(p) + (1 - y) \cdot \log(1 - p)\right) \tag{2.10}$$

here y is the ground truth value, and p is the predicted probability.

The total loss across all six labels is computed as:

$$\text{Total Loss} = \frac{1}{6} \sum_{i=1}^{6} \text{Loss}_i \tag{2.11}$$

2.4.7 Optimization

Adam optimizer was used to optimize the network. The parameters of the equation are updated according to the optimizer's equation.

This hybrid model of Conv-LSTM would assimilate the spatial and temporal information from the CT scan, hence producing robust and accurate predictions of all types of intracranial hemorrhages. The systematic windowing technique increases the visibility of ICH features, thereby further enhancing the outperformance of this model as against traditional methods.

2.5 Summary

The literature review captures four primary papers in ICH detection and state-of-the-art DL algorithms. In this paper, ICH is found to be easily detectable through CNN from the CT scans and the need for preprocessing techniques like normalization and windowing for robust feature extraction and high accuracy. The second paper introduces Grad-CAM, a visualization tool that generates heatmaps to identify regions influencing model predictions, thereby increasing trust and interpretability in medical AI. The third paper reviews XAI methods, including saliency maps and attention mechanisms, which improve clinician trust by clarifying decision-making processes and mitigating biases. The fourth paper is a hybrid Conv-LSTM model that combines CNNs with LSTM networks to classify six types of ICH using spatial and temporal data, achieving superior accuracy and sensitivity. Together, these works underscore the importance of preprocessing, interpretability, and improving reliable AI-driven medical imaging.

Chapter 3

Requirements

3.1 Hardware Requirements

Processor: An Intel Core i7 or AMD Ryzen 7 (or higher) processor is preferred to handle data pre-processing and model-related computations.
GPU (Graphics Processing Unit): For efficient training and inference, an NVIDIA GPU such as the RTX series is recommended. These GPUs significantly accelerate DL tasks.
RAM: At-least 16 GB RAM ensure smooth execution, especially when working with large datasets and deep learning frameworks.

3.2 Software Requirements

Operating System:
IDE: Visual Studio Code
TensorFlow: Used for building, training, and evaluating the CNN model.
Keras: A high-level API within TensorFlow which streamlines the definition and training of CNN architectures.
NumPy: Facilitates essential numerical operations for matrix manipulations and computations.
Pandas: Useful for data manipulation and data management.
OpenCV: Handles image pre-processing tasks such as resizing, grayscale conversion, and data augmentation.
Grad-CAM Library: TensorFlow-compatible libraries like tf-keras-vis enable Grad-CAM implementations for model explainability.

Chapter 4

System Architecture

4.1 System Overview

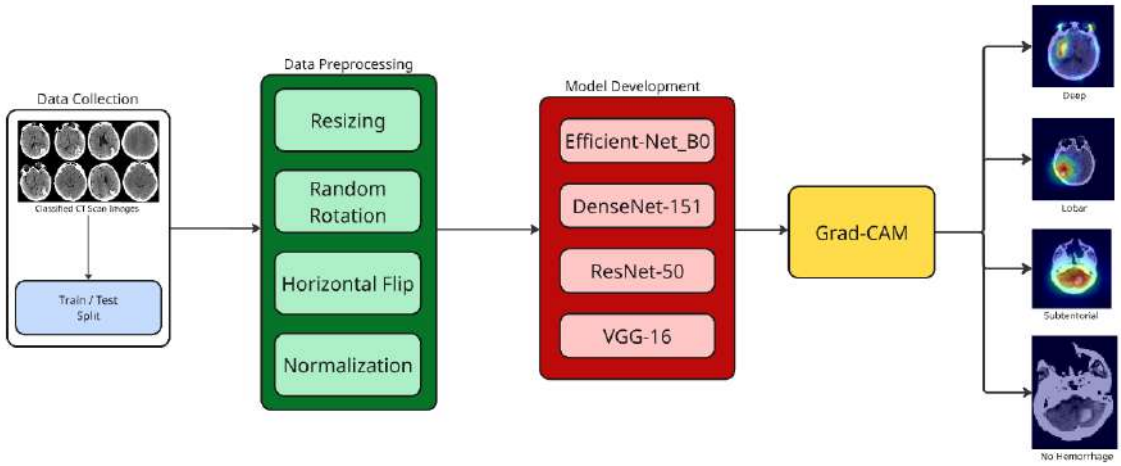


Figure 4.1: Architecture diagram of HemorAI

The brain hemorrhage detection and classification system architecture involves four main stages: Data Collection, Data Preprocessing, Model Development, and Explainability using Grad-CAM. The dataset comprises classified CT scan images, which are split into training, testing, and validation subsets for proper model evaluation. Preprocessing techniques such as resizing, random rotation, horizontal flipping, and normalization are applied to standardize input data and enhance model generalization. The model development phase involves training four different CNN architectures—EfficientNet-B0, DenseNet-121, ResNet-50, and VGG-16—to classify hemorrhages into four categories: Deep, Lobar, Subtentorial, and No Hemorrhage. To improve transparency and interpretability, Grad-CAM is used to create visual explanations by highlighting the regions

of CT scans that contributed to the model's decision. Additionally, the system includes a user-friendly UI where users can upload a CT scan image and receive predictions from all four models. The UI displays an overlayed image with Grad-CAM visualizations, highlighting the hemorrhage-affected regions and classifying the type of hemorrhage based on model predictions. This architecture provides an effective, explainable, and accessible AI-driven solution, assisting healthcare professionals in making faster and more accurate diagnoses.

4.2 DenseNet-121

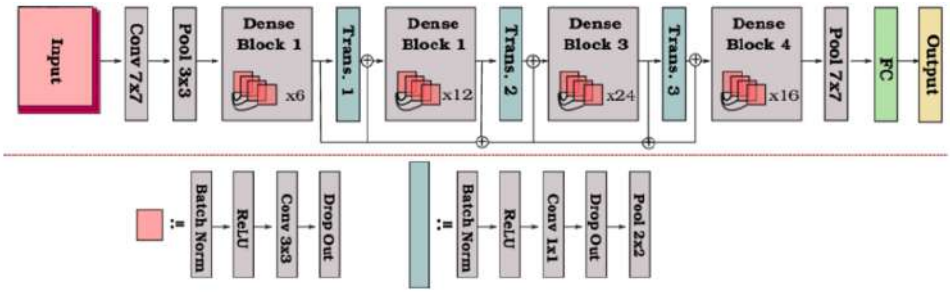


Figure 4.2: DenseNet-121 Architecture Diagram [5]

DenseNet is a deep learning model that enhances feature propagation and minimizes redundancy by introducing dense connections between layers. Unlike traditional CNNs, where each layer is independent, DenseNet ensures that each layer gets inputs from all the previous layers, improving gradient flow and encouraging feature reuse. In fig 4.2, the network begins with a 7×7 convolution and 3×3 pooling, followed by multiple Dense Blocks separated by Transition Layers. Each Dense Block contains several convolutional layers, in which feature maps are propagated to all following layers, thus reducing the need for too many parameters. The Transition Layers, consisting of 1×1 convolutions and 2×2 pooling, help regulate feature growth and reduce spatial dimensions. Finally, a global average pooling layer and a fully connected layer produce the final classification output. This connectivity pattern enables efficient learning, mitigates vanishing gradients, and achieves better performance with fewer parameters compared to regular deep networks.

4.3 ResNet-50

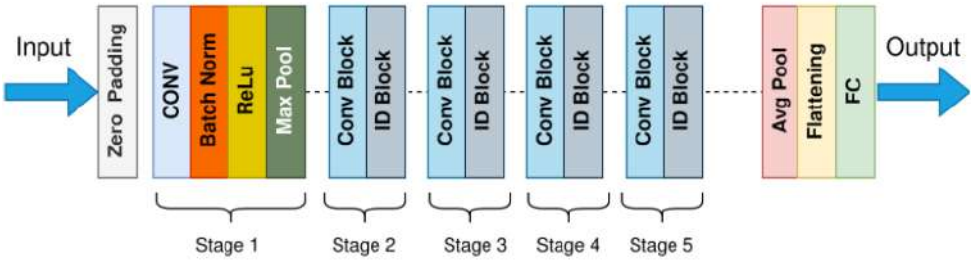


Figure 4.3: ResNet-50 Architecture Diagram [6]

ResNet-50 (Residual Network-50) is a deep CNN architecture network aimed at mitigating the vanishing gradient problem by introducing residual connections or skip connections. These connections enable the network to learn residual mappings instead of direct mappings, enabling deeper architectures without degradation in performance. As shown in fig 4.3, ResNet-50 begins with an initial conv layer followed by batch normalization, ReLU activation, and max pooling. The network consists of five stages, each containing multiple Convolutional (Conv) Blocks and Identity (ID) Blocks. The Conv Blocks adds new feature maps, while the ID Blocks maintain the dimensions and allow deeper feature extraction. The network ends with average pooling, flattening, and a fully connected layer for classification. The skip connections allow the network to maintain critical low-level information while enabling efficient gradient propagation, making ResNet-50 very effective in extracting hierarchical features from images.

4.4 EfficientNet-B0

EfficientNet-B0 is a lightweight and computationally efficient convolutional neural network for obtaining optimal performance with fewer parameters and lower computational costs. It is part of the EfficientNet family, which scales network dimensions—width, depth, and resolution—are scaled with a compound scaling method to balance efficiency and accuracy. From fig 4.4, EfficientNet-B0 begins with a standard 3×3 convolutional layer followed by mobile inverted bottleneck convolution (MBConv) layers, which improves efficiency by utilizing depthwise separable convolutions and squeeze-and-excitation mechanisms. The network has several blocks, each containing different MBConv layers with varying kernel

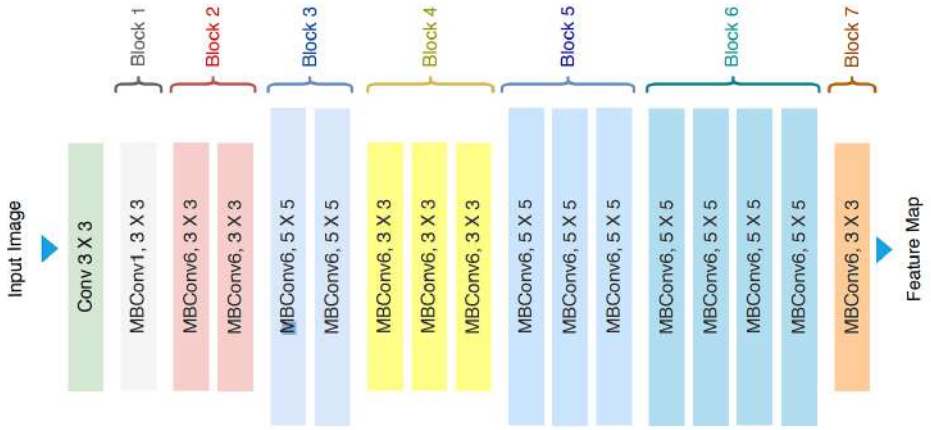


Figure 4.4: EfficientNet-B0 Architecture Diagram[7]

sizes (3×3 and 5×5) to capture both local and global features. The final layers generate a feature map, which can be further processed for classification tasks. Due to its optimized architecture, EfficientNet-B0 is able to produce better accuracy using fewer resources than traditional CNNs and is thus suited for real-time medical imaging applications like brain hemorrhage detection.

4.5 VGG-16

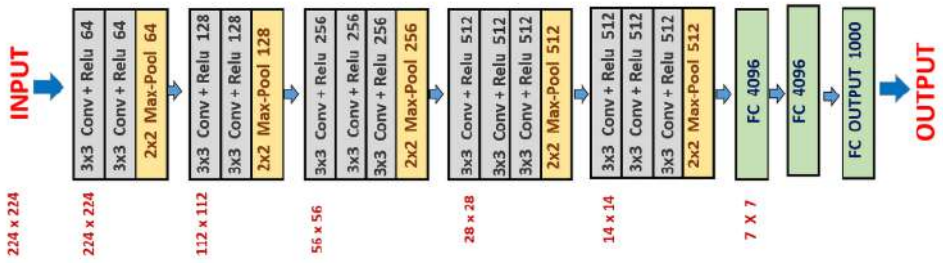


Figure 4.5: VGG-16 Architectural Diagram[8]

VGG-16 is a deep convolutional neural network architecture that is simple and efficient for image classification tasks. It is characterized by an extremely simple design of sequential 3×3 convolutional layers stacked with ReLU activation and max-pooling layers to progressively downsample the spatial dimensions while extracting hierarchical features. VGG-16 processes an input image of size 224×224 through multiple convolutional layers, radually adding the number of filters at each level (from 64 to 512). The max-pooling lay-

ers (2×2) serve to downsample the feature maps, retaining key spatial features but with less computational expense. The feature-extracted information is then processed through fully connected layers and then a final classification layer. VGG-16 is commonly used for transfer learning and medical imaging applications due to its well-structured feature extraction capabilities.

4.6 Sequence Diagram

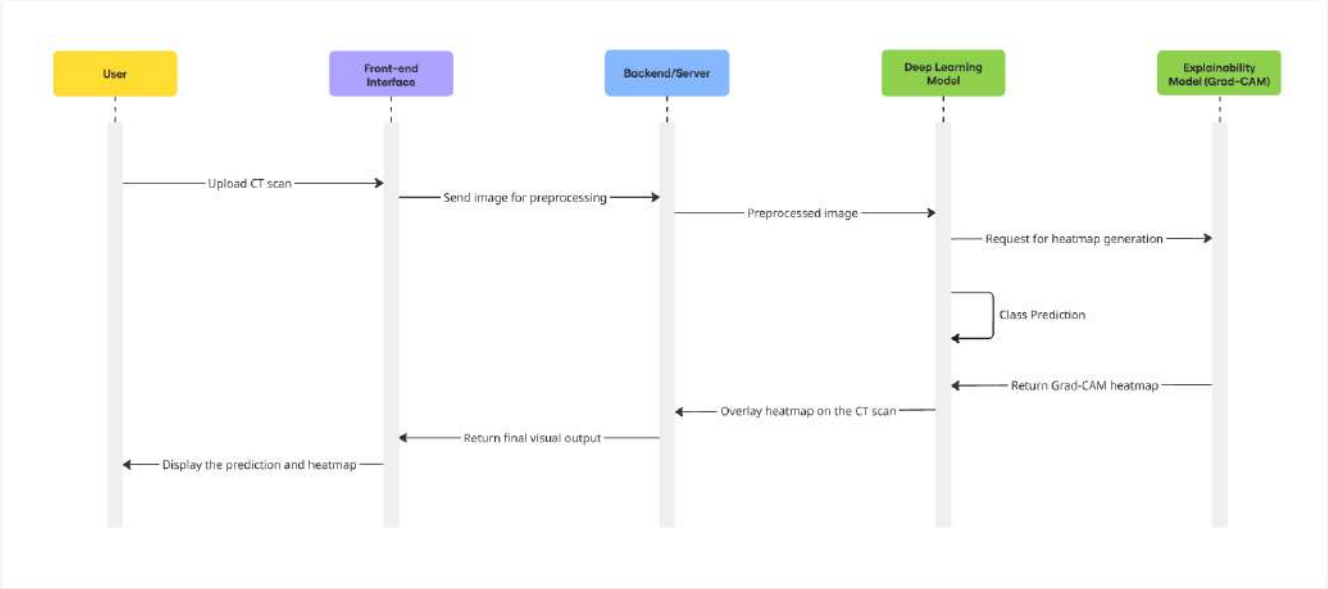


Figure 4.6: Sequence Diagram of HemorAI

4.7 Performance Metrics

Four common evaluation measures are used for classifier performance assessment: **Accuracy, Precision, Recall, and F1-score**. These metrics are derived from the confusion matrix components:

True Positive (TP): A hemorrhage-positive CT scan correctly classified as hemorrhage-positive. **True Negative (TN)**: A non-hemorrhage CT scan correctly classified as non-hemorrhage.

False Positive (FP): A non-hemorrhage CT scan incorrectly classified as hemorrhage-positive.

False Negative (FN): A hemorrhage-positive CT scan incorrectly classified as non-

hemorrhage.

Precision measures the accuracy of positive predictions and is defined as:

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

Recall quantifies the ability of the model to correctly identify positive cases:

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

F1-score is the harmonic mean of precision and recall, balancing both metrics to provide a single performance measure:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4.3}$$

Accuracy measures the overall accuracy of the model's predictions:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.4}$$

This research-based project compares four CNN architectures—EfficientNet-B0, VGG-16, ResNet-50, and DenseNet-121—for brain hemorrhage detection in CT scans. The system includes *data preprocessing, model training, performance evaluation, explainability using Grad-CAM, and a user interface.* Users can upload a CT scan, and the system classifies the hemorrhage type into *Deep, Lobar, Subtentorial, or No Hemorrhage.* Grad-CAM overlays heatmaps to highlight the affected region, enhancing interpretability. Each model's output is displayed for comparison. The project's effectiveness relies on rigorous evaluation, fine-tuning, and user feedback, with future improvements focusing on dataset expansion and UI refinement.

Chapter 5

System Implementation

5.1 Implementation Details

The Brain hemorrhage classification system involves multiple steps like dataset preprocessing, deep learning model building, training, and evaluation. For the comparative study, we use state-of-the-art models such as EfficientNet-B0, DenseNet-121, ResNet-50, and VGG-16. The implementation involves preprocessing of CT scans using normalization and data augmentation methods, followed by model optimization with transfer learning methods. To offer interpretability of the classification output, we employ Grad-CAM, which produces visual explanations by identifying the important regions in input images that play the most crucial role in making the model predict. This explainability feature enables clinicians to check if the model is paying attention to anatomically meaningful areas when identifying hemorrhages.

All the models were coded in PyTorch and trained on an NVIDIA RTX 3050 GPU with CUDA acceleration. The normalized input CT scans were trained in batch size 32 using Adam optimization with learning rate scheduling and early stopping for preventing overfitting. For this multi-class classification problem, we employed Categorical Cross-Entropy Loss with softmax activation, iterating through various learning rates and epochs to maximize each architecture's performance.

5.2 Proposed Methodology/Algorithms

5.2.1 Dataset Preprocessing

The four hemorrhage class dataset (Non-hemorrhagic, Lobar, Subtentorial, and Deep hemorrhage) was manually split into training (80%) and test (20%) sets before processing. All the CT scans were processed using a uniform transformation pipeline: resized to $128 \times$

128 pixels resolution, augmented by random rotations (up to 10 degrees) and horizontal flips, and normalization. To ensure evaluation integrity, we shuffled only the training batches (size=32) and left validation and test sets unchanged. The class distributions were maintained in all splits to avoid data.

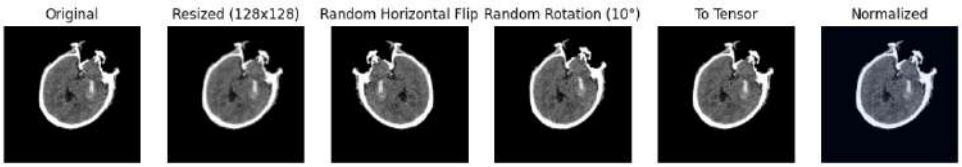


Figure 5.1: Preprocessing Pipeline for Brain Hemorrhage Classification

5.2.2 Model Development, Training and Evaluation

DenseNet-121

DenseNet-121 model with about 8 million trainable parameters (7,978,856) was used, the last being dependent on input sizes (128×128) and our four-class output. The training process started with a preliminary exploration of 20 epochs at several learning rates (10^{-3} to 10^{-5}). With validation performance trend observations, we increased training to 30 epochs with finer learning rate adjustment. We systematically evaluated hyperparameter combinations, tracking how batch size (32), optimizer selection (Adam vs. SGD), and regularization (dropout = 0.5, weight decay = 10^{-3}) affected both convergence speed and final performance. The framework preserved PyTorch's original DenseNet-121 architecture while customizing the classifier head with a 128-unit hidden layer and ReLU activation.

The training schedule utilized class-weighted cross-entropy loss to counter dataset imbalance, where weights are inversely proportional to class frequencies. The validation set was used to provide early stopping criteria and learning rate adjustment signals. Our experiments showed that the model improved with longer training (30 epochs) at reduced learning rates, with the last configuration showing strong performance on all hemorrhage classes. Training utilized CUDA acceleration on NVIDIA GPUs, with batch processing optimized using PyTorch's DataLoader implementation (shuffling enabled for training, disabled for validation). Performance metrics were monitored per epoch, including per-class precision and recall to ensure clinically relevant sensitivity.

Resnet-50

The ResNet-50 architecture was used with transfer learning from ImageNet weights, containing approximately 25 million trainable parameters. The original architecture was changed by replacing the final fully-connected layer with a custom classifier block (128-unit hidden layer with ReLU activation, and 4-class output).The model was trained using class-weighted cross-entropy loss to handle dataset imbalance, with weights being inversely proportional to class frequencies.

Training employed an 80-20 train-validation split with batch size 32, using Adam optimizer (initial LR = 0.00001, weight decay = 10^{-3}). We conducted systematic hyperparameter searches across learning rates (10^{-3} to 10^{-5}) and epochs (20-30), with the optimal validation accuracy achieved at 20 epochs. The implementation used PyTorch's built-in ResNet-50 architecture with CUDA acceleration, but added gradient clipping to stabilize training. Performance was monitored via per-class measures and confusion matrices, with training curves plotted to observe convergence. The end model exhibited well-balanced performance across all hemorrhage classes without compromising computational efficiency via optimized batch execution.

VGG-16

The deep feature extraction VGG-16 model was also utilized in this research for the classification of brain hemorrhage. The model adopted a pre-trained VGG-16 model and modified the fully connected layer to suit the four hemorrhage classes. A batch size of 32 was used with categorical cross-entropy loss and the Adam optimizer for updating weights. Various combinations of epochs (20 and 30) and learning rates (0.0001 and 0.001) were experimented to see the best possible combination. It was found that the highest unseen data performance resulted when a learning rate of 0.001 was used with 20 epochs, with an accuracy of validation of 95.65%. A reduced learning rate of 0.0001 gave slightly better validation accuracy (96.99%), but additional training to 30 epochs did not improve too much, suggesting that 20 epochs was enough.

Metrics of evaluation like trends in accuracy, loss plots, and confusion matrices all validated that VGG-16 performed incredibly well and possessed strong generalization ability. The model was able to classify images from unseen data sets with success, further

proving its viability for real-world applications of brain hemorrhage classification tasks. Future work may include fine-tuning lower layers or testing other augmentation techniques to further boost performance.

EfficientNet-B0

The EfficientNet-B0 model was utilized with PyTorch and pre-trained ImageNet weights. The classifier was adjusted to add a fully connected layer of 128 neurons, ReLU activation, dropout for regularization, and a final softmax layer for multi-class classification. Input CT scan images were resized to 224×224 pixels, augmented with random horizontal flip and rotation, and normalized. The dataset was loaded by torchvision.datasets.ImageFolder, divided into an 80-20 training-validation split, and preprocessed with DataLoader. Class imbalance was addressed through weighted categorical cross-entropy loss.

Training was conducted on an NVIDIA RTX 3050 GPU with the Adam optimizer and a batch size of 32. Different combinations of learning rate and epochs were experimented with 0.0001 producing the highest validation accuracy of 97.63%. Training performance was tracked with accuracy/loss curves, confusion matrices, and classification reports. The model was saved and loaded for inference, and Grad-CAM was employed to produce visual heatmaps, confirming its attention to clinically relevant areas.

Grad-CAM for Explainability

Grad-CAM was put into effect by initially obtaining gradients of the target class with regard to the final convolutional layer of every deep learning model. The gradients were averaged over the space to calculate importance weights, which were then multiplied with the activation maps of the identical layer. This weighted sum yielded a class-specific localization map, indicating the most significant areas within the input image. The resulting heatmap was normalized and upsampled to the same size as the original CT scan, and overlaid on top of the input image for providing a visual account of the model's decision-making process. The whole implementation was done through PyTorch, wherein hooks were registered to capture intermediate activations as well as gradients such that data needed to be extracted during inference efficiently.

To apply Grad-CAM to our trained models, we created a custom function that dynamically obtained feature maps and gradients. The function applied each image to the model, calculated backpropagation to get gradients, and produced heatmaps using OpenCV for visualization. The implementation was capable of multi-class classification by iterating over each class output and creating corresponding heatmaps. We further optimized the process by taking advantage of CUDA acceleration, decreasing computation time for large-scale inference. The ultimate visualizations, saved as superimposed images, gave intelligible information regarding model predictions, which helped validate model reliability and increase its usefulness for clinical applications.

5.2.3 System Deployment

A prototype Flask-based web application was developed to enable users to Upload CT scan images and receive automated classification results from all four models ,View Grad-CAM visualizations, highlighting critical hemorrhagic regions.There is a page displaying description on the 4 models used and also displaying Accuracy and Loss Trend ,Confusion Matrix of all four trained models. The system was implemented using a React-based frontend and a Flask backend. The web interface allows users to upload CT scan images, which are processed in the backend by each of the four deep learning models (DenseNet-121, ResNet-50, VGG-16, and a custom CNN). The classification results from each model are displayed alongside heatmap-overlaid images, generated using Grad-CAM to highlight the key regions influencing predictions. The frontend is designed to provide an intuitive user experience, with a upload feature and real-time classification results. The Flask backend handles image preprocessing, model inference, and Grad-CAM generation. The system enhances interpretability and usability, making it a potential tool for clinical decision support in brain hemorrhage diagnosis.

5.3 Description of Implementation Strategies

This outlines the strategies used to implement the brain hemorrhage classification system. The implementation includes dataset preprocessing, deep learning model training, and Grad-CAM visualization to enhance model interpretability.

Required Packages and Modeules

To implement the brain hemorrhage classification system, we use the following libraries:

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, random_split
from torchvision import datasets, transforms, models
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
import cv2
import os
from PIL import Image
import torchvision.transforms.functional as F
```

Figure 5.2: Required Libraries for Implementation

Dataset Organisation and Preprocessing

Before training the models, the original dataset, which contained *4296 No Hemorrhage, 6093 Deep Hemorrhage, 1656 Lobar Hemorrhage, and 495 Subtentorial Hemorrhage* images, was split into training (80%) and testing (20%) sets within each class. These training datasets underwent preprocessing steps to ensure uniformity and improve model performance.

To optimize model performance, preprocessing steps included *resizing, normalization, data augmentation, and dataset splitting.* The following code demonstrates these preprocessing steps:

```
# Define path to the dataset
data_dir = "path_to_dataset"

# Image transformations
transform = transforms.Compose([
    transforms.Resize((128, 128)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

# Load dataset
full_data = datasets.ImageFolder(root=data_dir, transform=transform)
num_classes = len(full_data.classes)

# Split into training and validation sets (80-20 split)
train_size = int(0.8 * len(full_data))
val_size = len(full_data) - train_size
train_data, val_data = random_split(full_data, [train_size, val_size])

# Data Loaders
train_loader = DataLoader(train_data, batch_size=32, shuffle=True)
val_loader = DataLoader(val_data, batch_size=32, shuffle=False)
```

Figure 5.3: Dataset Preprocessing and Splitting

5.3.1 Model Architectures

We implement and compare *DenseNet-121, ResNet-50, VGG-16*, and *EfficientNet-B0* for brain hemorrhage classification.

(a) DenseNet Model

```
class DenseNetMultiClass(nn.Module):
    def __init__(self, num_classes):
        super(DenseNetMultiClass, self).__init__()
        self.densenet = models.densenet121(weights=models.
   DenseNet121_Weights.IMAGENET1K_V1)
        num_ftrs = self.densenet.classifier.in_features
        self.densenet.classifier = nn.Sequential(
            nn.Linear(num_ftrs, 128),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(128, num_classes)
        )

    def forward(self, x):
        return self.densenet(x)
```

Figure 5.4: DenseNet-121 Model

(b) ResNet Model

```
class ResNetMultiClass(nn.Module):
    def __init__(self, num_classes):
        super(ResNetMultiClass, self).__init__()
        self.resnet = models.resnet50(weights=ResNet50_Weights.
   IMAGENET1K_V1)
        # Modify the fully connected layer
        num_ftrs = self.resnet.fc.in_features
        self.resnet.fc = nn.Sequential(
            nn.Linear(num_ftrs, 128),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(128, num_classes)
        )

    def forward(self, x):
        return self.resnet(x)
```

Figure 5.5: ResNet-50 Model

(c) VGG16 Model

```
class ResNetMultiClass(nn.Module):
    def __init__(self, num_classes):
        super(ResNetMultiClass, self).__init__()
        self.resnet = models.resnet50(weights=ResNet50_Weights.
   IMAGENET1K_V1)
        # Modify the fully connected layer
        num_ftrs = self.resnet.fc.in_features
        self.resnet.fc = nn.Sequential(
            nn.Linear(num_ftrs, 128),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(128, num_classes)
        )

    def forward(self, x):
        return self.resnet(x)
```

Figure 5.6: VGG-16 Model

(d) EfficientNet Model

```
class EfficientNetMultiClass(nn.Module):
    def __init__(self, num_classes):
        super(EfficientNetMultiClass, self).__init__()
        self.efficientnet = models.efficientnet_b0(weights=
   EfficientNet_B0_Weights.IMAGENET1K_V1)
        # Modify the classifier layer
        num_ftrs = self.efficientnet.classifier[1].in_features
        self.efficientnet.classifier = nn.Sequential(
            nn.Linear(num_ftrs, 128),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(128, num_classes)
        )

    def forward(self, x):
        return self.efficientnet(x)
```

Figure 5.7: EfficientNet-B0 Model

5.3.2 Model Training

All models utilize *CrossEntropyLoss* with class weights to handle class imbalance and the *Adam* optimizer for training.

```
# Compute class weights
class_counts = np.bincount([label for _, label in full_data.samples])
total_samples = sum(class_counts)
class_weights = torch.tensor(total_samples / class_counts, dtype=torch.
    float).to(device)

# Define loss function with class weights
criterion = nn.CrossEntropyLoss(weight=class_weights)

# Define optimizer
optimizer = optim.Adam(model.parameters(), lr=0.00001, weight_decay=1e
    -3)
```

Figure 5.8: Loss Function and Optimizer

5.3.3 Explainability Using Grad-CAM

To generate class activation maps, we implemented Grad-CAM, which highlights important regions in the input image that contributed to the model's prediction.

```
def generate_gradcam(model, image_tensor, target_layer):
    model.eval()
    image_tensor.requires_grad = True

    # Forward pass
    output = model(image_tensor)
    target_class = output.argmax(dim=1)

    # Backpropagate to compute gradients
    model.zero_grad()
    output[0, target_class].backward()

    # Extract gradients and activations
    gradients = target_layer.weight.grad
    activations = target_layer.output

    # Compute weighted activations
    alpha = torch.mean(gradients, dim=[2, 3], keepdim=True)
    heatmap = torch.relu((alpha * activations).sum(dim=1)).squeeze()

    return heatmap.cpu().detach().numpy()
```

Figure 5.9: Grad-CAM Implementation

5.3.4 Model Evaluation

```
all_preds = []
all_labels = []

with torch.no_grad():
    for images, labels in val_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        _, preds = torch.max(outputs, 1)
        all_preds.extend(preds.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())

cm = confusion_matrix(all_labels, all_preds)
class_names = full_data.classes

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=class_names, yticklabels=class_names)

plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

print(classification_report(all_labels, all_preds, target_names=
    class_names))
```

Figure 5.10: Evaluation: Confusion Matrix Classification Report

5.4 Conclusion

This implementation compares DenseNet, ResNet, VGG16, and Custom CNN for brain hemorrhage classification, integrates Grad-CAM for explainability, and evaluates the model using a confusion matrix and classification report.

Chapter 6

Results and Discussions

In this section, we present the results of the comparative analysis of four deep learning models— EfficientNet, ResNet, DenseNet, and VGG-16—applied to the task of brain hemorrhage classification. Our research primarily focused on evaluating and comparing the accuracy and effectiveness of these models.

Each model was trained using various hyperparameter combinations to determine the optimal configuration. After extensive experimentation, we identified the best-performing model among them.A detailed analysis of the performance of each model is presented in this section. The models were compared based on key evaluation metrics including accuracy, precision, recall, and F1-score. These metrics provide a comprehensive understanding of each model's strengths and weaknesses in classifying brain hemorrhage cases. By analyzing these performance indicators, we were able to determine which model offered the best balance between sensitivity and specificity for this medical imaging task.

Since deep learning models often function as black-box systems, we integrated Grad-CAM to enhance interpretability. This visualization technique provides insights into the model's reasoning for a given prediction, helping as well as bringing forward the essential regions within the input CT scans, which shaped the decision.

6.1 Overview

The front end for this application was made using Next JS, a framwork of ReactJS in buiding modern applications. Some of the features of Next JS include:-

- Server Side Rendering(SSR):Pages can be rendered on the server before being sent to the browser, improving performance and SEO.
- File-based Routing – Pages are created by adding files to the pages folder—no need to manually define routes.

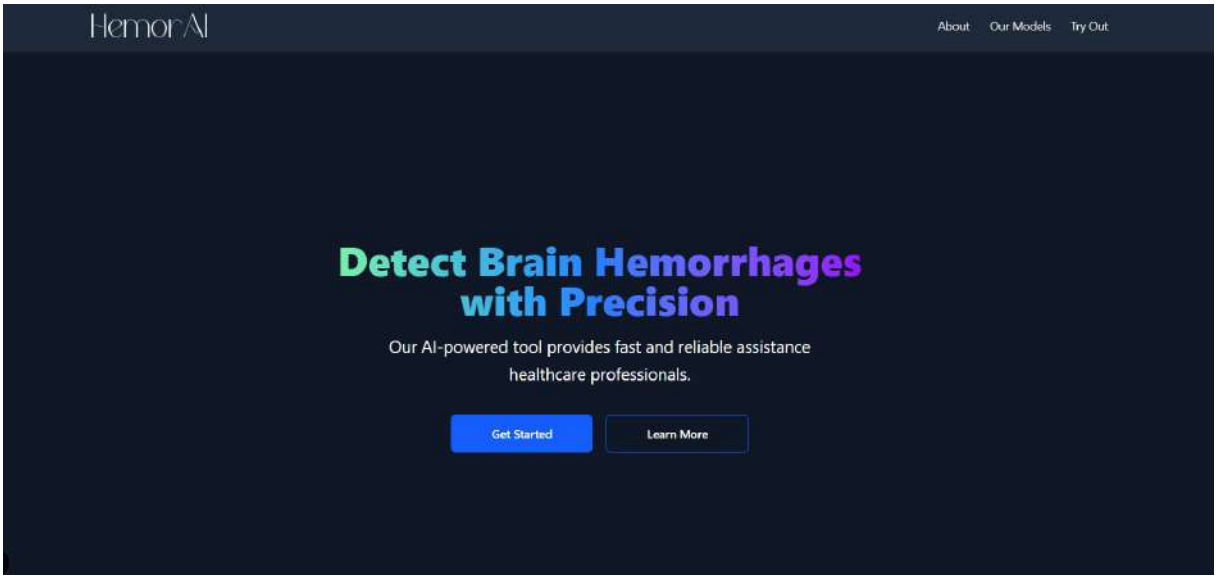


Figure 6.1: Home Page

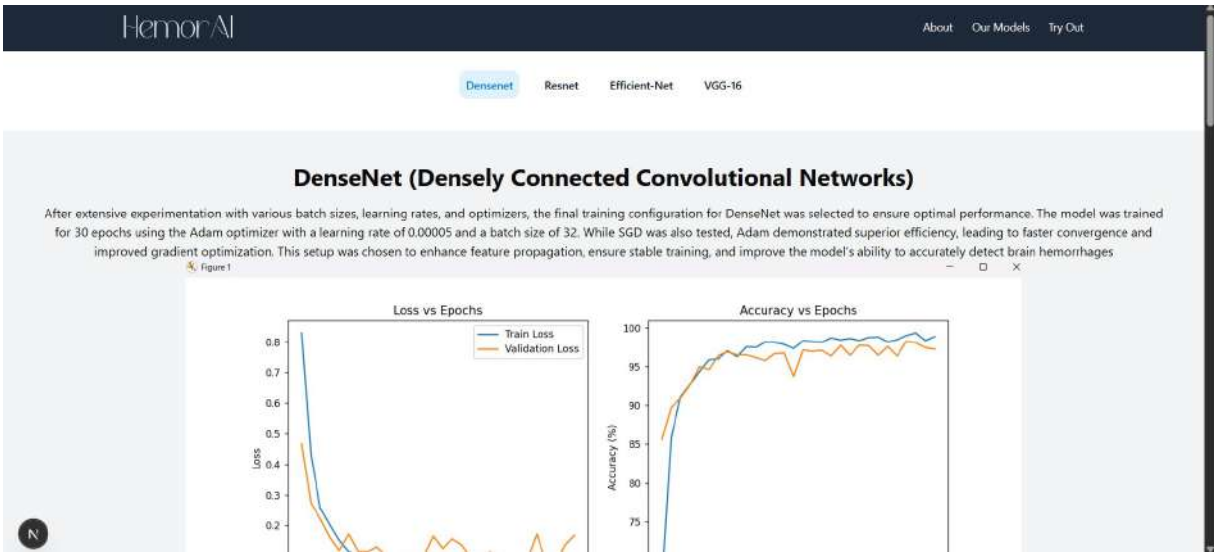


Figure 6.2: Models Page

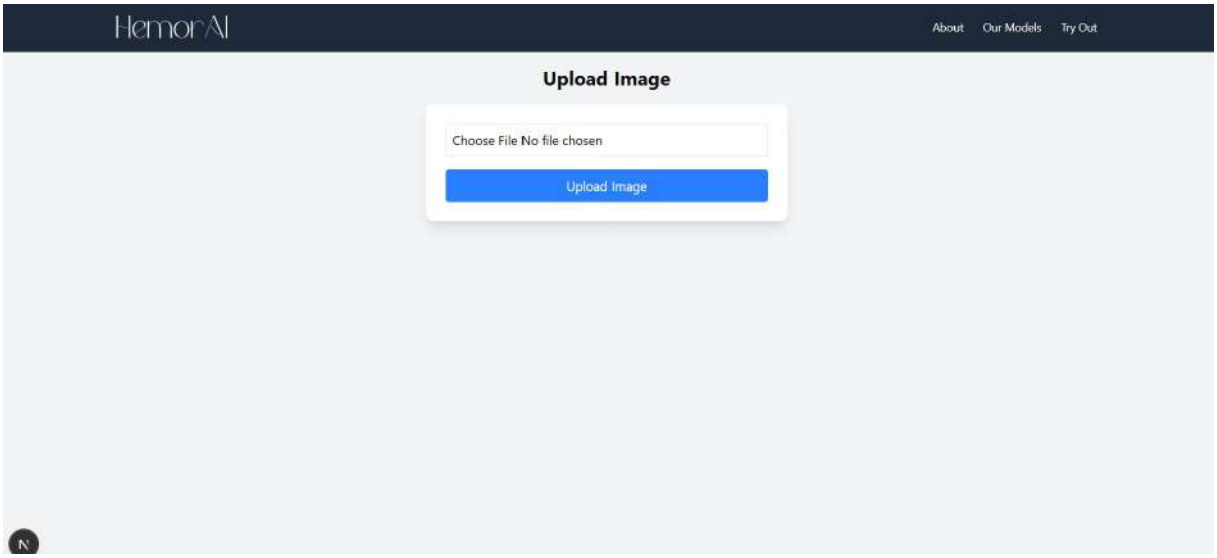


Figure 6.3: Upload Page

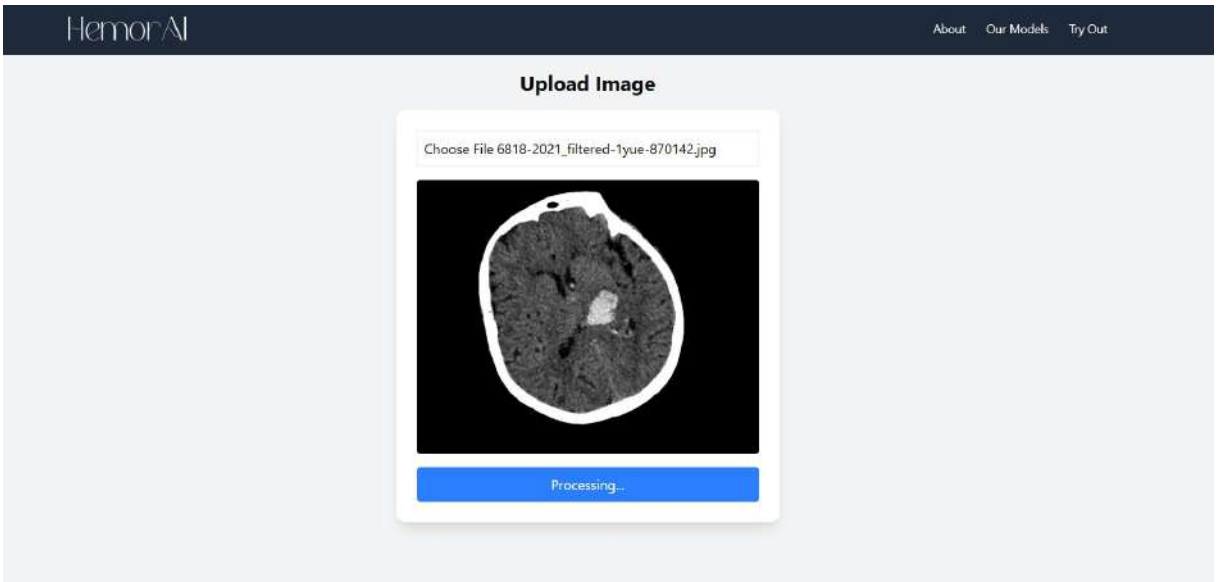


Figure 6.4: Upload Page-Processing

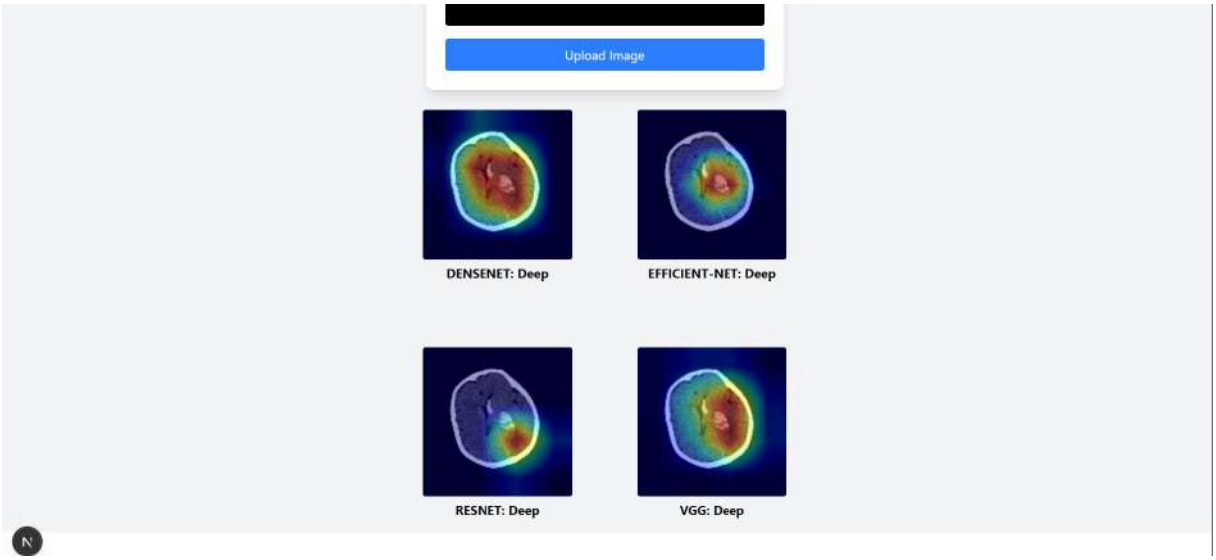


Figure 6.5: Results

The results and the performance of each of the models used is explained in the next section.

6.2 Quantitative Result

6.2.1 Densenet-121

In this study we used the Densenet 121 variant and it has around 7,978,856 trainable parameters.However this number can vary depending on the input image size and number of classes. Initially we started training with 20 epcohs with a numerous combination of learning rates and then increased the epochs to 30 and tried out with various combination of learning rates.In order to compute the loss between the predicted and the true labels, we used categorical cross entropy as the loss function. Adam optimizer was used in updating the weights of the model during backpropagation.

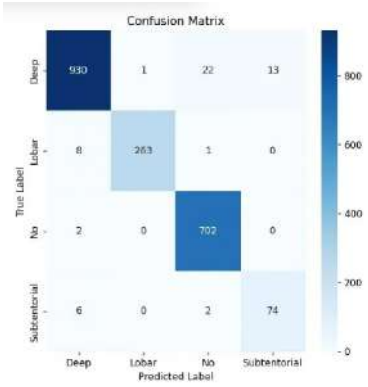


Figure 6.6: Confusion Matrix-Densenet

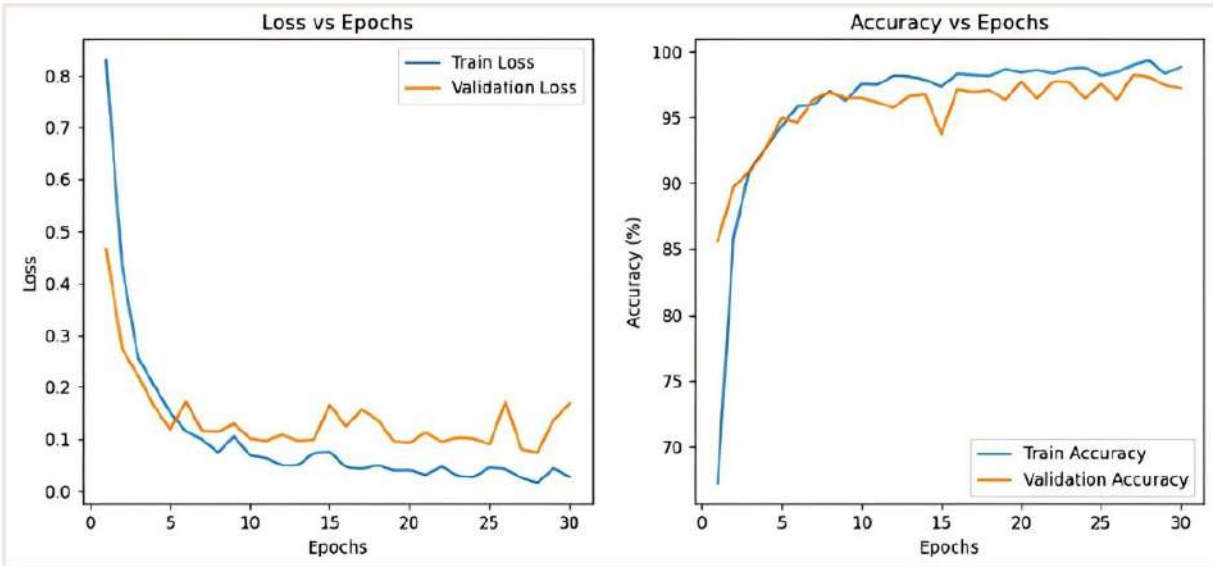


Figure 6.7: Loss vs Epochs and Accuracy vs Epoch (Densenet)

Table 6.1: Training and Validation Accuracy for DenseNet-121

Epochs	**Learning Rate**	**Train Accuracy (%)**	**Validation Accuracy (%)**
20	0.00001	97.45	96.69
20	0.0001	97.69	95.41
30	0.00005	98.84	97.23
30	0.00001	98.38	98.12
30	0.0001	98.46	97.97

As we can see from table 6.1 that the Densenet model with 30 epoch size and learning rate 0f 0.00005 was the best model and it achieved a validation accuracy of 97.23%. :ower

learning rates and fewer epochs allow for a better generalization, while higher learning rates with prolonged training can give us increased accuracy but it can lead to overfitting. So it is essentially a tradeoff between loss and accuracy

6.2.2 Resnet-50

The second model employed in this study is ResNet- 50, which leverages skip connections to enable the flow of information across layers, effectively addressing the vanishing gradient problem. Initially, the model was trained for 20 epochs, during which various learning rates were tested. Subsequently, the number of epochs was increased to 30, and different optimizers were explored to enhance performance. It was observed that the Adam optimizer with a learning rate of 0.00005 yielded the best results on the test dataset. The performance metrics for ResNet-50, obtained from different combinations of learning rates and epochs, are summarized in table 6.2

Table 6.2: Training and Validation Accuracy for ResNet-50

Epochs	**Learning Rate**	**Train Accuracy (%)**	**Validation Accuracy (%)**
20	0.00001	98.76	97.43
20	0.0001	96.45	86.36
30	0.00005	98.65	97.43
30	0.00001	98.88	97.88
30	0.0001	98.05	76.33

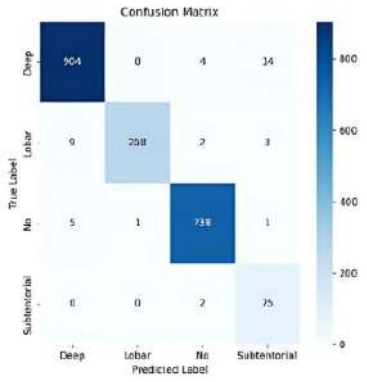


Figure 6.8: Confusion Matrix (Resnet)

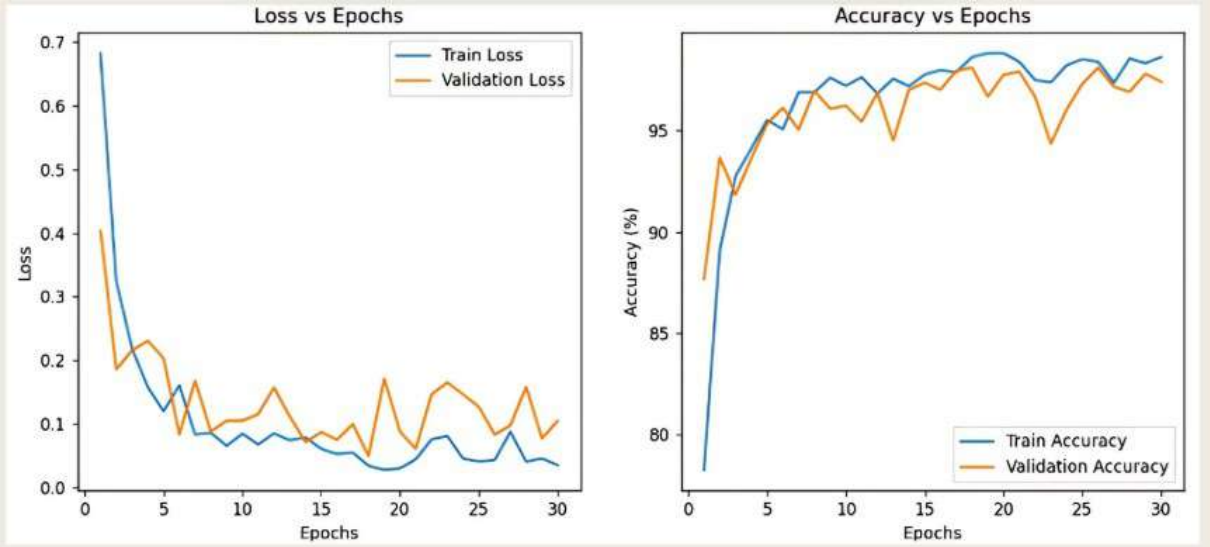


Figure 6.9: Loss vs Epochs and Accuracy vs Epoch (Resnet)

6.2.3 VGG-16

The next model employed in this study was VGG-16. To assess performance,combinations of learning rates (0.0001 and 0.001) and epochs (20 and 30 epochs) were evaluated. The model that performed best on unseen data used a learning rate of 0.001 and achieved a validation accuracy of 95.65% at 20 epochs. It is important to note that higher accuracy does not necessarily translate to better model interpretability. Additionally, extending the training to 30 epochs did not lead to any significant performance gains, indicating that 20 epochs were sufficient for effective training in this case.

Table 6.3: Training and Validation Accuracy for VGG-16

Epochs	**Learning Rate**	**Train Accuracy (%)**	**Validation Accuracy (%)**
20	0.0001	97.54	94.99
20	0.001	96.86	95.65
30	0.0001	95.30	94.80
30	0.001	96.45	95.39

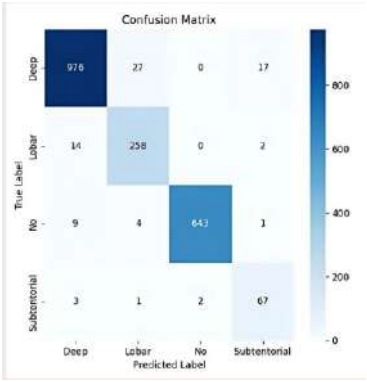


Figure 6.10: Confusion Matrix (VGG-16)

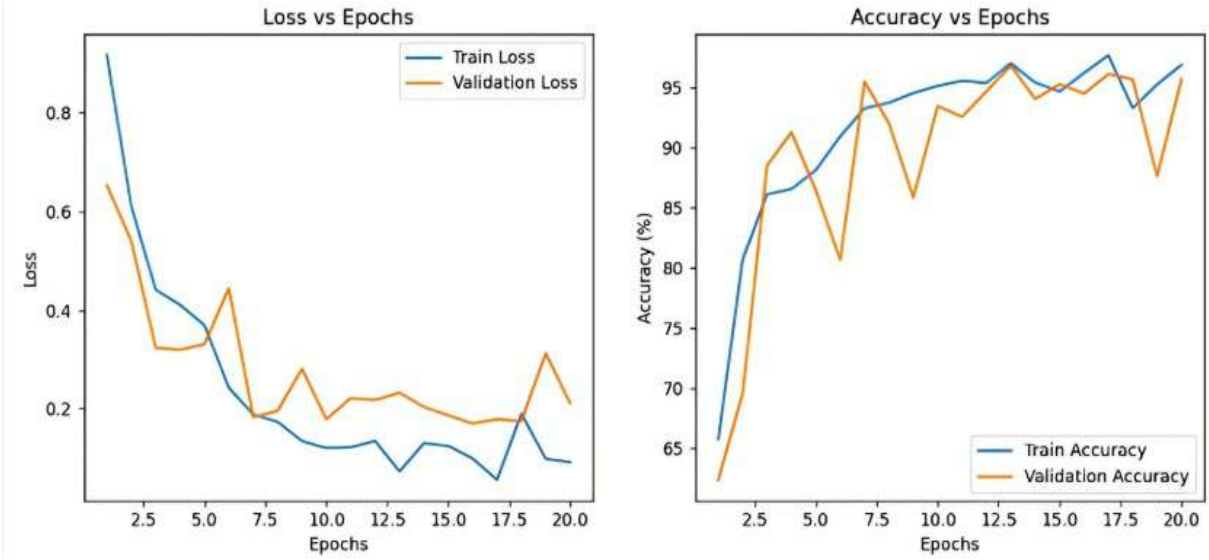


Figure 6.11: Loss vs Epochs and Accuracy vs Epoch (VGG-16)

6.2.4 EfficientNet B0

EfficientNet B0 is a variant of the EfficientNet family, known for its optimized performance and computational efficiency. In this study, we utilized the EfficientNet B0 model, which consists of 82 layers. The model was trained with different combinations of epochs and learning rates to determine the best-performing configuration.

Among these, training EfficientNet B0 for 20 epochs with a learning rate of 0.0001 yielded the highest accuracy. However, the accuracy differences across various hyperparameter combinations were minimal. For optimization, we employed the Adam optimizer, and Categorical Cross-Entropy was used as the loss function.

Table 6.4: Training and Validation Accuracy for EfficientNet

Epochs	**LR**	**Train Accuracy (%)**	**Validation Accuracy (%)**
20	0.0001	97.85	97.63
20	0.001	95.76	94.17
30	0.00005	97.70	97.07
30	0.001	96.49	97.58

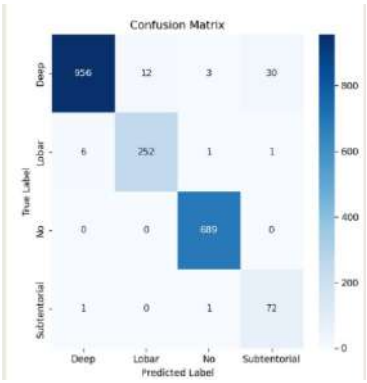


Figure 6.12: Confusion Matrix (EfficientNet-B0)

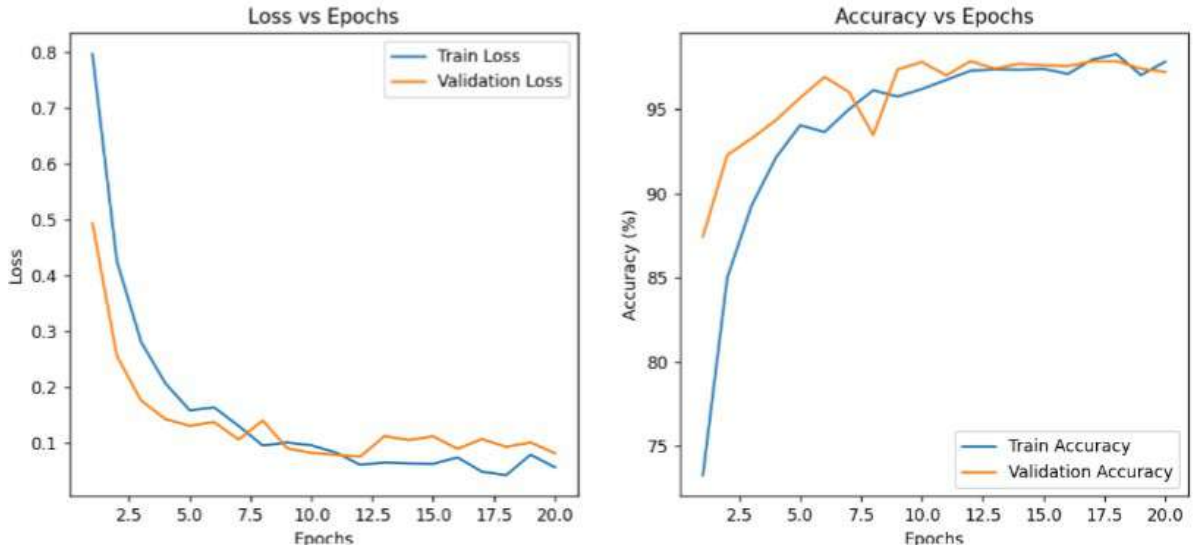


Figure 6.13: Loss vs Epochs and Accuracy vs Epoch (EfficientNet-B0)

6.2.5 Summary

In this chapter, we delved into the analysis of the results obtained from various deep learning models used for the task of brain hemorrhage classification. The performance of each model was evaluated and compared based on different metrics. The results discussed above highlight how each model performed under various hyperparameter configurations and training conditions. Among the models evaluated in this study, **EfficientNet-B0** exhibited the best per- formance, achieving an accuracy of 97.3%, along with a superior balance of precision, recall, and efficiency. **ResNet** remains a strong alternative, offering competitive accu- racy and F1-score. **DenseNet** also performed well but did not surpass EfficientNet or ResNet. **VGG-16**, as an older architecture, had the lowest performance, confirming that modern deep learning architectures outperform traditional CNN models. All the models showed little overfitting, as seen in the training and validation accuracy plots closely tracking each other. The lack of a wide gap between these plots showed that the models generalized well to new data, with no evident indications of overfitting visible in the performance plots. Although high training accuracy is desirable, it does not necessarily imply better gen- eralization. Overfitting remains a potential issue, requiring careful regularization and validation strategies. Grad-CAM was integrated into all models, and results showed that all four models were able to effectively highlighted hemorrhage regions in most cases, further reinforcing their potential for clinical applications. .

Chapter 7

Conclusions and Future Scope

This research demonstrates the efficacy of deep learning models in classifying brain hemorrhage. The objective of this study is to create an interpretable and accurate deep learning model that has the ability to automatically detect brain hemorrhages from CT scans. By leveraging CNN architectures along with explainability techniques like Grad-CAM, the system enhances both accuracy and interpretability so that medical professionals can interpret and verify the model's conclusions.

Among the models evaluated in this study, **EfficientNet-B0** exhibited the best performance. **ResNet** remains a strong alternative, offering competitive accuracy and F1-score. **DenseNet** also performed well but did not surpass EfficientNet or ResNet. **VGG-16**, as an older architecture, had the lowest performance, confirming that modern deep learning architectures outperform traditional CNN models.

Although high training accuracy is desirable, it does not necessarily imply better generalization. Overfitting remains a potential issue, requiring careful regularization and validation strategies. Grad-CAM was integrated into all models, and results showed that all four models were able to effectively highlighted hemorrhage regions in most cases, further reinforcing their potential for clinical applications.

Future work on this project can aim at improving model robustness by adding larger and more diverse datasets, including multi-modal medical imaging data such as MRI and PET scans. The integration of more advanced explainability techniques, like SHAP or LIME, can further enhance model transparency and trustworthiness. Additionally, deployment of the system as a cloud-based or mobile application can make it more accessible for real-world clinical use. Further research can also explore federated learning to support model training collaboration across multiple hospitals without compromising data privacy.

References

[1] S. E. Arman, S. S. Rahman, N. Irtisam, S. A. Deowan, M. A. Islam, S. Sakib, and M. Hasan, "Intracranial hemorrhage classification from ct scan using deep learning and bayesian optimization," *IEEE Access*, pp. 3446–3460, 2023.

[2] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, pp. 618–626, 2017.

[3] A. Singh, S. Sengupta, and V. Lakshminarayanan, "Explainable deep learning models in medical image analysis," *Journal of Imaging*, vol. 6, no. 6, p. 52, 2020.

[4] M. Rajagopal, S. Buradagunta, M. Almeshari, Y. Alzamil, R. Ramalingam, and V. Ravi, "An efficient framework to detect intracranial hemorrhage using hybrid deep neural networks," *Brain Sciences*, vol. 13, no. 3, p. 400, 2023.

[5] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[7] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 6105–6114. [Online]. Available: https://arxiv.org/abs/1905.11946

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations (ICLR)*, pp. 1–11, 2015, arXiv:1409.1556. [Online]. Available: https://arxiv.org/abs/1409.1556

Appendix A: Presentation

Main Project on

BRAIN HEMORRHAGE DETECTION

USING DEEP LEARNING AND EXPLAINABLE AI

GROUP MEMBERS

RANJIT SHINE (U2103170)
REUBEN SIMON GEORGE (U2103173)
ROHAN THUNDIL RAJEEV (U2103182)
RON THOMAS VIJY (U2103184)
S8 CSE GAMMA

GUIDED BY

MS. SREEJADEVI P
ASST. PROFESSOR, DEPT. OF CSE
RSET

Contents

Introduction

- Brain hemorrhage is a type of bleeding within the brain caused by ruptured blood vessels.
- It can lead to severe neurological damage or fatality, making accurate detection crucial for effective treatment.

Introduction

Brain Hemorrhage is mainly caused due to:

- High blood pressure (Hypertension)
- Head injury
- Aneurysm
- Brain tumors

Problem Definition

- Traditional diagnosis methods rely on manual interpretation of CT scans by radiologists.
- Early signs of hemorrhage may be missed by radiology technicians.
- Deep Learning models have shown promising results in medical image analysis tasks

Problem Definition

In a study conducted by analyzing the delay in Radiological Reporting shows that:

- 17.8% of reports were delayed beyond 72 hours.
- The leading causes of delay :
  - incomplete clinical history (34.8%)
  - Radiologist overburden (29.7%)
  - Poor communication (14%) between clinicians and radiologists

Purpose & Need

- To ensure fast and accurate classification of hemorrhage to minimize turnaround time
- To support clinicians by providing a reliable second opinion, reducing diagnostic errors.

Applications & Beneficiaries

- Radiology Technicians
  - Can decrease the turnaround time.
  - Aid in priotizing scans for review by radiologists
  - AI-powered models can assist the healthcare workers in making their decisions.
- Medical Students & Researchers
  - Learn from visual explanations (Grad-CAM) to understand real hemorrhagic patterns.

Objectives

- This study presents a comparative analysis of different deep learning models including EfficientNet, ResNet, DenseNet, and VGG16 in hemorrhage classification .
- The performance of the models are evaluated based on their accuracy, precision recall and F1 score.
- Apply Grad-CAM for visual interpretability of predictions

Literature Survey

Paper	Dataset	Methods	Results	Advantages	Disadvantages
Luis Cortés-Ferre, Miguel Angel Gutiérrez-Naranjo, Soledad Pérez-Sánchez, Marcin Balcerzyk, Juan José Egea-Guerrero. "Deep Learning Applied to Intracranial Hemorrhage Detection." J. Imaging, 2023, 9(2), 37.	Dataset: CT scan slices from patients (details unspecified)	• Classification Model: Combined EfficientDet and ResNet architectures. • Explainability: Grad-CAM for visual explanation	Achieved 92.7% accuracy in classifying presence/absence of hemorrhage. Obtained 0.978 ROC AUC (Receiver Operating Characteristic - Area Under Curve).	• **High accuracy** with **low computational resources** (EfficientDet D0 model). • Integrated **ResNet for deep learning stability** and EfficientDet for **resource optimization**. • Supports **rapid diagnosis**, enabling decision support in clinical settings.	• Limited to D0 model (simplest EfficientDet) • Post-hoc explainability, not inherently interpretable • May need more testing on large-scale datasets
Mohammed Ammara , Mohamed Amine Lamria , Saïd Mahmoudib , Amel Laidia "Deep Learning Models for Intracranial Hemorrhage Recognition: A comparative study" (2021)	Dataset: images of DICOM format with over 25000 brain CT scans provided by RSNA (Radiological society of North America).	• Models used: ResNet50 , VGG16 , Xception , InceptionV3, InceptionResNetV2. • Parameters Kept: Batch_size=16 ,Optimizer function = Adam and learning rate =0.001	• Among the 5 models VGG16 provides best accuracy of 96%. • Accuracy of other models: ResNet50 :90 , Xception :91 , InceptionV3 :91, InceptionResNetNetV2 :90	• **High accuracy**: VGG16 achieved 96%. • **Large dataset**: 25,000+ CT scans from RSNA. • **Model comparison**: Tested 5 top CNNs under same settings.	• Basic tuning: Limited hyperparameter optimization. • Heavy models: Some architectures are resource-intensive. • Lower accuracy for others: ResNet50 and InceptionResNetV2 only reached 90%.

Literature Survey

Paper	Dataset	Methods	Results	Advantages	Disadvantages
Yassine Barhoumi, Nidhal Carla Bouayaya, and Ghulam Rasool. "Efficient Scopeformer: Toward Scalable and Rich Feature Extraction for Intracranial Hemorrhage Detection." IEEE Access, 2023.	• Dataset: RSNA ICH dataset (870,301 CT scans, 5 hemorrhage subtypes). • Preprocessing: Hounsfield Unit (HU) windowing (brain, subdural, soft tissue).	• Used a hybrid CNN-ViT model (Scopeformer) with modular backbones like Xception, DenseNet. • Applied feature projection to reduce redundancy before Transformer input. • Pretrained on ImageNet-1k, then fine-tuned on RSNA dataset. • Used weighted multi-label log loss to handle class imbalance.	• Accuracy: The model is 96.94% accurate at spotting brain bleeds in CT scans. • Efficiency: Uses 8× fewer parameters (smaller/faster) but works just as well. • Sweet Spot: Works best with 8 ViT layers—more layers hurt performance.	• Diverse features from multiple CNNs improve robustness • Scalable design suits different resource limits • Efficient ViT with reduced complexity • Flexible pretraining options	• Complex tuning required • Risk of overfitting in large models • High resource use due to ViT attention • Needs medical pretraining for best results.
Ramprasaath R. Selvaraju ,Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra "Visual Explanations from Deep Networks via Gradient-based Localization." Georgia Tech / Facebook AI Research	Dataset not mentioned	• Used Grad-CAM to show which parts of an image a CNN focuses on. • Combines with Guided Grad-CAM for clearer, detailed visuals. • Tested on image classification, captioning, and visual Q&A.	• Shows why a model made a prediction. • Helps people trust model decisions and find biases. • Can be used with many tasks and models.	• Works on any CNN without retraining. • Useful across many AI tasks. • Shows how models think. • Helps spot biased data	• Blurry visuals unless combined with other methods. • Only explains final CNN layer. • Not detailed at the pixel level.

Literature Survey

Paper	Dataset	Methods	Results	Advantages	Disadvantages
Shifat E. Arman et al. "Intracranial Hemorrhage Classification From CT Scan Using Deep Learning and Bayesian Optimization." University of Dhaka / Aromo Health	RSNA ICH dataset (1M+ CT scans, 5 hemorrhage subtypes).	• Model: DenseNet121 optimized with Bayesian Optimization (BO). • BO tuned: learning rate, optimizer (Adam/SGD/RMSProp), dense layer nodes. • Training: 50 epochs, binary cross-entropy loss, sigmoid output.	• Best Model: Achieved 85.45% precision, 82.20% recall for intraventricular hemorrhage. • Optimization: BO reduced validation loss to 0.067 (19th iteration).	• Faster hyperparameter tuning with Bayesian Optimization (BO). • Helps radiologists quickly detect hemorrhages. • HU windowing improves subtle hemorrhage visibility. • Scales well to large datasets.	• Struggles with rare hemorrhage types. • High GPU usage due to DenseNet + BO. • Doesn't use patient history. • Fixed threshold may miss some cases.

Software/Hardware Requirements

Software

- Programming Language: Python 3.12
- Libraries Used:
  - PyTorch: For building and training the models.
  - Grad-CAM: PyTorch implementaion for XAI.
  - OpenCV: For image pre processing.
  - Seaborn: For visualizing the models.
  - Pandas: For data manipulation and loading.

Hardware

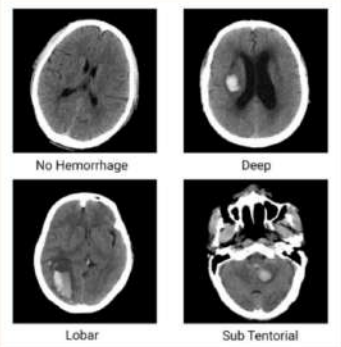
- CPU and Memory:
  - A multi-core CPU (e.g., Intel Core i7 or higher) to handle data pre processing and training tasks efficiently.
  - Sufficient RAM (at least 16 GB).
- GPU:
  - NVIDIA GPUs GeForce RTX series.

Dataset Organization

- The dataset was split into 80% for training and 20% for testing.
- Dataset Identified: https://www.kaggle.com/datasets/louisfan/dataset-for-dtvit

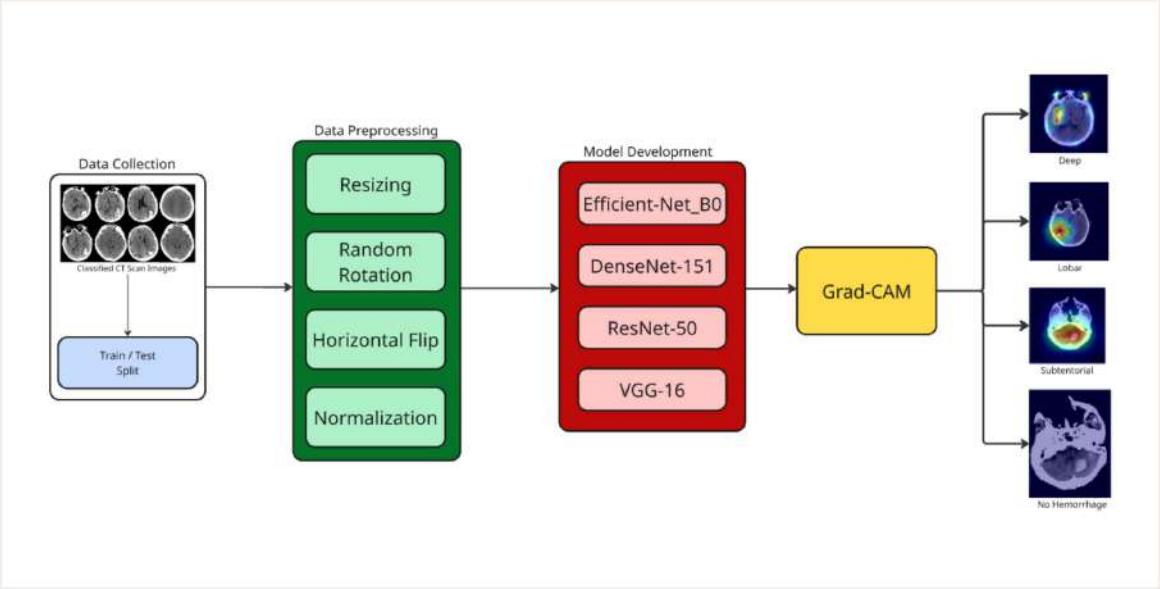
Classes	Train(80%)	Test(20%)	Total
No Hemorrhage	3526	770	4296
Deep	4874	1219	6093
Lobar	1321	335	1656
Sub Tentorial	396	99	495

Class-wise Train-Test Data Distribution

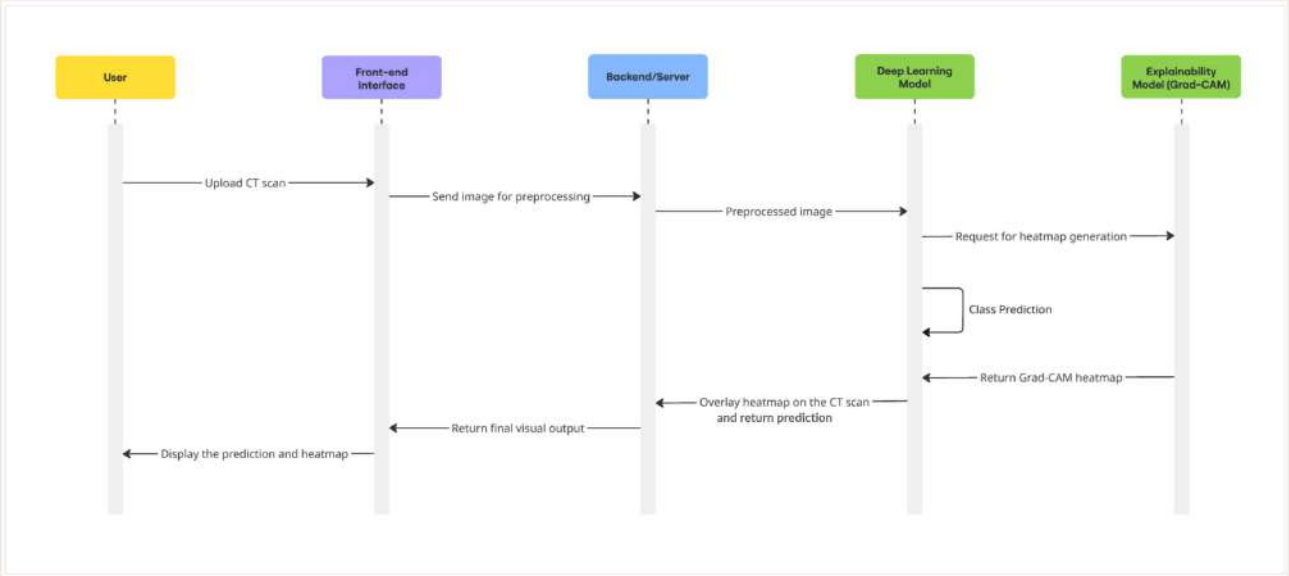


Data Samples

Architectural Diagram

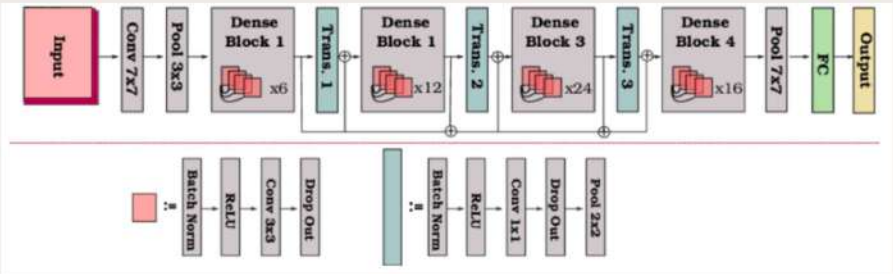


Sequence Diagram



Model Descriptions

DenseNet -121



- DenseNet stands for Dense Convolutional Network, known for its dense connectivity pattern.
- Each layer connects to every other layer in a feed-forward manner within a dense block.
- Starts with a 7×7 convolution layer followed by a 3×3 max pooling layer.

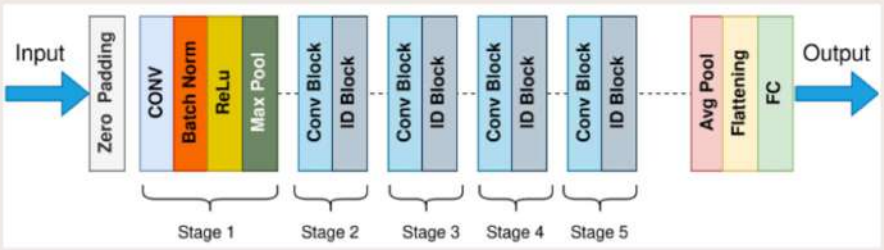
DenseNet -121

- DenseNet-121 has 4 Dense Blocks with 6, 12, 24, and 16 layers each.
- Inside each block, every layer shares its output with all following layers.
- Between these blocks, Transition Layers are used to shrink the image size and reduce features.
- These dense connections keep important features and help detect even small signs of brain hemorrhage.

DenseNet -121

- A Global Average Pooling layer is used to summarize the image features.
- Then, a Fully Connected layer gives the final output – classifying the type of hemorrhage (Lobar, Deep, Subtentorial, or None).
- DenseNet is lightweight and efficient, so it works well even with smaller medical datasets.

Resnet-50



- ResNet-50 is a deep neural network with 50 layers.
- It starts with a 7×7 convolution layer and max pooling to extract basic image features.
- Followed by Batch Normalization and ReLU activation.
- Max Pooling layer reduces the size of the feature map.

Resnet-50

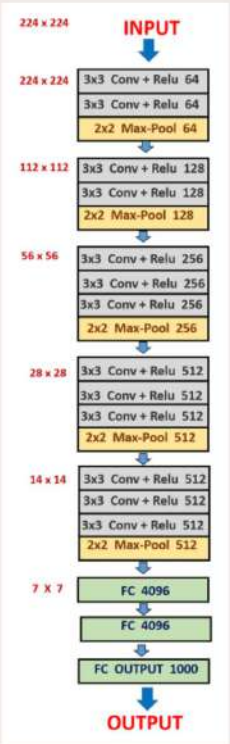
- Each stage contains a series of Convolution Blocks and Identity Blocks.
- Skip connections (shortcuts) are used to pass outputs from earlier layers to later ones.
- Average Pooling reduces spatial dimensions.

Resnet-50

- Flattening prepares the output for classification.
- A Fully Connected (FC) layer predicts the type of brain hemorrhage (e.g., Lobar, Deep, Subtentorial, Non-hemorrhagic).
- Helps in learning subtle features in medical images.
- Skip connections make training easier and more accurate.

VGG-16 (Visual Geometric Group -16)

- VGG16 is a deep Convolutional neural network with 16 layers.
- First stage: two 3×3 Conv layers (64 filters) + Max Pooling.
- Second stage: two Conv layers (128 filters) + Max Pooling.
- Third stage: three Conv layers (256 filters) + Max Pooling.
- Fourth stage: three Conv layers (512 filters) + Max Pooling.
- Fifth stage: three Conv layers (512 filters) + Max Pooling.

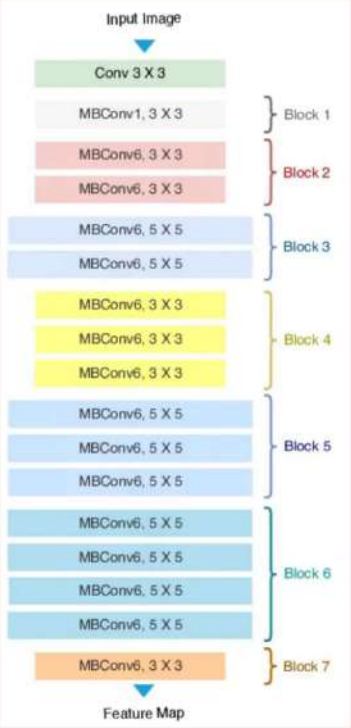


VGG-16 (Visual Geometric Group -16)

- ReLU activation is used throughout for non-linearity.
- Output is flattened into a 1D feature vector.
- Two Fully Connected layers (4096 neurons each) learn deep representations.
- Final FC layer predicts the hemorrhage type (e.g., Lobar, Deep).
- Small 3×3 filters help detect fine features in CT scans.
- Effective for subtle pattern detection in limited medical data.

Efficient-Net_B0

- EfficientNet is a family of models for image classification developed by Google Brain.
- EfficientNet models range from B0 to B7, increasing in size and accuracy.
- Efficientnet B0 has 82 layers
- Uses compound scaling to balance depth, width, and resolution.



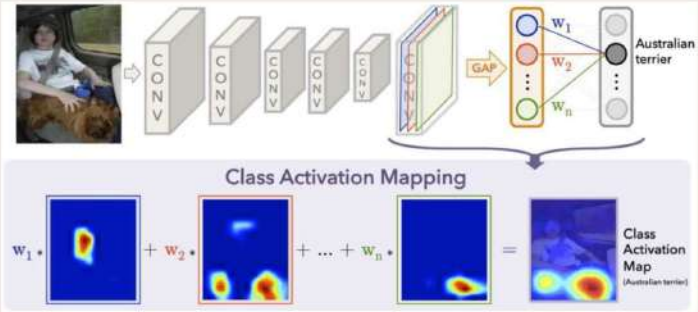
Efficient-Net_B0

- A 3×3 convolutional layer extracts basic features like edges and textures.
- Block 1 uses MBConv1 to perform efficient feature extraction.
- Block 2 applies MBConv6 with 3×3 filters to capture early patterns.
- Block 3 uses MBConv6 with 5×5 filters for mid-level feature learning.
- Block 4 applies MBConv6 with 3×3 filters to refine spatial features.
- Block 5 deepens the model with four MBConv6 layers using 5×5 filters.

Efficient-Net_B0

- Block 6 continues deep learning with two MBConv6 layers (5×5).
- Block 7 finishes with a 3×3 MBConv6 layer for final feature encoding.
- A global average pooling and dense layer classify hemorrhage types.
- EfficientNetB0 is ideal for medical images due to its high accuracy and efficiency.

Grad CAM



Grad-CAM (Gradient-weighted Class Activation Mapping) is a technique that provides visual explanations to CNNs by generating heatmaps that highlight the regions of an image that were most influential in the network's decision.

This method enables layer-wise visualization and multi-class visualizations, providing enhanced interpretability and insights into the decision- making process.

Modules

1.Data Acquisition and Preprocessing

.

- All images were resized to 128×128 pixels.
- Data Augmentation applies transformations (rotation, flipping) to enhance model generalization.
- All images were normalized to a range of [0,1]

Modules

2. Feature Extraction and Classification

VGG16

- Extracts spatial features using a simple deep architecture.
- Acts as a strong baseline for hemorrhage classification.

ResNet50

- Uses skip connections to train deeper networks.
- Learns robust features with improved accuracy.

Modules

DenseNet121

- Promotes feature reuse through dense connections.
- Efficient and performs well on limited data.

EfficientNetB0

- Balances depth, width, and resolution using compund scaling.
- Lightweight and accurate, suitable for fast inference.

Modules

- Loss Function: Categorical Cross-Entropy
- Optimizer: Adam
- Different learning rate–epoch combinations were tested for each model.
- Classes: Deep, Subtentorial, Lobar, and No Hemorrhage

Modules

3. Explainability using Grad-CAM

- Grad-CAM (Gradient-weighted Class Activation Mapping)
- Applied to all models (VGG16, ResNet50, DenseNet121, EfficientNetB0).
- Highlights important regions in brain scans influencing model predictions.
- Helps interpret model decisions and builds trust in clinical settings.

Modules

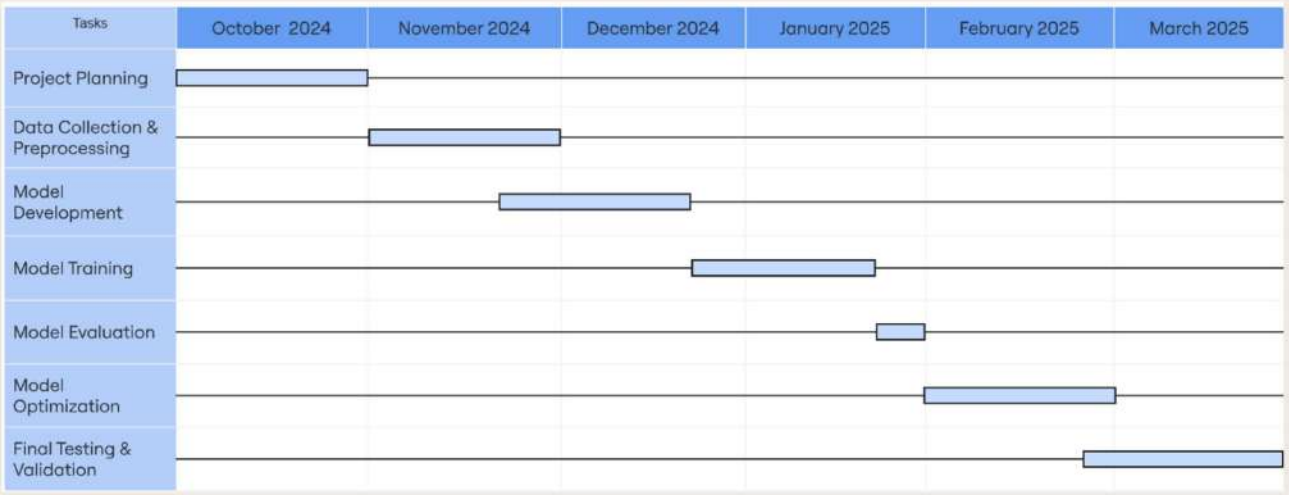
4. Comparison and Evaluation

- Models Compared: VGG16, ResNet50, DenseNet121, EfficientNetB0 using metrics like Accuracy, Precision, Recall, and F1-Score with an 80:20 train-validation split.
- Explainability: Grad-CAM heatmaps overlaid on CT images revealed focus areas, enhancing interpretability.

Work breakdown & Responsibilities

- **Model Designing and Implementation**:
  - Ranjit Shine, Reuben Simon, Rohan Thundil, Ron Thomas
- **Grad-CAM Implementation**:
  - Reuben Simon
- **Data Acquisition and UI**:
  - Ranjit Shine

Gantt Chart



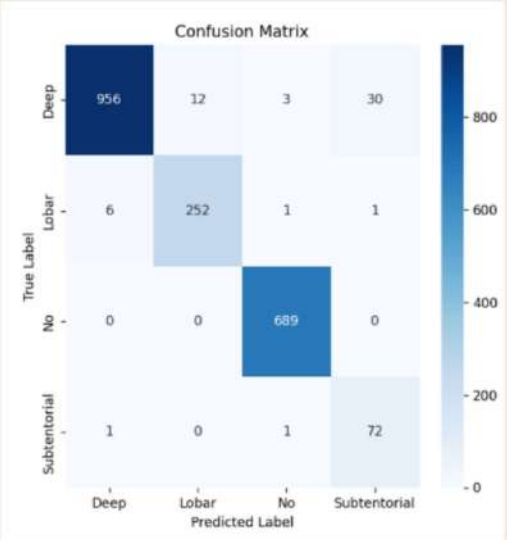
Risk and Challenges

- **Dataset Availability**: Access to high-quality, annotated CT scan datasets for brain hemorrhage classification are scarce.
- **Computational Resources and Time:** Training deep neural networks requires significant GPU power, memory, and time.

Results and Discussion

Efficient-Net

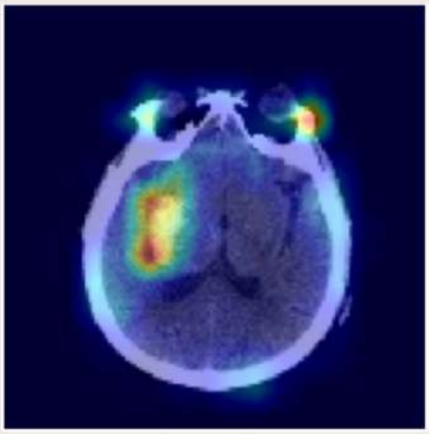
- Efficient Net Model was trained for 20 epochs
- Adam optimizer and Categorical Cross Entropy loss function was used .
- The learning rate used is **0.0001**.
- We were able to achieve a train accuracy of **97.85%** and validation accuracy of **97.63%**.



Efficient-Net

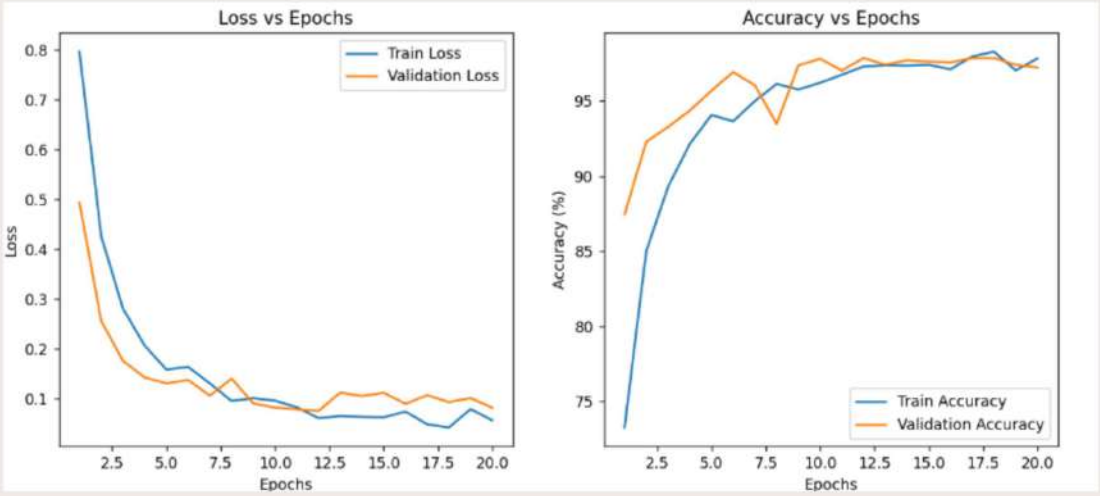
Epochs	LR	Train Accuracy	Validation Accuracy
20	0.0001	97.85	97.63
20	0.001	95.76	94.17
30	0.00005	97.7	97.07
30	0.001	96.49	97.58

Comparison - Training and Validation Accuracy



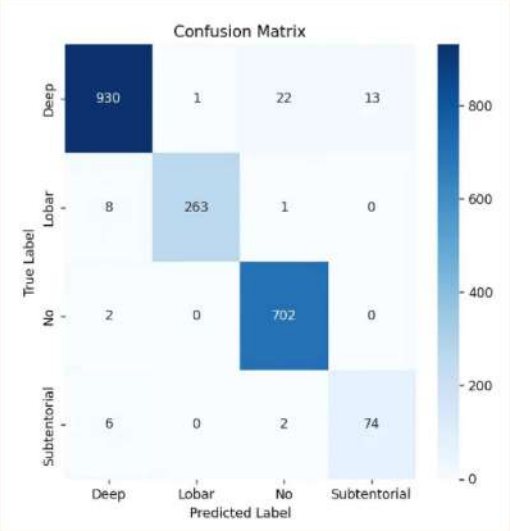
Grad-CAM

Efficient-Net



DenseNet

- The Densenet model was trained for 30 epochs.
- The optimizer we used was Adam optimizer and loss function used is Categorical Cross Entropy.
- The learning rate used is **0.00005**.
- We were able to achieve a train accuracy of **98.84%** and validation accuracy of **97.23%**.



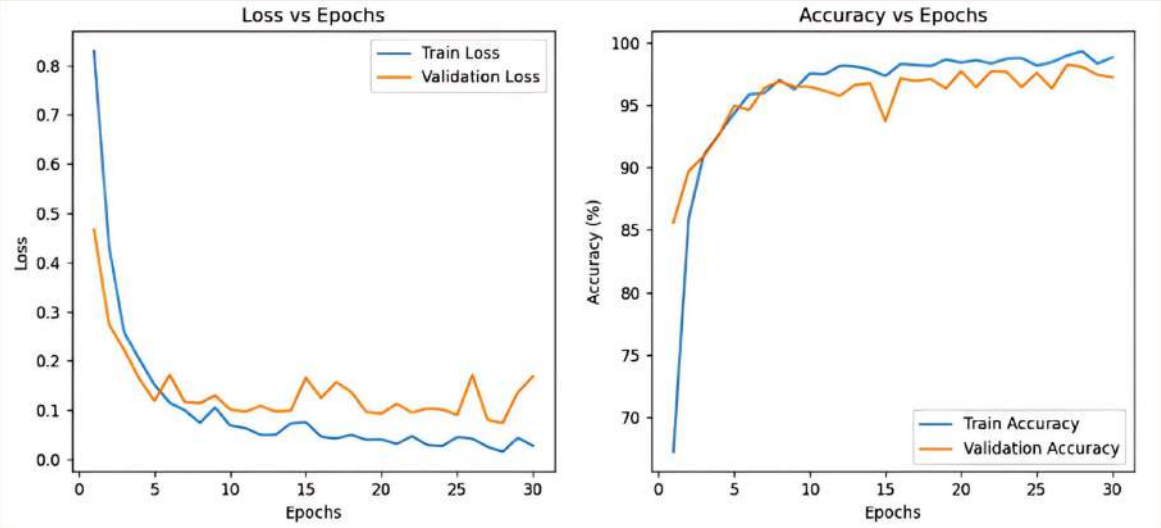
DenseNet

Epochs	LR	Train Accuracy	Validation Accuracy
20	0.00001	97.45	96.69
20	0.0001	97.69	95.41
30	0.00005	98.84	97.23
30	0.00001	98.38	98.12
30	0.0001	98.46	97.97

Comparison - Training and Validation Accuracy

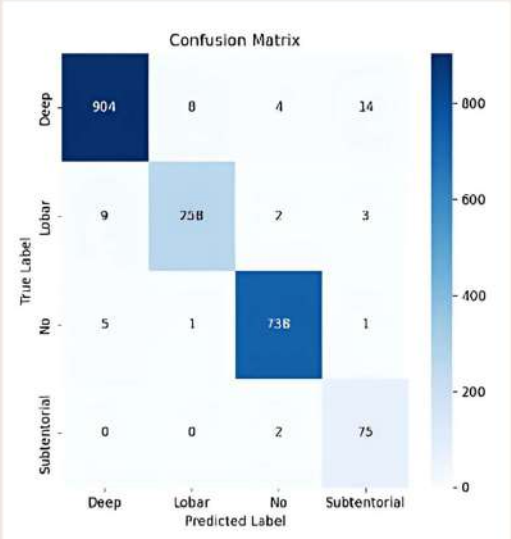
Grad-CAM

DenseNet



ResNet

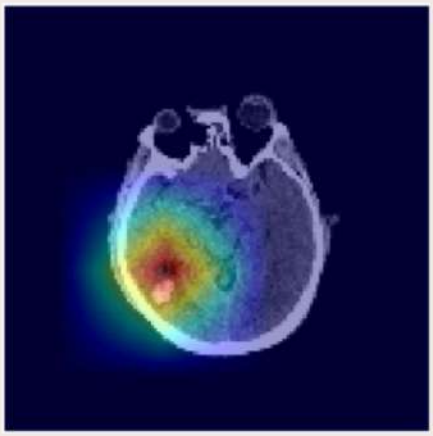
- The Resenet model was trained for 30 epochs.
- The optimizer used was Adam optimizer and loss function used is Categorical Cross Entropy.
- The learning rate used is **0.00005**.
- We were able to achieve a train accuracy of **98.65%** and validation accuracy of **97.43%**.



ResNet

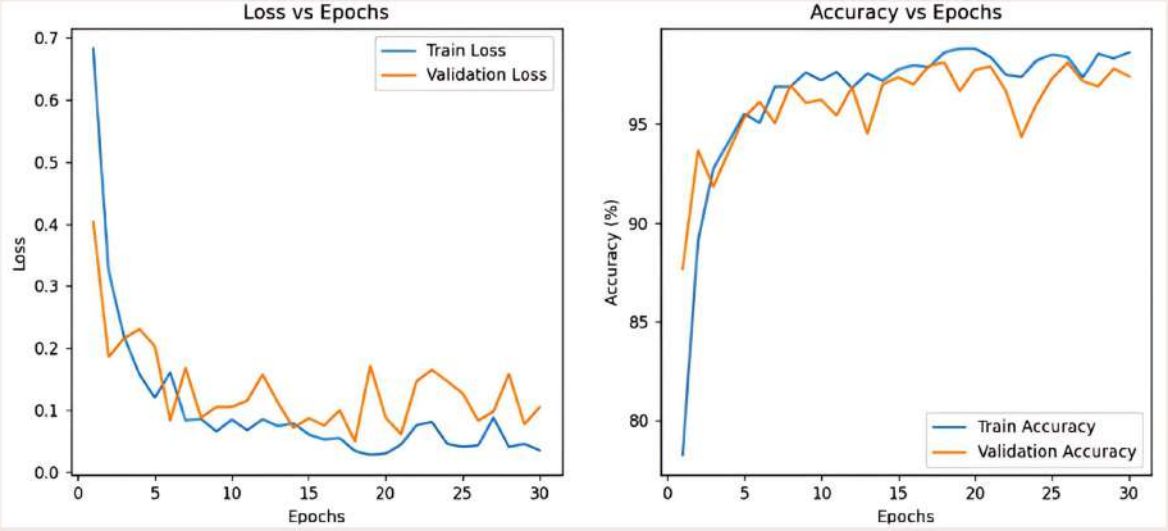
Epochs	LR	Train Accuracy	Validation Accuracy
20	0.00001	98.76	97.43
20	0.0001	96.45	86.36
30	0.00005	98.65	97.43
30	0.00001	98.88	97.88
30	0.0001	98.05	76.33

Comparison - Training and Validation Accuracy



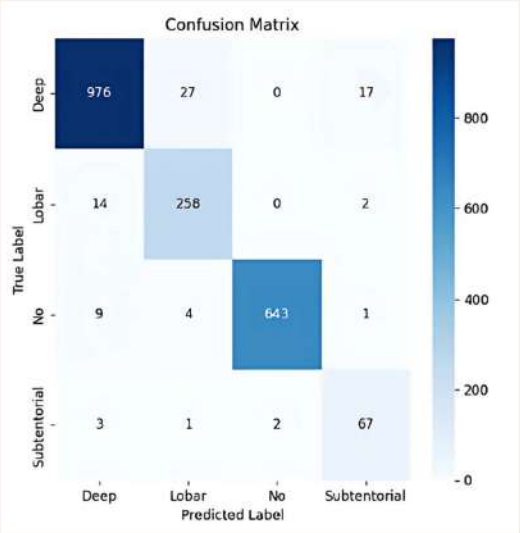
Grad-CAM

ResNet



VGG-16

- The VGG16 model was trained for 20 epochs.
- The optimizer used was Adam optimizer and loss function used is Categorical Cross Entropy.
- The learning rate used is **0.001**.
- We were able to achieve a train accuracy of **96.86%** and validation accuracy of **95.65%**.



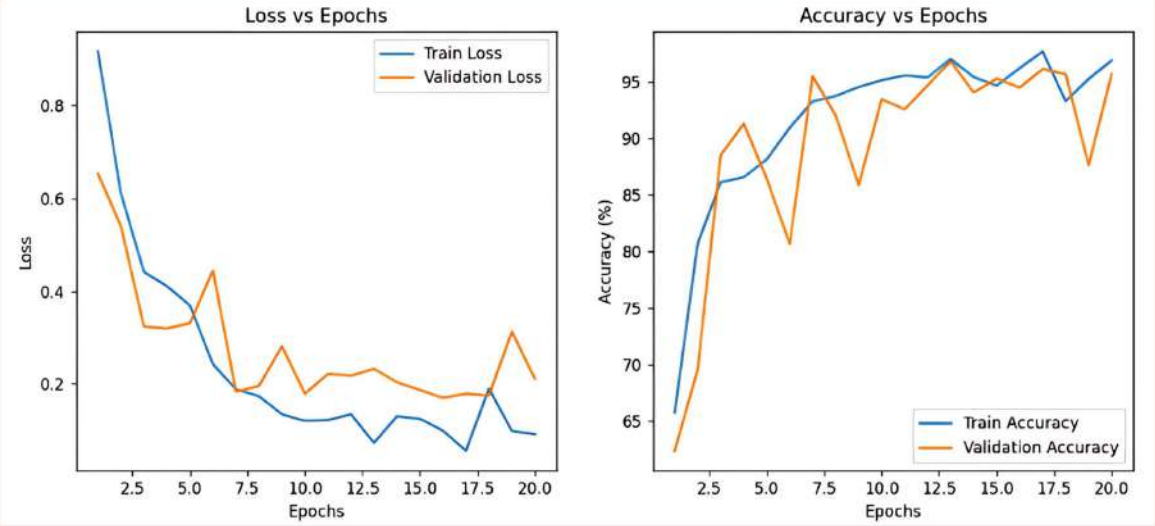
VGG-16

Epochs	LR	Train Accuracy	Validation Accuracy
20	0.0001	96.54	94.99
20	0.001	96.86	95.65
30	0.0001	95.3	94.8
30	0.001	96.45	95.39

Comparison - Training and Validation Accuracy

Grad-CAM

VGG-16



Accuracy, Recall, Precision & F1 Score

ACCURACY

$$\text{Accuracy} = \frac{\sum TP}{\sum (TP + FP + FN + TN)}$$

where TP = True Positives, FP = False Positives, TN =True Negative FN = False Negative
Measures the overall correctness of the model in classifying brain hemorrhage and non-hemorrhage cases.

PRECISION

$$\text{Precision} = \frac{TP}{TP + FP}$$

where TP = True Positives, FP = False Positives)

Measures how many predicted brain hemorrhage cases are actually correct.

Accuracy ,Recall ,Precision & F1 Score

RECALL

$$\text{Recall} = \frac{TP}{TP + FN}$$

where FN = False Negatives

Measures how many actual brain hemorrhage cases are correctly identified.

F1-SCORE

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Balances precision and recall for a more reliable performance measure

Accuracy ,Recall ,Precision & F1 Score

Model	Accuracy	Precision	Recall	F1-Score
DenseNet	97.3	97.3	97.3	97.3
ResNet	97.4	97.6	97.4	97.4
VGG-16	95.7	96.3	96.0	96.1
Efficient-Net	97.6	97.7	97.6	98.6

Conclusion

- This study demonstrates the effectiveness of deep learning models in brain hemorrhage classification.
- Among the models used for this study, Efficient Net B0 exhibited slightly superior performance, achieving higher performance metrics and validation accuracy, followed by Resnet, Densenet and VGG-16.
- Efficient-Net showed an accuracy of 97.3%
- Higher accuracy during training does not always make the model generalized. It can still lead to overfitting or poor generalization.
- Incase of GradCAM, all the four models performed well in the heatmap generation in most of the classifications.

Future Scope

- Improve model robustness by using larger, more diverse datasets, including multi-modal imaging like MRI and PET scans.
- Integrate advanced explainability techniques such as SHAP or LIME for deeper model transparency.
- Can perform Model ensembling where we can combine the outputs of multiple models to create a single, more robust output

References

- Cortés-Ferre, L., Gutiérrez-Naranjo, M.A., Egea-Guerrero, J.J., Pérez-Sánchez, S. and Balcerzyk, M., 2023. Deep learning applied to intracranial hemorrhage detection. J. Imaging, 9(2), p.37.
- Ammara, M., Lamri, M.A., Mahmoudi, S. and Laidia, A., 2022. Deep learning models for intracranial hemorrhage recognition: A comparative study. Procedia Computer Science, 196(1), pp.418–425.
- Barhoumi, Y., Bouaynaya, N.C. and Rasool, G., 2023. Efficient Scopeformer: Toward scalable and rich feature extraction for intracranial hemorrhage detection. IEEE Access, 11, pp.81656–81671.
- Singh, A., Sengupta, S. and Lakshminarayanan, V., 2020. Explainable deep learning models in medical image analysis. Journal of Imaging, 6(6), p.52.

References

- Arman, Shifat E., Rahman, Sayed Saminur, Irtisam, Niloy, and Deowan, Shamim Ahmed, 2023. Intracranial Hemorrhage Classification From CT Scan Using Deep Learning and Bayesian Optimization. IEEE Access, 11, pp. 83446–83460
- Selvaraju, Ramprasaath R., Cogswell, Michael, Das, Abhishek, and Vedantam, Ramakrishna, . Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 618–626.
- Zhou, Bolei, Khosla, Aditya, Lapedriza, Agata, Oliva, Aude, and Torralba, Antonio, 2016. Learning Deep Features for Discriminative Localization. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, Las Vegas, NV, USA, pp. 2921–2929.

Thank You!

Appendix B: Vision, Mission, Programme Outcomes and Course Outcomes

Vision, Mission, Programme Outcomes and Course Outcomes

Institute Vision

To evolve into a premier technological institution, moulding eminent professionals with creative minds, innovative ideas and sound practical skill, and to shape a future where technology works for the enrichment of mankind.

Institute Mission

To impart state-of-the-art knowledge to individuals in various technological disciplines and to inculcate in them a high degree of social consciousness and human values, thereby enabling them to face the challenges of life with courage and conviction.

Department Vision

To become a centre of excellence in Computer Science and Engineering, moulding professionals catering to the research and professional needs of national and international organizations.

Department Mission

To inspire and nurture students, with up-to-date knowledge in Computer Science and Engineering, ethics, team spirit, leadership abilities, innovation and creativity to come out with solutions meeting societal needs.

Programme Outcomes (PO)

Engineering Graduates will be able to:

1. Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. 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.

3. 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 the cultural, societal, and environmental considerations.

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

5. Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. 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.

7. 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.

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

9. Individual and Team work: Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.

10. Communication: Communicate effectively with the engineering community and with society at large. Be able to comprehend and write effective reports documentation. Make effective presentations, and give and receive clear instructions.

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

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

Programme Specific Outcomes (PSO)

A graduate of the Computer Science and Engineering Program will demonstrate:

PSO1: Computer Science Specific Skills
The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas by understanding the core principles and concepts of computer science and thereby engage in national grand challenges.

PSO2: Programming and Software Development Skills
The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry.

PSO3: Professional Skills
The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur.

Course Outcomes (CO)

After the completion of the course the student will be able to:

Course Outcome 1: Model and solve real world problems by applying knowledge across domains (Cognitive knowledge level: Apply).

Course Outcome 2: Develop products, processes or technologies for sustainable and socially relevant applications (Cognitive knowledge level: Apply).

Course Outcome 3: Function effectively as an individual and as a leader in diverse teams and to comprehend and execute designated tasks (Cognitive knowledge level: Apply).

Course Outcome 4: Plan and execute tasks utilizing available resources within timelines, following ethical and professional norms (Cognitive knowledge level: Apply).

Course Outcome 5: Identify technology/research gaps and propose innovative/creative solutions (Cognitive knowledge level: Analyze).

Course Outcome 6: Organize and communicate technical and scientific findings effectively in written and oral forms (Cognitive knowledge level: Apply).

Appendix C: CO-PO-PSO Mapping

COURSE OUTCOMES:

After completion of the course, the student will be able to:

SL.NO	DESCRIPTION	Bloom's Taxonomy Level
CO1	Model and solve real-world problems by applying knowledge across domains (Cognitive knowledge level:Apply).	Level3: Apply
CO2	Develop products, processes, or technologies for sustainable and socially relevant applications. (Cognitive knowledge level:Apply).	Level 3: Apply
CO3	Function effectively as an individual and as a leader in diverse teams and comprehend and execute designated tasks. (Cognitive knowledge level:Apply).	Level 3: Apply
CO4	Plan and execute tasks utilizing available resources within timelines, following ethical and professional norms (Cognitive knowledge level:Apply).	Level 3: Apply
CO5	Identify technology/research gaps and propose innovative/creative solutions (Cognitive knowledge level:Analyze).	Level 4: Analyze
CO6	Organize and communicate technical and scientific findings effectively in written and oral forms (Cognitive knowledge level:Apply).	Level 3: Apply

CO-PO AND CO-PSO MAPPING

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	2	2	2	1	2	2	2	1	1	1	1	2	3	3	1
CO2	2	2	2		1	3	3	1	1		1	1	3	3	1
CO3									3	2	2	3	3	3	2
CO4				2				3	2	2	3	2	3	3	3
CO5	2	3	3	1	2							1	3	3	2
CO6					2			2	2	3	1	1	3	3	3

3/2/1: high/medium/low

JUSTIFICATIONS FOR CO-PO MAPPING

Mapping	Level	Justification
101003/CS822U.1-PO1	M	Knowledge in the area of technology for project development using various tools results in better modeling.
101003/CS822U.1-PO2	M	Knowledge acquired in the selected area of project development can be used to identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions.
101003/CS822U.1-PO3	M	Can use the acquired knowledge in designing solutions to complex problems.
101003/CS822U.1-PO4	L	Can use the acquired knowledge in designing solutions to complex problems.
101003/CS822U.1-PO5	M	Students are able to interpret, improve, and redefine technical aspects for design of experiments, analysis, and interpretation of data, and synthesis of the information to provide valid conclusions.
101003/CS822U.1-PO6	M	Students are able to interpret, improve, and redefine technical aspects by applying contextual knowledge to assess societal, health, and consequential responsibilities relevant to professional engineering practices.
101003/CS822U.1-PO7	M	Project development based on societal and environmental context solution identification is the need for sustainable development.
101003/CS822U.1-PO8	L	Project development should be based on professional ethics and responsibilities.
101003/CS822U.1-PO9	L	Project development using a systematic approach based on well-defined principles will result in teamwork.
101003/CS822U.1-PO10	L	Project brings technological changes in society.
101003/CS822U.1-PO11	L	Acquiring knowledge for project development gathers skills in design, analysis, development, and implementation of algorithms.

101003/CS822U.1-PO12	M	Knowledge for project development contributes engineering skills in computing and information gatherings.
101003/CS822U.2-PO1	M	Knowledge acquired for project development will also include systematic planning, developing, testing, and implementation in computer science solutions in various domains.
101003/CS822U.2-PO2	M	Project design and development using a systematic approach brings knowledge in mathematics and engineering fundamentals.
101003/CS822U.2-PO3	M	Identifying, formulating, and analyzing the project results in a systematic approach.
101003/CS822U.2-PO5	L	Systematic approach is the tip for solving complex problems in various domains.
101003/CS822U.2-PO6	H	Systematic approach in the technical and design aspects provides valid conclusions.
101003/CS822U.2-PO7	H	Systematic approach in the technical and design aspects demonstrates the knowledge of sustainable development.
101003/CS822U.2-PO8	L	Identification and justification of technical aspects of project development demonstrates the need for sustainable development.
101003/CS822U.2-PO9	L	Apply professional ethics and responsibilities in engineering practice of development.
101003/CS822U.2-PO11	L	Systematic approach also includes effective reporting and documentation, which gives clear instructions.
101003/CS822U.2-PO12	L	Project development using a systematic approach based on well-defined principles will result in better teamwork.
101003/CS822U.3-PO9	H	Project development as a team brings the ability to engage in independent and lifelong learning.

101003/CS822U.3-PO10	M	Identification, formulation, and justification in technical aspects will be based on acquiring skills in design and development of algorithms.
101003/CS822U.3-PO11	M	Identification, formulation, and justification in technical aspects provides the betterment of life in various domains.
101003/CS822U.3-PO12	H	Students are able to interpret, improve, and redefine technical aspects with mathematics, science, and engineering fundamentals for the solutions of complex problems.
101003/CS822U.4-PO4	M	Students are able to interpret, improve, and redefine technical aspects with identification, formulation, and analysis of complex problems.
101003/CS822U.4-PO8	H	Students are able to interpret, improve, and redefine technical aspects to meet the specified needs with appropriate consideration for public health and safety, and the cultural, societal, and environmental considerations.
101003/CS822U.4-PO9	M	Students are able to interpret, improve, and redefine technical aspects for design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
101003/CS822U.4-PO10	M	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools for better products.
101003/CS822U.4-PO11	H	Students are able to interpret, improve, and redefine technical aspects by applying contextual knowledge to assess societal, health, and consequential responsibilities relevant to professional engineering practices.
101003/CS822U.4-PO12	M	Students are able to interpret, improve, and redefine technical aspects for demonstrating the knowledge of, and need for sustainable development.

101003/CS822U.5-PO1	M	Students are able to interpret, improve, and redefine technical aspects, apply ethical principles, and commit to professional ethics and responsibilities and norms of the engineering practice.
101003/CS822U.5-PO2	H	Students are able to interpret, improve, and redefine technical aspects, communicate effectively on complex engineering activities with the engineering community and 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.
101003/CS822U.5-PO3	H	Students are able to interpret, improve, and redefine technical aspects to demonstrate knowledge and understanding of the engineering and management principle in multidisciplinary environments.
101003/CS822U.5-PO4	L	Students are able to interpret, improve, and redefine technical aspects, 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.
101003/CS822U.5-PO5	M	Students are able to interpret, improve, and redefine technical aspects in acquiring skills to design, analyze, and develop algorithms and implement those using high-level programming languages.
101003/CS822U.5-PO12	L	Students are able to interpret, improve, and redefine technical aspects and contribute their engineering skills in computing and information engineering domains like network design and administration, database design, and knowledge engineering.
101003/CS822U.6-PO5	M	Students are able to interpret, improve, and redefine technical aspects and develop strong skills in systematic planning, developing, testing, implementing, and providing IT solutions for different domains, which helps in the betterment of life.

101003/CS822U.6-PO8	M	Students will be able to associate with a team as an effective team player for the development of technical projects by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
101003/CS822U.6-PO9	M	Students will be able to associate with a team as an effective team player to identify, formulate, review research literature, and analyze complex engineering problems.
101003/CS822U.6-PO10	H	Students will be able to associate with a team as an effective team player for designing solutions to complex engineering problems and design system components.
101003/CS822U.6-PO11	L	Students will be able to associate with a team as an effective team player, use research-based knowledge and research methods including design of experiments, analysis, and interpretation of data.
101003/CS822U.6-PO12	L	Students will be able to associate with a team as an effective team player, applying ethical principles and committing to professional ethics and responsibilities and norms of the engineering practice.
101003/CS822U.1-PSO1	H	Students are able to develop Computer Science Specific Skills by modeling and solving problems.
101003/CS822U.1-PSO2	H	Students develop Programming and Software Development Skills by designing algorithms and applying software development practices to model and solve real-world problems across multiple domains, ensuring industry-relevant solutions.
101003/CS822U.1-PSO3	L	Students develop Professional Skills by applying computer science fundamentals to model and solve real-world problems, fostering innovation and research to create impactful solutions that address societal needs.

101003/CS822U.2-PSO1	H	Students are able to develop Computer Science Specific Skills by designing solutions for complex engineering problems that contribute to sustainable and socially relevant applications.
101003/CS822U.2-PSO2	H	Developing products, processes, or technologies for sustainable and socially relevant applications can promote Programming and Software Development Skills.
101003/CS822U.2-PSO3	L	Developing products, processes, or technologies for sustainable and socially relevant applications enhances Professional Skills by promoting research, innovation, and entrepreneurship to address societal needs.
101003/CS822U.3-PSO1	H	Students develop Computer Science Specific Skills by effectively collaborating in diverse teams to design solutions for complex engineering problems, applying core computational principles to address national grand challenges and socially relevant applications.
101003/CS822U.3-PSO2	H	Teamwork enhances Programming Skills by ensuring efficient algorithm design and software development best practices.
101003/CS822U.3-PSO3	M	Working in a team can result in the effective development of Professional Skills.
101003/CS822U.4-PSO1	H	Students develop Computer Science Specific Skills by efficiently planning and executing tasks within given constraints, applying core computational principles to solve complex engineering problems while adhering to ethical and professional standards.
101003/CS822U.4-PSO2	H	Planning and executing tasks efficiently enhances Programming and Software Development Skills by ensuring structured development, timely delivery, and adherence to industry standards.
101003/CS822U.4-PSO3	H	Planning and scheduling can result in the effective development of Professional Skills.

101003/CS822U.5-PSO1	H	Students are able to develop Computer Science Specific Skills by creating innovative solutions to problems.
101003/CS822U.5-PSO2	H	Identifying gaps enhances Programming Skills by fostering innovation in algorithms and software development.
101003/CS822U.5-PSO3	M	Identifying gaps enhances Professional Skills by fostering research, innovation, and entrepreneurial solutions.
101003/CS822U.6-PSO1	H	Effective communication enhances Computer Science Skills by conveying complex technical solutions clearly and accurately.
101003/CS822U.6-PSO2	H	Organizing and communicating technical and scientific findings can help in the effective development of Professional Skills.
101003/CS822U.6-PSO3	H	Effective communication enhances Programming Skills by clearly presenting algorithms and software solutions.