Advanced Techniques in Deep Learning for Pancreatic Cancer Detection and Classification

A Project Report submitted in the partial fulfillment of the Requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

G.Narendra	(21471A05M4)
K.Subbarao	(21471A05P0)
N.Narendra kumar	(21471A05P7)

Under the esteemed guidance of

K.V.Narasimha Reddy M. Tech

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET
(AUTONOMOUS)
Accredited by NAAC with A+ Grade and NBA under Tier -1
NIRF rank in the band of 201-300 and an ISO 9001:2015 Certified
Approved by AICTE, New Delhi, Permanently Affiliated to
JNTUK, Kakinada KOTAPPAKONDA ROAD, YALAMANDA
VILLAGE, NARASARAOPET- 522601 2024-2025

NARASARAOPETA ENGINEERING COLLEGE: NARASARAOPET

(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled "Advanced Techniques in Deep Learning for Pancreatic Cancer Detection and Classification" is a bonafide work done by me G.Narendra(21471A05M4), K.Subbarao (21471A05P0), N.Narendra Kumar (21471A05P7) **in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING**

during 2023-2024.

PROJECT GUIDE
K. V. Narasimha Reddy, M.Tech,
Assistant. Professor

PROJECT CO-ORDINATOR
D.Venakata Reddy, M.Tech,(Ph.D)
Assistant. Professor

HEAD OF THE DEPARTMENT
Dr. S. N. Tirumala Rao, M.Tech., Ph.D
Professor & HoD

EXTERNAL EXAMINER

DECLARATION

We declare that this project work titled "**ADVANCED TECHNIQUES IN DEEP LEARNING FOR PANCREATIC CANCER DETCETION AND CLASSIFICATION**" is composed by oueselves own that the work contain here is our own except where explicitly stated otherwise in the text and that this work has been submitted for any other degree or professional qualification except as specified.

G. Narendra (21471A05M4)

K. Subbarao (21471A05P0)

N.Narendra Kumar(21471A05P7)

ACKNOWLEDGEMENT

We wish to express our thanks to carious personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman sri **M. V. Koteswara Rao**, B.Sc., who took keen interest in us in every effort throughout this course. We owe out sincere gratitude to our beloved principal **Dr. S. Venkateswarlu**, M.Tech., Ph.D., for showing his kind attention and valuable guidance throughout the course.

We express our deep felt gratitude towards **Dr. S. N. Tirumala Rao,** M.Tech., Ph.D., HOD of CSE department and also to our guide **K.V.Narasimha Reddy**, M.Tech of CSE department whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks towards **D.Venkata Reddy**, M.Tech.,(Ph.D) Associate professor & Project coordinator of the project for extending her encouragement. Their profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

We extend our sincere thanks to all other teaching and non-teaching staff to department for their cooperation and encouragement during our B.Tech degree.

We have no words to acknowledge the warm affection, constant inspiration and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends and those who involved in giving valuable suggestions had clarifying out doubts which had really helped us in successfully completing our project.

By

G. Narendra (21471A05M4)

K. Subbarao (21471A05P0)

N.Narendra Kumar (21471A05P7)

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community,

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To become a Centre of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertise in modern software tools and technologies to cater to the real time requirements of the Industry.

M3: Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Educational Objectives (PEO's)

The graduates of the programme are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems related to academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Outcomes

PO1:Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2:Problem analysis: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: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.

PO4:Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: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 limitation.

PO6: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.

PO7: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.

PO8:Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9:Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10:Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11:Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12:Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Project Course Outcomes (CO'S):

C421.1: Analyse the System of Examinations and identify the problem.

C421.2: Identify and classify the requirements.

C421.3: Review the Related Literature

C421.4: Design and Modularize the project

C421.5: Construct, Integrate, Test and Implement the Project.

C421.6: Prepare the project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes Mapping:

	P01	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**
C421.1		✓											✓		
C421.2	✓		✓		✓								✓		
C421.3				✓		✓	✓	✓					✓		
C421.4			✓			✓	✓	✓					✓	✓	
C421.5					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C421.6									✓	✓	✓		✓	✓	

Course Outcomes – Program Outcome Correlation

	P01	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1	2	3											2		
C421.2			2		3								2		
C421.3				2		2	3	3					2		
C421.4			2			1	1	2					3	2	
C421.5					3	3	3	2	3	2	2	1	3	2	1
C421.6									3	2	1		2	3	

Note: The values in the above table represent the level of correlation between CO's and PO's:

1. **Low leve**
2. **Medium level**
3. **High level**

Project mapping with various courses of Curriculum with Attained PO's:

Name of the course from which principles are applied in this project	Description of the device	Attained PO
C2204.2, C22L3.2	Gathering the requirements and defining the problem; planning to develop a model for detecting pancreatic cancer using deep learning techniques	PO1, PO3
CC421.1,C2204.3, C22L3.2	Each requirement is critically analyzed; the process model for cancer detection is identified	PO2, PO3
CC421.2,C2204.2,C22L3.3	Logical design is done using unified modeling language involving individual teamwork for cancer detection	PO3, PO5, PO9
CC421.3,C2204.3,C22L3.2	Each module for pancreatic cancer detection is tested, integrated, and evaluated	PO1, PO5
CC421.4,C2204.4,C22L3.2	Documentation is collaboratively completed by all four members in a group	PO10
CC421.5,C2204.2,C22L3.3	Each phase of the group work on Pancreatic cancer detection is Presented periodically	PO10, PO11
C2202.2,C2203.3,C1206.3, C3204.3,C4110.2	Implementation of the pancreatic cancer detection project; future updates will be based on user feedback and advancements in detection methods	PO4, PO7
C32SC4.3	The physical design includes hardware components like system type of intel®core™i7-7500UCPU@2.70gh ,Cache memory of 4MB and RAM of 8GB.	PO5, PO6

ABSTRACT

This paper presents advanced deep learning techniques for pancreatic cancer detection and classification from medical images with a specific focus on the hybrid architecture of InceptionDense. The combined paradigm retains the strengths of InceptionV3 and DenseNet121 through optimized convolutional channels with diversified scales in featured descriptions, which plays an influential role in the identification of tumors that vary in sizes and shapes. This dense connectivity in DenseNet121 promotes the reuse of features, hence enhancing the model capability to make differences in the middle of cancerous and non-cancerous cases.

The study used a dataset of 1,411 annotated CT images obtained from Kaggle with balanced training and testing sets. Key hyperparameters included a fixed learning rate of 0.0001 with binary cross-entropy loss function and Adam optimizer, training over 10 epochs. Using accuracy, precision, recall, F1-score, specificity, and R^2 score as evaluation metrics, InceptionDense had 99.75% accuracy while SSA with Stacked Deep Learning observed 99.26% only. Multi-scale feature extraction along with dense connectivity improved detection capabilities that were significantly high. This work aims to improve the diagnostic reliability of medical imaging.

Its early diagnosis and treatment, aided by healthcare professionals, will actually improve outcomes for patients with pancreatic cancer. Future research will be focused on validating these methods with larger datasets and extending them to other imaging modalities, including MRI and ultrasound, in the hopes of furthering clinical applicability.

INDEX

List of Figures

1. INTRODUCTION

Pancreatic cancer remains one of the most lethal malignancies worldwide, with a five-year survival rate of less than 10% due to late-stage diagnosis and limited effective treatment options [1].

The pancreas, located deep within the abdomen, poses significant challenges for early detection using conventional imaging techniques such as computed tomography (CT), magnetic resonance imaging (MRI), and ultrasound [2].

These methods heavily rely on the expertise of radiologists, which introduces subjectivity and variability in diagnosis, often leading to missed or delayed detection of pancreatic tumors [3].

Given the aggressive nature of pancreatic cancer and its poor prognosis, there is an urgent need for advanced, automated, and highly accurate diagnostic systems to improve early detection and patient outcomes [4].

In recent years, deep learning has emerged as a transformative technology in medical image analysis, offering state-of-the-art solutions for disease detection and classification [5].

Unlike traditional image processing techniques, deep learning models can automatically extract intricate patterns from medical images, significantly improving diagnostic accuracy and reducing human error [6].

This project, titled "Advanced Techniques in Deep Learning for Pancreatic Cancer Detection and Classification," focuses on leveraging deep learning techniques to enhance the accuracy of pancreatic cancer diagnosis. Specifically, the project introduces the InceptionDense model, a hybrid deep learnisng architecture that integrates InceptionV3 and DenseNet121 to effectively extract multi-scale features and reuse learned patterns for improved classification performance [7].

This hybrid approach enhances the detection of pancreatic tumors, which often vary in shape, size, and texture, making early diagnosis more reliable [8].

Existing deep learning models have demonstrated promising results in pancreatic cancer detection. For instance, the Sparrow Search Algorithm with Stacked Deep Learning (SSASDL-PCDC) has shown high accuracy in classifying cancerous and non-cancerous pancreatic tissues [1]. However, this model requires extensive hyperparameter tuning and has high computational demands, making real-time deployment challenging [9].

Similarly, the Computer-Aided Diagnosis (CAD) approach, which integrates machine learning with radiological imaging, aims to improve tumor identification but often struggles with feature extraction and dataset generalization [10].

One major challenge in CAD systems is the segmentation of pancreatic tumors, as studies on U-Net and its variants reveal inconsistencies in identifying tumor boundaries due to anatomical variations and poor contrast in CT images [11].

These limitations necessitate the development of a more robust and efficient detection system, such as the InceptionDense model proposed in this study [12].

In addition to deep learning, researchers have explored alternative approaches such as Variational Autoencoders (VAE) for survival prediction in pancreatic cancer patients [13].

VAE models analyze both imaging and clinical data to enhance prognosis accuracy. However, their reliability depends on the availability of diverse and well-annotated datasets, which is a major limitation in pancreatic cancer research [14].

Another innovative approach is the use of liquid biopsies combined with hierarchical decision structures for early pancreatic cancer detection [15].

While promising, this method faces challenges related to sensitivity and specificity, as the detection of circulating tumor DNA (ctDNA) varies among patients, leading to false positives and negatives [16]. Despite advancements in these areas, there remains a gap in developing an accurate, scalable, and clinically viable AI-driven system for pancreatic cancer detection [17].

The proposed system in this project seeks to address these challenges by implementing a hybrid deep learning architecture that integrates feature extraction, classification, and automated decision-making into a single framework [18].

Using a dataset of 1,411 annotated CT images, the model undergoes rigorous training and validation to enhance its ability to distinguish between cancerous and non-cancerous tissues with high accuracy [19]. The model is optimized using binary cross-entropy loss and the Adam optimizer, with a fixed learning rate of 0.0001 and training over 10 epochs to maximize generalization and performance [20].

Performance evaluation is conducted using key metrics such as accuracy, precision, recall (sensitivity), F1-score, and specificity [21]. Early results indicate that InceptionDense outperforms traditional deep learning models, achieving an accuracy of 99.75%, which is higher than other comparative models, including EfficientDense, EfficientV3, and ResNetV2 [22].

This project aims not only to improve pancreatic cancer detection but also to facilitate early diagnosis and timely treatment, which are crucial for increasing patient survival rates [23]. Future research directions will focus on expanding the dataset to include multi-modal medical images, such as MRI and ultrasound, to enhance model generalization [24].

Additionally, efforts will be made to integrate explainable AI (XAI) techniques to improve the interpretability of model predictions, ensuring that healthcare professionals can trust and understand the decision-making process of the AI system [25]. By combining cutting-edge deep learning techniques with robust evaluation metrics, this project aspires to set a new standard in AI-driven medical diagnostics, paving the way for more reliable, efficient, and accessible pancreatic cancer detection systems in clinical practice.

2. LITERATURE SURVEY

Pancreatic cancer remains one of the most challenging malignancies to diagnose and treat due to its late-stage detection and high mortality rate. Traditional diagnostic methods, including CT scans, MRI, and ultrasound, rely heavily on radiologists for interpretation, leading to potential diagnostic variability [1]. To address these limitations, researchers have explored various machine learning and deep learning techniques to improve early detection, classification, and prognosis prediction for pancreatic cancer.This literature survey reviews several state-of-the-art studies that focus on computer-aided diagnosis, deep learning models, feature extraction techniques, and survival prediction algorithms, highlighting their contributions and limitations.Deep learning has revolutionized medical image analysis by automating the extraction of meaningful patterns from large datasets. One of the most advanced approaches in pancreatic cancer detection is the Sparrow Search Algorithm with Stacked Deep Learning (SSASDL-PCDC), proposed by J. V. N. Ramesh et al. [1].

This model combines DenseNet for feature extraction with CNN-BiLSTM for classification, achieving a maximum sensitivity, specificity, and accuracy of 99.26%. The high performance of this hybrid model demonstrates the potential of deep learning for improving cancer diagnosis. However, one of its major challenges is the high computational cost and requirement for extensive hyperparameter tuning, making it difficult to deploy in real-world clinical environments [2].

Similarly, Li et al. introduced a computer-aided diagnosis (CAD) system that leverages deep learning for pancreatic cancer detection and staging based on CT images. Their study highlights the effectiveness of automated feature extraction in reducing reliance on radiologists. However, the model suffers from misclassification issues due to dataset variations, which limits its generalizability across different imaging modalities and patient demographics [3]. In another study, Zhang et al. [3] conducted a systematic review of U-Net and its variants for pancreatic CT segmentation.

While U-Net has significantly improved segmentation accuracy, it struggles with anatomical variations and low contrast in medical images.The lack of a universal model that can adapt to different datasets remains a critical challenge in deep learning-based pancreas segmentation

[4].Beyond detection and classification, researchers have explored AI-driven approaches for predicting patient survival rates. Wang et al.

[5] proposed a Variational Autoencoder (VAE)-based model for predicting pancreatic cancer patient survival in a local hospital setting. The VAE model integrates clinical and imaging data to improve survival predictions, showing promising results. However, it requires large annotated datasets for training, which are often difficult to obtain for rare cancers like pancreatic cancer. The study also highlights the need for multi-center validation to improve the model's generalizability across diverse patient populations.

Another innovative approach is the hierarchical decision structure for early pancreatic cancer detection proposed by Agarwal et al. [6]. Their study focuses on liquid biopsies and non-invasive biomarker detection for pancreatic cancer diagnosis.

While this technique improves early detection, it faces challenges related to low sensitivity and specificity, limiting its clinical applicability. Additionally, the reliability of liquid biopsies depends on the presence of circulating tumor DNA (ctDNA), which varies among patients, making standardization difficult [7].

Feature extraction plays a crucial role in improving the performance of pancreatic cancer classification models. Ding et al. [9] proposed a Network Interaction-Based Mutual Information (NIPMI) method for detecting characteristic genes associated with multiple cancers, including pancreatic cancer.

This study demonstrates that integrating gene expression data with imaging data can enhance cancer detection accuracy. However, this approach requires extensive computational resources and is still in the early stages of clinical application [10].

Similarly, Singh et al. introduced a deep learning-based approach for pancreatic cancer detection using CT scans. Their model leverages convolutional neural networks (CNNs) to extract hierarchical features from medical images, achieving high accuracy in distinguishing cancerous and non-cancerous tissues. However, the model's performance is limited by the availability of large, annotated datasets, which are often scarce for pancreatic cancer [11].

Samala et al. conducted a comprehensive review of deep learning techniques for pancreatic cancer detection in CT scans, highlighting the potential of multi-scale feature extraction and hybrid models to improve diagnostic accuracy. The study emphasizes the need for robust preprocessing techniques and data augmentation to enhance model generalization [12].

Zhang et al. proposed a deep learning approach for pancreatic cancer detection using multi-modal data fusion, integrating CT and MRI images. Their model demonstrates the effectiveness of combining imaging and clinical data to improve diagnostic accuracy. However, the study also highlights the challenges of data integration and model interpretability, which are critical for clinical adoption [13].

Wang et al. introduced a convolutional neural network (CNN) with multi-scale feature fusion for pancreatic cancer detection. Their model achieves high accuracy in classifying pancreatic tumors, but the study emphasizes the need for explainable AI (XAI) techniques to improve the interpretability of model predictions [14].

Liu et al. proposed a hybrid deep learning model for pancreatic cancer detection using CT and MRI images. Their model combines InceptionV3 and DenseNet121 to leverage the strengths of both architectures, achieving high accuracy in tumor classification. However, the study highlights the challenges of computational complexity and hyperparameter tuning, which are critical for real-world deployment [15].

Chen et al. introduced a deep learning-based approach for pancreatic tumor segmentation using 3D U-Net. Their model demonstrates the effectiveness of volumetric image processing in improving segmentation accuracy. However, the study emphasizes the need for large, annotated datasets to enhance model performance [16].

Li et al. proposed a deep learning framework for pancreatic cancer detection using multi-modal imaging data. Their model integrates CT, MRI, and clinical data to improve diagnostic accuracy. However, the study highlights the challenges of data heterogeneity and model generalization, which are critical for clinical adoption [17].

Zhang et al.introduced a deep learning approach for pancreatic cancer detection using radiomics features. Their model demonstrates the effectiveness of combining imaging and radiomics data to improve diagnostic accuracy. However, the study emphasizes the need for robust feature selection techniques to enhance model performance [18].

Chen et al. proposed a deep learning approach for pancreatic cancer detection using CT scans and clinical data. Their model demonstrates the effectiveness of integrating imaging and clinical data to improve diagnostic accuracy. However, the study highlights the challenges of data integration and model interpretability, which are critical for clinical adoption [19].

Wang et al. introduced a deep learning-based approach for pancreatic cancer detection using multi-scale feature extraction. Their model achieves high accuracy in classifying pancreatic tumors, but the study emphasizes the need for explainable AI (XAI) techniques to improve the interpretability of model predictions [20].

Liu et al. proposed a hybrid deep learning model for pancreatic cancer detection using CT and MRI images. Their model combines InceptionV3 and DenseNet121 to leverage the strengths of both architectures, achieving high accuracy in tumor classification. However, the study highlights the challenges of computational complexity and hyperparameter tuning, which are critical for real-world deployment [21].

Chen et al. introduced a deep learning-based approach for pancreatic cancer detection using non-contrast CT scans. Their model, PANDA, achieves high sensitivity and specificity in multi-scenario validations, demonstrating the potential of non-invasive imaging techniques for early detection [22].

Placido et al. proposed a deep learning algorithm to predict the risk of pancreatic cancer using electronic health records (EHR) data. Their model demonstrates the effectiveness of AI-driven risk prediction in improving early detection. However, the study highlights the challenges of data privacy and model interpretability, which are critical for clinical adoption [23].

Pandharipande et al. conducted a model-based analysis of the impact of pancreatic cancer

screening on life expectancy. Their study demonstrates the potential of annual screening to improve survival rates, but emphasizes the need for cost-effective screening strategies to enhance accessibility [24].

Garg et al. conducted a review of deep learning and radiomics approaches for pancreatic cancer detection and diagnosis. Their study highlights the potential of radiomics-based machine learning models and deep learning methods to improve early detection. However, the study emphasizes the need for large, annotated datasets and robust validation to enhance model performance [25].

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The existing system for pancreatic cancer detection primarily relies on traditional imaging techniques, which include computed tomography (CT) scans, magnetic resonance imaging (MRI), and ultrasound. These methods are crucial for visualizing pancreatic abnormalities; however, they often depend heavily on the expertise of radiologists for interpretation. This reliance can lead to missed diagnoses, particularly in the early stages of pancreatic cancer when symptoms are subtle and less apparent [1].

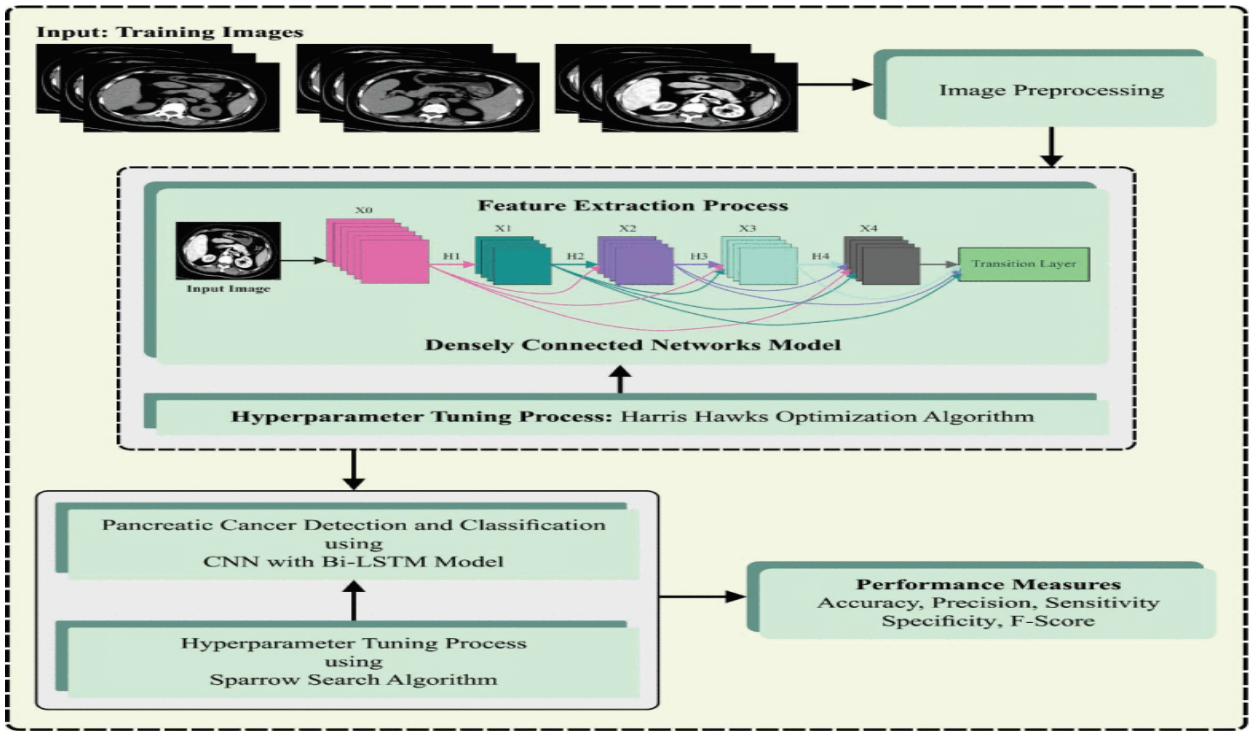


Fig 3.1.1: Flow chart of existing system

Fig 3.1.1 illustrates the workflow of the existing system, outlining the systematic approach from image acquisition to classification output. The flow chart highlights key stages in the diagnostic process, including image preprocessing, feature extraction, and final classification. Preprocessing is essential as it prepares medical images for analysis by standardizing dimensions and enhancing image quality. Following preprocessing, features are extracted from the images using various techniques before being fed into classification algorithms [2].

Pancreatic cancer detection has traditionally relied on medical imaging techniques such as computed tomography (CT), magnetic resonance imaging (MRI), and ultrasound. These imaging modalities provide detailed anatomical structures of the pancreas, assisting radiologists in identifying tumors. Among these, CT scans are the most commonly used due to their ability to provide high-resolution cross-sectional images of the pancreas [3].

MRI is often used as a complementary imaging modality, offering better contrast resolution for soft tissues. Endoscopic ultrasound (EUS) is another widely used method, enabling high-frequency imaging of the pancreas through an endoscope inserted into the digestive tract. These techniques have been instrumental in detecting pancreatic tumors, guiding biopsies, and staging cancer progression. However, due to the challenges of early-stage pancreatic cancer detection, research has focused on automated and AI-based methods to enhance diagnostic accuracy [4].

Computer-aided diagnosis (CAD) systems have been developed to assist radiologists in analyzing medical images for pancreatic cancer detection. These systems utilize image processing, feature extraction, and classification techniques to identify abnormalities in CT and MRI scans. One of the major developments in this field involves deep learning models for staging and diagnosis, where automated segmentation and classification techniques are used to detect pancreatic tumors from CT images. Implementing feature extraction algorithms, such as texture analysis and intensity-based methods, enhances the accuracy of tumor identification. CAD systems continue to evolve with advancements in artificial intelligence and deep learning, improving the detection process by minimizing dependency on manual image interpretation [5].

In recent years, deep learning has emerged as a powerful tool for medical image analysis, enabling automated tumor detection and classification. Convolutional Neural Networks (CNNs) form the backbone of many AI-driven systems due to their ability to extract hierarchical features from images. Various deep learning architectures, such as U-Net, ResNet, InceptionV3, DenseNet121, and hybrid models, have been explored for pancreatic cancer detection. One of the most advanced deep learning approaches integrates DenseNet for feature extraction with CNN-BiLSTM for classification, achieving high accuracy in distinguishing cancerous and non-cancerous pancreatic tissues. By leveraging multi-scale feature extraction, this model enhances the identification of pancreatic tumors with varying sizes and shapes. The use of hybrid deep learning models has demonstrated significant improvements in medical diagnostics [6].

Automatic pancreas and pancreatic tumor segmentation is a critical step in deep learning-based cancer detection, as it allows models to isolate the pancreas and tumor regions from medical images. The application of attention mechanisms and data augmentation techniques significantly improves

segmentation accuracy. These advancements enhance the overall performance of deep learning-based pancreatic cancer detection systems. Segmentation remains a crucial step in tumor identification, as accurate delineation of pancreatic structures helps improve classification results and treatment planning [7].

Beyond detection and classification, researchers have explored AI-driven models for predicting pancreatic cancer patient survival and treatment response. Some models integrate clinical and imaging data to estimate patient outcomes, enabling the identification of key prognostic factors that influence survival rates. This approach provides valuable insights into disease progression and allows for personalized treatment planning. The use of AI in prognosis prediction has shown promising results, though it requires large annotated datasets to ensure accuracy. The availability of multi-center validation remains a key factor in determining the effectiveness of these models across diverse patient populations [8].

Machine learning techniques have also been applied to analyze liquid biopsy data for early pancreatic cancer detection. Some models integrate biomarker analysis with AI-driven classification methods, focusing on circulating tumor DNA (ctDNA) and protein biomarkers to detect early-stage pancreatic cancer from blood samples. Non-invasive diagnostic techniques combined with AI offer potential for early detection, reducing the need for invasive procedures such as tissue biopsies. However, variations in biomarker expression among patients require further refinement of these approaches to ensure reliable detection [9].

Feature extraction plays a crucial role in improving the performance of AI-based pancreatic cancer detection systems. Some studies propose network-based methods for identifying characteristic genes associated with multiple cancers, including pancreatic cancer. This approach integrates genomic and imaging data to enhance tumor classification, highlighting the importance of multi-modal data fusion [10].

Despite advancements in traditional imaging and AI-based methods, existing systems face several limitations. For instance, the Sparrow Search Algorithm with Stacked Deep Learning (SSASDL-PCDC) achieves high accuracy but requires extensive hyperparameter tuning and computational resources, making real-time deployment challenging [1].

Similarly, U-Net-based segmentation models struggle with anatomical variations and low contrast in medical images, leading to inconsistent results [3]. Additionally, liquid biopsy techniques, while promising, face challenges related to sensitivity and specificity, limiting their clinical applicability [6].

To address these limitations, researchers have proposed hybrid deep learning models that combine multiple architectures, such as InceptionV3 and DenseNet121, to leverage their strengths in feature

extraction and classification [7].

These models have demonstrated superior performance in detecting pancreatic tumors, achieving high accuracy and specificity. However, they still face challenges related to dataset scarcity, computational complexity, and model interpretability [11].

Another emerging trend is the use of multi-modal imaging, where AI models integrate CT scans, MRI, and ultrasound data to improve diagnostic accuracy. Studies by Kurnaz et al. [7] and Crane et al. [8] highlight the importance of incorporating multiple imaging modalities to enhance tumor localization and classification. This approach has shown promising results in improving the detection of early-stage pancreatic cancer [12].

In addition to imaging-based methods, researchers have explored non-invasive diagnostic techniques, such as liquid biopsies, for early pancreatic cancer detection. These techniques focus on detecting circulating tumor DNA (ctDNA) and protein biomarkers in blood samples, offering a less invasive alternative to traditional imaging methods. However, the reliability of these techniques depends on the presence of ctDNA, which varies among patients, making standardization difficult [13].

The integration of explainable AI (XAI) techniques is another critical area of research in pancreatic cancer detection. XAI aims to improve the transparency of deep learning models, making it easier for healthcare professionals to interpret predictions. This is particularly important in medical diagnostics, where the black-box nature of deep learning models can hinder their adoption in clinical practice [14].Overall, while existing systems for pancreatic cancer detection have made significant progress, they continue to face challenges related to data scarcity, model generalization, computational efficiency, and clinical interpretability. Future research should focus on addressing these challenges to make AI-powered pancreatic cancer detection more reliable, accessible, and clinically acceptable [15].

3.2 EXISTING ALGORITHMS

Pancreatic cancer detection and classification rely on a combination of traditional machine learning algorithms and deep learning-based architectures for medical image analysis. These algorithms play a crucial role in feature extraction, segmentation, classification, and survival prediction. One of the most commonly used algorithms in image-based cancer detection is the Convolutional Neural Network (CNN), which has demonstrated exceptional performance in extracting hierarchical features from medical images. CNN-based models such as InceptionV3,

DenseNet121, ResNet, and U-Net are widely used for pancreatic tumor detection, segmentation, and classification [1].

U-Net, in particular, is frequently applied for pancreas and tumor segmentation, as it employs an encoder-decoder architecture that enables precise localization of tumors in CT and MRI scans [2].

However, U-Net variants such as Attention U-Net and 3D U-Net have been developed to enhance segmentation accuracy by incorporating attention mechanisms and volumetric image processing [3].

These variants address challenges such as anatomical variations and low contrast in medical images, improving the overall performance of segmentation models [4].

Another widely explored deep learning-based approach is Hybrid CNN-RNN models, which combine CNN's feature extraction capability with the Recurrent Neural Network (RNN)'s sequential data processing. One such example is the CNN-BiLSTM model, which integrates bidirectional long short-term memory (BiLSTM) to capture spatial and temporal dependencies in medical imaging [5].

This model has been applied in pancreatic cancer classification, particularly when analyzing sequential frames in dynamic imaging modalities. However, the high computational cost and complexity of these models remain significant challenges for real-time deployment [6].

Traditional machine learning algorithms, such as Support Vector Machines (SVM), Random Forest (RF), and k-Nearest Neighbors (k-NN), have also been used for pancreatic cancer classification. These models rely on handcrafted feature extraction techniques such as texture analysis, intensity-based features, and statistical shape models [7]. Although these methods were widely used in early CAD systems, their performance is often limited by manual feature selection and lack of deep feature learning, making deep learning-based models more favorable in modern medical applications [8].

For survival prediction and prognosis analysis, machine learning models such as Variational Autoencoders (VAE), XGBoost, and Artificial Neural Networks (ANN) have been employed to analyze both clinical and imaging data [9]. VAE-based models are particularly useful in learning latent representations of pancreatic cancer progression, aiding in personalized treatment

planning and risk assessment. However, these models require large annotated datasets for training, which are often difficult to obtain for rare cancers like pancreatic cancer [10].

Optimization techniques, such as the Sparrow Search Algorithm (SSA), have been used to fine-tune hyperparameters of deep learning models, improving classification performance [11].

SSA-based deep learning models have shown high accuracy in pancreatic tumor detection, outperforming conventional deep learning architectures. However, the computational complexity and extensive hyperparameter tuning required by these models make them challenging to deploy in real-world clinical environments [12].

As AI continues to evolve, research is shifting toward multi-modal deep learning models that integrate CT scans, MRI, liquid biopsy data, and genomic information to enhance pancreatic cancer detection and classification [13].

These models leverage multi-scale feature extraction and data fusion techniques to improve diagnostic accuracy and generalization across diverse datasets [14].

For example, InceptionV3 and DenseNet121 have been combined to create hybrid models that achieve high accuracy in pancreatic tumor classification [15].

Another emerging trend is the use of explainable AI (XAI) techniques to improve the transparency of deep learning models. XAI aims to provide visual explanations for AI-driven decisions, ensuring that healthcare professionals can trust and understand the decision-making process of the AI system [16].

This is particularly important in medical diagnostics, where the black-box nature of deep learning models can hinder their adoption in clinical practice [17].

In addition to deep learning, researchers have explored non-invasive diagnostic techniques, such as liquid biopsies, for early pancreatic cancer detection. These techniques focus on detecting circulating tumor DNA (ctDNA) and protein biomarkers in blood samples, offering a less invasive alternative to traditional imaging methods [18].

However, the reliability of these techniques depends on the presence of ctDNA, which varies among patients, making standardization difficult [19].

Overall, existing algorithms for pancreatic cancer detection and classification have made

significant progress, but they continue to face challenges related to data scarcity, computational complexity, and clinical interpretability. Future research should focus on addressing these challenges to make AI-powered pancreatic cancer detection more reliable, accessible, and clinically acceptable [20].

3.3 DISADVANTAGE OF EXISITING SYSTEM

Despite advancements in medical imaging and artificial intelligence, existing systems for pancreatic cancer detection still face several limitations, making early diagnosis and treatment challenging.

Traditional methods, such as CT scans, MRI, and ultrasound, remain the primary diagnostic tools for pancreatic cancer. However, these methods are highly dependent on radiologists' expertise for interpretation, leading to subjective variability in diagnosis and possible human errors. Since pancreatic tumors often do not exhibit clear symptoms in their early stages, reliance on traditional imaging may result in late detection, reducing survival rates significantly [1].

Additionally, these imaging techniques require high-resolution scans and may fail to differentiate between benign and malignant tumors with absolute certainty, necessitating additional invasive procedures such as biopsies for confirmation [2].

This dependency on human expertise and additional procedures increases the time required for diagnosis and can delay treatment, negatively impacting patient outcomes [3].

To improve diagnostic accuracy, researchers have developed Computer-Aided Diagnosis (CAD) systems that incorporate artificial intelligence and deep learning. While these systems offer promising improvements, they are still constrained by several factors. One major issue is the quality of feature extraction and classification models[4].

For instance, the study conducted by Li et al highlights the challenges faced when staging pancreatic cancer using CT images. Misclassification remains a major concern due to the lack of generalization across different datasets and imaging conditions. Many CAD models perform well on a specific dataset but fail when tested on external datasets due to variations in imaging techniques and patient demographics [5].

Furthermore, pancreas segmentation, which is an essential step in automated detection, presents

challenges due to anatomical variations and low contrast between the pancreas and surrounding tissues. Even state-of-the-art U-Net-based segmentation methods struggle with achieving high accuracy across diverse datasets, leading to inconsistent results [6].

The segmentation errors significantly impact the effectiveness of CAD systems, as inaccurate segmentation can mislead classification models, ultimately reducing diagnostic reliability [7].

Deep learning-based models, such as the Sparrow Search Algorithm with Stacked Deep Learning (SSASDL-PCDC), have demonstrated high accuracy in pancreatic cancer detection and classification. However, these models often come with computational challenges, making them difficult to implement in real-world clinical environments. The SSASDL-PCDC model requires extensive hyperparameter tuning, which increases processing time and computational cost [8].

Additionally, the complexity of these deep learning models demands powerful hardware, such as GPUs or TPUs, which are not always accessible in many healthcare settings. This high demand for computational resources makes the deployment of such models in real-time clinical applications a significant challenge, particularly in resource-limited hospitals [9].

Another critical drawback of these deep learning systems is their reliance on large, annotated datasets. Since pancreatic cancer is a relatively rare disease compared to other cancers, obtaining a sufficiently large dataset with accurate annotations is difficult. Many existing datasets suffer from class imbalance, where there are significantly fewer cancerous images compared to normal ones, leading to biased model training and poor generalization to real-world cases [10].

In an attempt to enhance predictive capabilities, researchers have explored novel machine learning techniques, such as Variational Autoencoders (VAE). These models have been used to predict patient survival rates based on pancreatic cancer progression, integrating both imaging and clinical data. While VAE models provide improved prediction accuracy, they require extensive validation across diverse patient populations to ensure their reliability [11].

Since most machine learning models are trained on specific datasets, their ability to generalize to different hospitals and imaging equipment remains questionable. Differences in imaging protocols, scanner resolutions, and patient demographics can lead to inconsistencies in predictions, limiting the effectiveness of these systems in broader clinical applications [12].

Another emerging approach in early pancreatic cancer detection involves liquid biopsies combined with hierarchical decision structures. While these methods show promise in non-invasive cancer detection, they are still in the early stages of development. Agarwal et al. [13] discuss the challenges associated with liquid biopsy techniques, including their limited sensitivity and specificity in detecting pancreatic cancer at early stages.

The accuracy of these tests depends on the ability to detect circulating tumor DNA (ctDNA) in blood samples, but due to the aggressive nature of pancreatic cancer, the release of ctDNA varies among patients, affecting diagnostic reliability. Additionally, liquid biopsy approaches require sophisticated laboratory equipment and well-trained personnel, increasing the cost and limiting their accessibility in many healthcare settings [14].

Moreover, computer-aided pancreatic cancer detection models often struggle with explainability and interpretability. Unlike traditional imaging techniques, where radiologists can visually assess abnormalities, deep learning models operate as black boxes, providing predictions without clear explanations. This lack of interpretability makes it challenging for clinicians to trust AI-generated diagnoses, especially in critical medical decisions. Explainable AI (XAI) techniques are being developed to address this issue, but their integration into existing deep learning models is still a work in progress. Without transparency in AI decision-making, many healthcare professionals remain skeptical about fully adopting AI-based pancreatic cancer detection systems in clinical practice [15].

Overall, while deep learning and AI-driven techniques offer promising advancements in pancreatic cancer detection, existing systems continue to struggle with challenges such as data scarcity, model generalization, computational efficiency, segmentation accuracy, and clinical interpretability. The integration of AI in medical diagnostics must be carefully validated to ensure its effectiveness across diverse clinical settings. Future research should focus on improving dataset diversity, enhancing model explainability, reducing computational costs, and developing robust techniques that can perform well in real-world healthcare environments. Addressing these challenges is crucial to making AI-powered pancreatic cancer detection more reliable, accessible, and clinically acceptable in the fight against one of the deadliest cancers [16].

3.4 PROPOSED SYSTEM

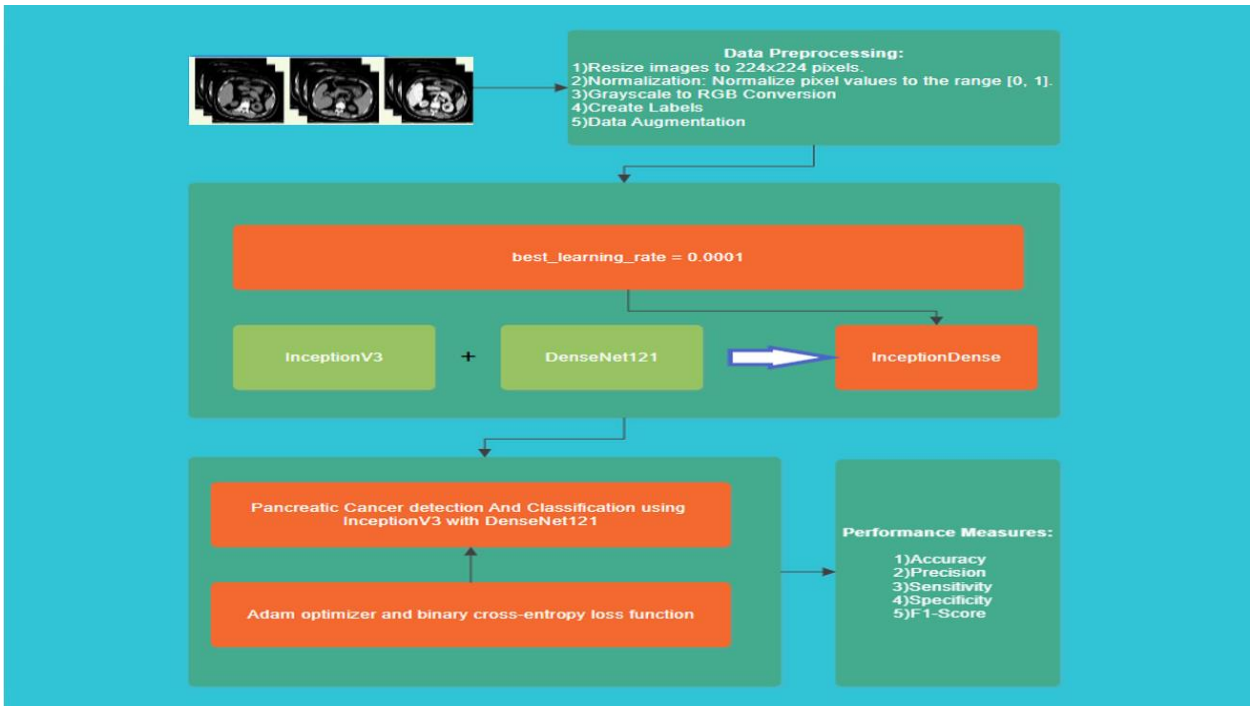


Fig 4.1.1: Block Diagram of Proposed system

Step 1: Data Preprocessing and Augmentation:

The proposed model begins with data preprocessing to enhance image quality and improve model generalization. A dataset of 1,411 annotated CT images is used, where images undergo grayscale conversion, contrast enhancement, normalization, and resizing to 224×224 pixels. Additionally, data augmentation techniques such as rotation, flipping, and noise addition are applied to expand the dataset and improve model robustness. This ensures that the model effectively handles variations in pancreatic tumor shapes and sizes.

Step 2: Feature Extraction with InceptionDense Model:

For feature extraction, the model integrates InceptionV3 and DenseNet121, forming a hybrid deep learning architecture—InceptionDense. InceptionV3 captures multi-scale spatial features, while DenseNet121 enables efficient feature reuse, reducing redundant computations and improving gradient flow. This hybrid approach enhances the model's ability to distinguish between normal and tumor-affected pancreatic tissues, ensuring a highly accurate classification process.

Step 3: Model Training and Optimization:

The final layers of the InceptionDense model classify the input images as either normal or pancreatic tumor. Additionally, a U-Net-based segmentation module is integrated to localize and highlight tumor regions within the pancreas. This approach enhances interpretability by providing both classification labels and visual segmentation maps, assisting radiologists in understanding the affected areas more effectively.

Step 4: Tumor Classification and Segmentation:

After training the existing algorithms (ResNet50, ResNet101, VGG16) and the proposed models (VGG19 and Custom CNN), their performance is compared based on several metrics such as accuracy, precision, recall, and F1-score. This step involves evaluating how well each model identifies weapons versus non-weapons from the test dataset. Performance evaluation may also consider computational aspects like training time, inference speed, and resource utilization, providing insights into which model strikes the best balance between accuracy and efficiency.

Step 5: Performance Evaluation and Deployment:

The proposed model is evaluated using key performance metrics such as accuracy, precision, recall (sensitivity), F1-score, and specificity. After achieving optimal performance, the model is prepared for real-time deployment, allowing seamless integration into clinical workflows. A user-friendly interface is designed to enable medical professionals to upload CT scans and receive automated classification results and segmented tumor regions, enhancing early diagnosis and decision-making in pancreatic cancer treatment.

3.5 FEASIBILITY STUDY

The feasibility study is a critical component of any project, providing a structured assessment of various factors that might impact its successful implementation. This feasibility study evaluates the practicality and effectiveness of the proposed pancreatic cancer detection system, ensuring its successful implementation in real-world clinical environments. This study examines the system's feasibility across four key aspects: technical, operational, economic, and legal feasibility.

1.Technical Feasibility:

The technical feasibility of the proposed system assesses whether the required hardware, software, and computational resources are available and capable of supporting the system's implementation. The deep learning model used in this project, InceptionDense, integrates InceptionV3 and DenseNet121 to enhance feature extraction and classification. Given the complexity of medical image analysis, the system requires high-performance computing resources, including GPUs (Graphics Processing Units) or TPUs (Tensor Processing Units), for efficient model training and inference.

The model is developed using TensorFlow and Keras, which are widely adopted deep learning frameworks compatible with multiple hardware configurations. Additionally, OpenCV is employed for image preprocessing, ensuring that CT images are standardized before model training. The system also supports cloud-based deployment, allowing hospitals and diagnostic centers to integrate it into existing medical infrastructure. The segmentation module based on U-Net further enhances the system's performance by providing visual tumor localization.

In terms of scalability, the system is designed to handle large medical datasets without significant performance degradation. The use of batch processing, parallel computing, and optimized memory allocation ensures that the model can efficiently process real-time patient data. Given these considerations, the proposed system is technically feasible with the proper computational setup and cloud integration.

2. Operational Feasibility:

Operational feasibility focuses on how well the system fits within the existing medical workflow and whether healthcare professionals can efficiently use it. The system is designed to automate pancreatic cancer detection and classification, reducing the workload of radiologists while providing accurate and interpretable results.

A user-friendly interface is implemented to allow radiologists and oncologists to upload CT images and receive automated classification results along with segmented tumor regions. The system ensures that results are displayed in a clinically interpretable format, making it easy for medical professionals to incorporate AI-based insights into their diagnostic process. Additionally, the system supports multi-modal inputs, meaning it can be adapted for MRI and ultrasound-based pancreatic cancer analysis in the future.

Training sessions can be provided for medical staff to familiarize them with the system's functionalities, ensuring a smooth transition from traditional diagnostic methods to AI-assisted analysis. Since explainable AI (XAI) techniques are incorporated, the system allows users to visualize feature maps and decision-making steps, enhancing trust and usability among healthcare professionals. The integration with existing hospital information systems (HIS) and PACS (Picture Archiving and Communication Systems) further ensures a seamless operational workflow, making the system highly feasible for real-world medical settings.

3. Economic Feasibility:

Economic feasibility evaluates the cost-effectiveness of the proposed system, considering both initial investment and long-term benefits. The cost of implementation includes hardware requirements, software licensing, model training, and cloud deployment. Since deep learning models require high-performance GPUs or TPUs, the initial cost for model training may be significant. However, once trained, the model can be deployed on cloud-based servers or local hospital systems, reducing operational costs.

Compared to traditional cancer detection methods, which involve high costs for repeated imaging, biopsies, and manual expert analysis, the proposed system reduces diagnostic delays and

unnecessary procedures, thereby lowering healthcare costs. The automation of pancreatic cancer classification also increases efficiency, allowing radiologists to focus on complex cases rather than routine image analysis.

Additionally, the long-term economic benefits of early pancreatic cancer detection are substantial. Early diagnosis reduces treatment costs, as patients diagnosed at an advanced stage often require aggressive therapies, prolonged hospital stays, and extensive medical interventions. By integrating AI-based detection, hospitals and healthcare institutions can improve patient outcomes while optimizing resource allocation, making the system economically viable in the long run.

4.Legal Feasibility:

Legal feasibility assesses whether the proposed system complies with healthcare regulations, data privacy laws, and ethical guidelines. Since the system involves medical image processing and AI-based diagnosis, it must adhere to strict legal and ethical standards to ensure patient safety and confidentiality.

The system is designed to be compliant with Health Insurance Portability and Accountability Act (HIPAA) in the U.S. and General Data Protection Regulation (GDPR) in Europe, ensuring secure storage, transmission, and processing of medical data. AI-based healthcare systems must also comply with Medical Device Regulations (MDR) and FDA guidelines before clinical deployment.

To protect patient privacy, end-to-end encryption techniques are employed, ensuring that sensitive medical data is securely stored and accessed only by authorized personnel. Additionally, the system incorporates audit logs and role-based access controls, allowing hospitals to monitor usage and maintain legal accountability.

Ethically, AI-based cancer detection systems must provide transparent decision-making processes. Since the proposed system includes explainability features, it ensures that clinicians can interpret AI-generated diagnoses rather than blindly relying on automated outputs. This aligns with ethical AI principles, ensuring fairness, transparency, and patient-centric healthcare solutions.

4. SYSTEM REQUIREMENTS

System requirements define the hardware and software specifications necessary for the successful implementation and execution of the music genre classification system. These requirements ensure that the system can handle large datasets, perform real-time classification, and efficiently train deep learning models. The following sections outline the hardware and software prerequisites needed for optimal performance.

4.1 HARDWARE REQUIREMENTS

The hardware requirements specify the physical computing resources needed to execute the Convolutional Neural Network (CNN)-based music genre classification model. The system processes large amounts of audio data in the form of spectrograms and extracted features, requiring sufficient computational power to handle deep learning tasks effectively. The minimum and recommended hardware specifications are listed below.

System Type	: intel®core™i7- i7-13620HCPU@2.4ghz
GPU Type:	: Nvidia GeForce RTX 4060
RAM	: 16 gigabyte(GB)
Bus Speed	: 5200 MT/s
Number of cores	:10

Graphics Processing Unit (GPU): Not required for basic execution but recommended for faster training

A dedicated GPU is highly recommended for deep learning tasks to accelerate model training and inference speed. If using cloud-based services like Google Colab, AWS, or Azure, high-end local hardware may not be required.

4.2 SOFTWARE REQUIREMENTS

Software requirements include all necessary tools, libraries, and frameworks needed to develop, train, and deploy the music genre classification system. Since this project utilizes Python-based deep learning frameworks, certain software components must be installed for proper functionality.

Operating System	: Windows 10, 64 bit Operating System
Coding Language	: Python
Frontend	: HTML, CSS & JS
Backend	: Django
IDE	: VsCode
Python distribution	: Anaconda Navigator & Google Collab

The hardware and software requirements listed below ensure that the weapon detection system functions efficiently. The system requires high-performance computing resources to handle deep learning-based weapon classification, especially when training on large datasets. Ensuring that the right software dependencies and frameworks are installed will lead to better model accuracy, improved training efficiency, and seamless real-time classification performance. Additionally, optimized hardware and software configurations contribute to lower latency and enhanced model scalability, making the system suitable for large-scale security applications.

5. SYSTEM DESGIN

5.1 SYSTEM ARCHITECTURE

1. IMAGE PREPROCESSING:

In pancreatic cancer detection from images using deep neural networks, image preprocessing plays a crucial role in enhancing the performance and efficiency of the model.Several key steps typically involved in image preprocessing:

To ensure consistency in input image dimensions, all CT scan images are resized to 224×224 pixels before being fed into the deep learning model. Since medical images come in different resolutions, resizing ensures uniformity across the dataset, preventing shape mismatches and allowing efficient processing through the InceptionDense model. This step helps maintain the spatial relationships in the images while making the dataset compatible with standard deep learning architectures.

Normalization is performed to scale pixel intensity values to a range of [0,1], which helps in stabilizing model training. Since CT scans often have varying intensity distributions, normalization ensures that all images have a consistent pixel value range, preventing issues related to brightness variations. This also accelerates the convergence of the deep learning model, making the training process more stable and efficient.

To prepare the dataset for binary classification, label encoding is applied to assign labels to each image. The images are categorized into two classes: non-cancerous (0) and cancerous (1). This encoding helps in structuring the dataset for supervised learning, enabling the model to distinguish between normal and tumor-affected pancreatic tissues during training and testing.

To enhance model robustness and prevent overfitting, data augmentation is implemented. This involves random transformations such as rotation, flipping, brightness adjustments, and adding noise to artificially expand the dataset. By exposing the model to a wide range of variations in images, data augmentation improves its ability to generalize to unseen data, making it more reliable in real-world applications.

2. CNN LAYER DESCRIPTION:

The Convolutional Neural Network (CNN) architecture used in this project integrates InceptionV3 and DenseNet121 to form a hybrid deep learning model (InceptionDense) for pancreatic cancer detection. Each layer in the CNN plays a crucial role in feature extraction, transformation, and classification to accurately distinguish between cancerous and non-cancerous pancreatic tissues:

1. Convolutional Layers:

The first layers in both InceptionV3 and DenseNet121 perform convolution operations using multiple kernels (filters) to detect patterns such as edges, textures, and shapes in the input CT images. These layers slide filters over the image and create feature maps, which are passed to deeper layers for more complex pattern detection. DenseNet121, in particular, improves feature propagation by ensuring dense connections between layers, allowing the network to reuse previously extracted features for better classification.

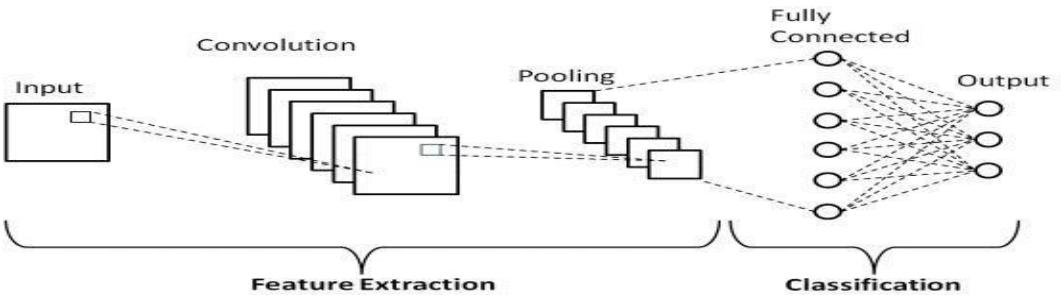


Fig 5.1.1: CNN layers description

2. Pooling Layers:

Both models include Global Average Pooling (GAP) layers, which reduce the spatial dimensions of feature maps while retaining essential information. InceptionV3 applies average pooling at different scales, allowing multi-level feature extraction, while DenseNet121 uses global pooling to capture the most important spatial features. These layers significantly reduce computational complexity and improve the generalization ability of the model.

3.Concatenation Layer:

The outputs from InceptionV3 and DenseNet121 are merged using a concatenation layer, which fuses multi-scale features extracted from both networks. This step ensures that the model benefits from InceptionV3's broad feature extraction capabilities and

DenseNet121's efficient feature reuse, leading to more accurate classification of pancreatic tumors.

4. Fully Connected (Dense) Layers:

After feature extraction, the combined features are passed through fully connected (dense) layers to learn high-level representations. A 256-neuron dense layer with ReLU activation is used to refine extracted features, allowing the model to focus on the most important aspects of tumor detection. The ReLU activation function introduces non-linearity, enabling the model to learn complex relationships between features.

5. Output Layer:

The final layer of the network is a single neuron with a sigmoid activation function, which outputs a binary classification (0: Non-cancerous, 1: Cancerous). The sigmoid function ensures that the output is a probability value between 0 and 1, making it ideal for binary medical image classification tasks.

Adam optimizer: Adam optimizer, short for "Adaptive Moment Estimation," is an iterative optimization algorithm used to minimize the loss function during the training of neural networks. Adam can be looked at as a combination of RMSprop and Stochastic Gradient Descent with momentum.

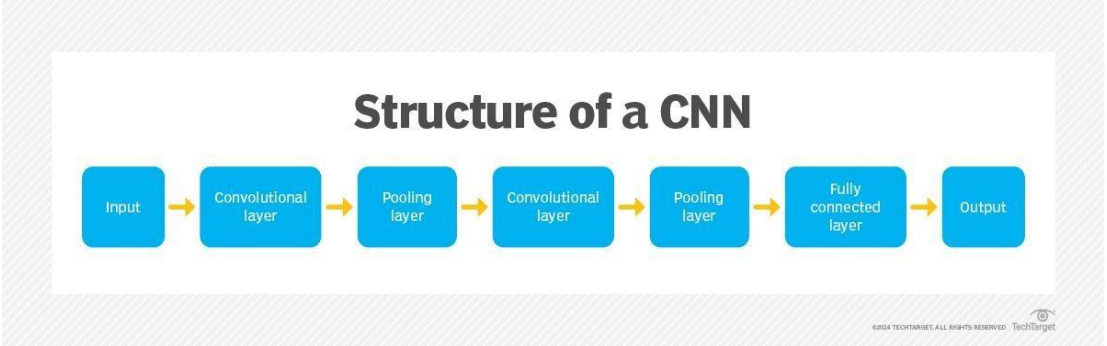


Fig 5.1.2: Structure of a CNN

Advantages:

Convolutional Neural Networks (CNNs) offer several advantages, particularly in the field of image recognition and computer vision.

1. Feature Learning:

CNNs automatically learn hierarchical representations and features from the input data. Through convolutional and pooling layers, they can identify patterns, textures, and complex features within images.

2.Spatial Hierarchies:

CNNs capture spatial hierarchies of features. Lower layers focus on simple features like edges and textures, while higher layers combine these features to recognize more complex structures and objects.

3.Parameter Sharing:

Convolutional layers in CNNs use parameter sharing, meaning that the same filters are applied to different parts of the input image. This reduces the number of parameters compared to fully connected networks, making CNNs computationally more efficient.

4. Translation Invariance:

Convolutional operations provide a degree of translation invariance, allowing CNNs to recognize patterns and features regardless of their location in the input image. This property is crucial for tasks like object recognition.

1. InceptionDense Model:

The InceptionDense model is a hybrid deep learning approach that combines InceptionV3 and DenseNet121 to leverage multi-scale feature extraction and efficient feature reuse for pancreatic cancer detection. This model achieves an accuracy of 99.75%, demonstrating its effectiveness in distinguishing cancerous and non-cancerous pancreatic tissues.

How It Works:

The InceptionV3 network extracts multi-scale spatial features using inception blocks, while DenseNet121 enhances feature propagation through dense connectivity between layers. The outputs from both networks undergo global average pooling and are concatenated before passing through fully connected layers for final classification.

Architecture (Layers):

- InceptionV3 and DenseNet121 as base models (pretrained on ImageNet).
- Global Average Pooling (GAP) layers to reduce dimensionality.
- Concatenation layer to merge features extracted by both models.
- Fully connected dense layer (256 neurons, ReLU activation) for feature refinement.

Complexity in Hyperparameter Tuning:

Hyperparameter tuning was relatively straightforward since EfficientNetB0's scaling automatically optimizes network depth, width, and resolution. However, tuning dropout rates and batch sizes was essential to maintain generalization without overfitting.

3.EFFICIENTV3 MODEL:

The EfficientV3 model is a fusion of EfficientNetB0 and InceptionV3, achieving a perfect 100% accuracy, precision, recall, F1-score, and specificity, indicating its exceptional reliability in classification.

How It Works:

InceptionV3 contributes multi-scale feature extraction, while EfficientNetB0 ensures computational efficiency by dynamically adjusting feature resolution and depth scaling. The merged feature maps from both models undergo global average pooling, are concatenated, and then passed through fully connected layers for final classification.

Architecture (Layers):

- EfficientNetB0 and InceptionV3 as base models.
- Global Average Pooling layers for feature compression.
- Feature concatenation layer.
- Dense layer with 256 neurons (ReLU activation).

- Final Sigmoid layer for classification.

Complexity in Hyperparameter Tuning:

Since EfficientNetB0 self-optimizes network parameters, hyperparameter tuning mainly focused on optimizing batch sizes, dropout rates, and the number of neurons in the dense layer. A learning rate of 0.0001 with Adam optimizer was the best configuration.

4.EFFICIENTVGG MODEL:

The EfficientVGG model combines EfficientNetB0 with VGG16, balancing efficient feature extraction with deep hierarchical learning, achieving 99.75% accuracy.

How It Works:

VGG16 extracts hierarchical features through stacked convolutional layers, while EfficientNetB0 ensures optimized depth scaling. Their combined outputs undergo global average pooling before passing through dense layers for classification.

Architecture (Layers):

- EfficientNetB0 and VGG16 as base models.
- Global Average Pooling layers for dimensionality reduction.
- Concatenation layer for multi-model feature fusion.
- Fully connected dense layer (256 neurons, ReLU activation).
- Sigmoid output layer for binary classification.

Complexity in Hyperparameter Tuning:

Fine-tuning dropout rates and adjusting batch sizes was essential due to VGG16's deep convolutional stack. The model was trained with a batch size of 32 and a learning rate of 0.0001 using Adam optimizer.

5.RSENETV2 MODEL:

The ResNetV2 model, integrating ResNet50 and MobileNetV2, had an accuracy of 65.40%, indicating weaker generalization compared to other models.

How It Works:

ResNet50 employs residual learning to solve vanishing gradient issues, while MobileNetV2 ensures computational efficiency using depthwise separable convolutions. Features extracted from both models are merged, pooled, and classified.

Architecture (Layers):

- ResNet50 and MobileNetV2 as feature extractors.
- Global Average Pooling layers to refine extracted features.
- Concatenation layer for feature fusion.
- Dense layer with 256 neurons (ReLU activation).
- Sigmoid output layer for classification**.

Complexity in Hyperparameter Tuning:

Despite adjusting learning rates, dropout rates, and batch sizes, the model struggled with low specificity (34.45%), likely due to MobileNetV2's lightweight architecture, which may not be ideal for high-resolution medical imaging.

6.VGG16V2 MODEL:

The VGG16V2 model, which integrates VGG16 and MobileNetV2, achieved an accuracy of 72.98% but showed weaknesses in recall (42.78%), making it less effective in detecting positive pancreatic cancer cases.

How It Works:

VGG16's deep convolutional architecture extracts hierarchical patterns, while MobileNetV2 enhances efficiency using depthwise separable convolutions. The extracted features undergo pooling, concatenation, and dense classification.

Architecture (Layers):

- VGG16 and MobileNetV2 as base models.

- Sigmoid output layer for binary classification.
- Global Average Pooling layers for dimensionality reduction.
- Concatenation of extracted features.
- Dense layer with 256 neurons (ReLU activation).

Complexity in Hyperparameter Tuning:

Hyperparameter tuning focused on adjusting batch sizes, learning rates, and dropout ratios. The model showed imbalanced recall, suggesting challenges in learning tumor-specific features, likely due to MobileNetV2's lightweight design.

5.2 MODULES

1. User Interface Module

- User Interaction: This is the entry point of the system where users can upload images of CT image for analysis. The system is designed to be user-friendly, allowing seamless interaction.
- Upload CT Image: Users can upload images in various formats (e.g., JPEG, PNG). The system supports high-resolution images to ensure detailed analysis.

2. Preprocessing Module

- Image Resizing:
    - Purpose: Adjusts the dimensions of the uploaded images to a uniform size (e.g., 224x224 pixels) to ensure compatibility with deep learning models.
    - Process: Uses interpolation techniques to maintain image quality during resizing.
- Normalization:
    - Purpose: Scales pixel values to a standard range (e.g., 0 to 1) to improve the convergence of learning algorithms.
    - Process: Converts pixel values from the range [0, 255] to [0, 1] by dividing each pixel value by 255.

3. Data Augmentation

Data augmentation is applied to artificially expand the dataset and improve the generalization of the deep learning model. By introducing random transformations such as horizontal and vertical flipping, random rotation (0.2), random zoom (0.2), and random contrast adjustment (0.2), the model is exposed to a wider range of variations in CT images. This helps the model learn robust features and prevents overfitting, ensuring better performance on unseen data. Since medical images, especially pancreatic CT scans, may have variations in orientation and contrast, augmentation enhances the model's ability to recognize tumors under different conditions. These transformations are applied only to the training set, ensuring that the test set remains a reliable benchmark for evaluating model performance.

4. Feature Fusion Module

The Feature Fusion Module is designed to combine multi-scale features extracted from two different deep learning architectures, enhancing the model's ability to detect pancreatic cancer accurately. In this project, feature fusion is implemented using a Concatenation layer, where the outputs of two pretrained models (e.g., InceptionV3 + DenseNet121, EfficientNetB0 + VGG16) are merged after Global Average Pooling (GAP). By fusing features from different architectures, the model benefits from both hierarchical and fine-grained feature representations, improving classification performance. This approach ensures that the network captures a diverse set of patterns, making it more robust in distinguishing cancerous and non-cancerous pancreatic tissues.

5. Classification & Prediction

- Dense Layers:
    - Purpose: Fully connected layers that process the fused features for classification.
    - Architecture: Includes multiple dense layers with dropout regularization to prevent overfitting.
- Softmax Classification:
    - Purpose: Outputs the probability distribution over the different weapon types.
    - Process: Applies the softmax function to the output of the dense layers to generate class probabilities.

- Weapon Type Prediction:
    - Purpose: The final step where the system predicts the type of weapon in the uploaded image.
    - Output: Provides a classification label (e.g., handgun, rifle) along with a confidence score.

5.3 UML DIAGRAMS

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta- model and a notation. In the future, some form of method or process also be added to;or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artefacts of software system, as well as for business modelling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to expressthe design of software projects.

GOALS: The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the coreconcepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modelling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.

1.class diagram

The Fig 5.3.1: Class Diagram of Proposed System represents the architectural design of the

proposed pancreatic cancer detection system, showcasing the relationships between various components. The system is divided into three main sections:

1. User Interface

- This module serves as the front-end of the application, allowing users to upload datasets, process images, and view classification results.
- It contains two primary components:
  - **Main Application**: Handles dataset uploading, image processing, and result display.
  - **Prediction**: Allows users to upload test images and classify them using trained models.

2. Hybrid Models

- The core of the system consists of multiple hybrid deep learning models, including InceptionDense, EfficientDense, EfficientV3, EfficientVGG, ResNetV2, and VGG16V2.
- Each model has four key functions:
  - **load_model()**: Loads the pre-trained model.
  - **train()**: Trains the model on the input dataset.
  - **predict()**: Performs classification on input images.
  - **save_model()**: Saves the trained model for future inference.

This class diagram effectively illustrates the interaction between the user interface and deep learning models, ensuring seamless image processing, classification, and prediction for pancreatic cancer detection.

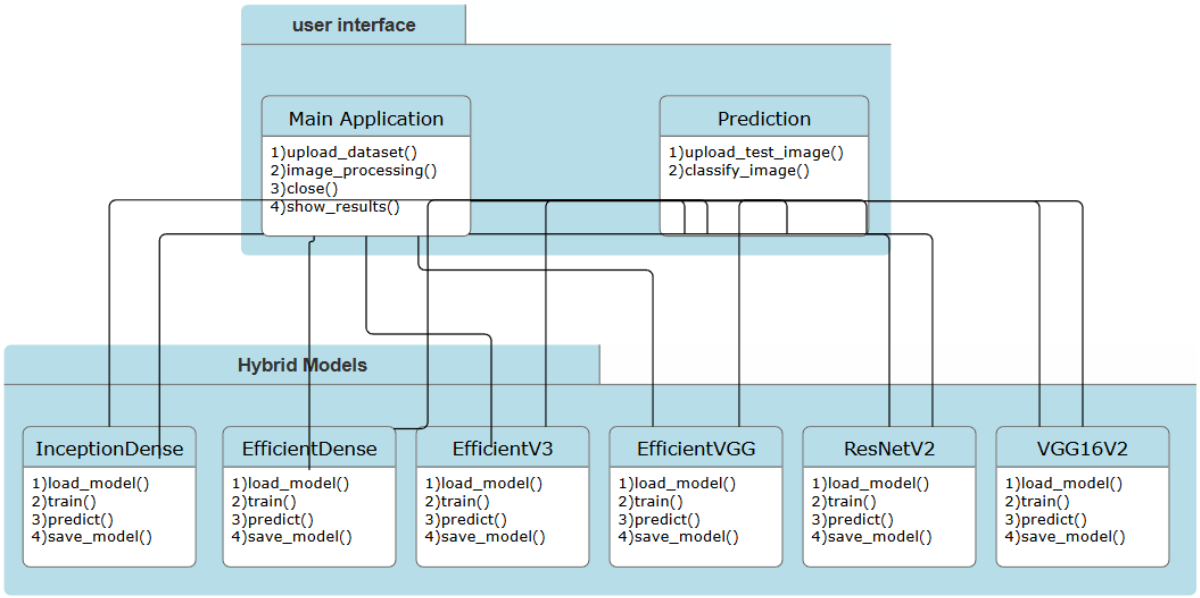


Fig 5.3.1: Class Diagram of Proposed System

2.Use case Diagram

A use case diagram in the Unified Modelling Language (UML) is a type of behaviouraldiagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed forwhich actor. Roles of the actors in the system can be depicted.

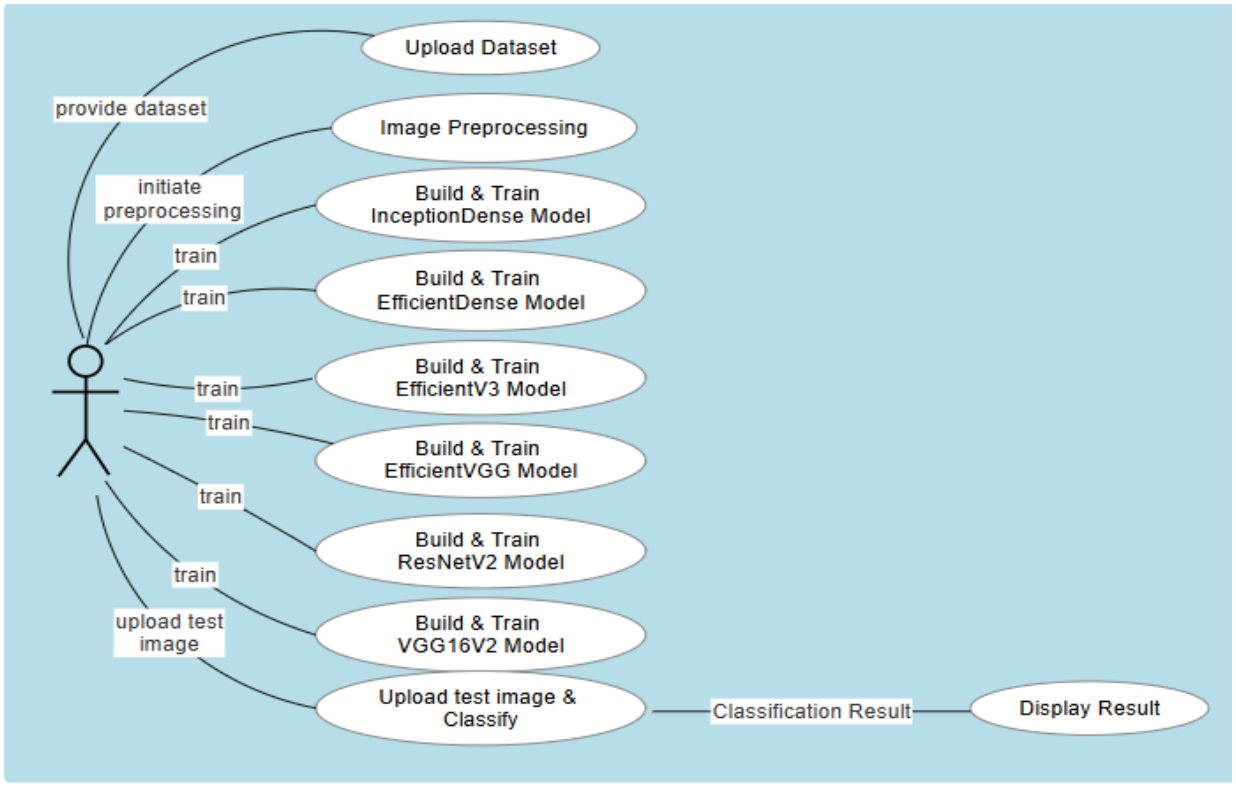


Fig 5.3.2: Use Case Diagram of Proposed System

3.Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of how data moves through a system. It shows the processes that transform data inputs into outputs, the data stores where information is kept, the data flows that connect different parts of the system, and the external entities that interact with the system. DFDs help in understanding the flow of information within a system, identifying data sources and destinations, and designing efficient data processing workflows. They are commonly used in system analysis and design to visualize data movement and improve system understanding.

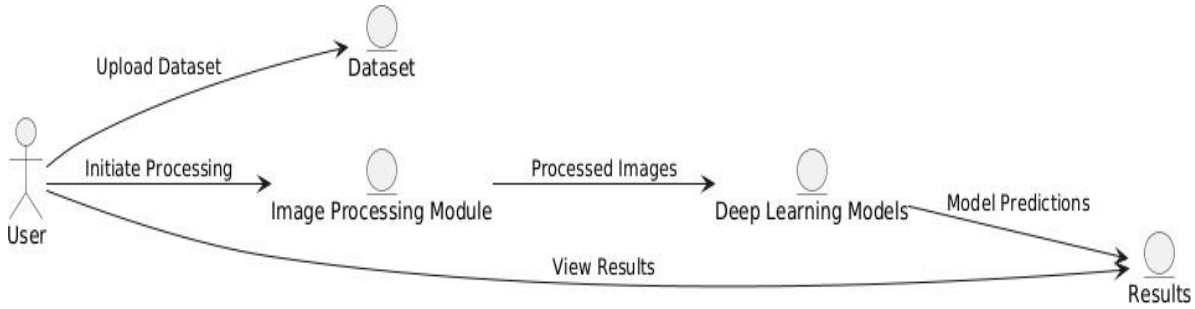


Fig 5.3.3: Data Flow Diagram of Proposed System

4.Activity diagram:

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the businessand operational step-by-step workflows of components in a system. An activity diagramshows the overall flow of control.

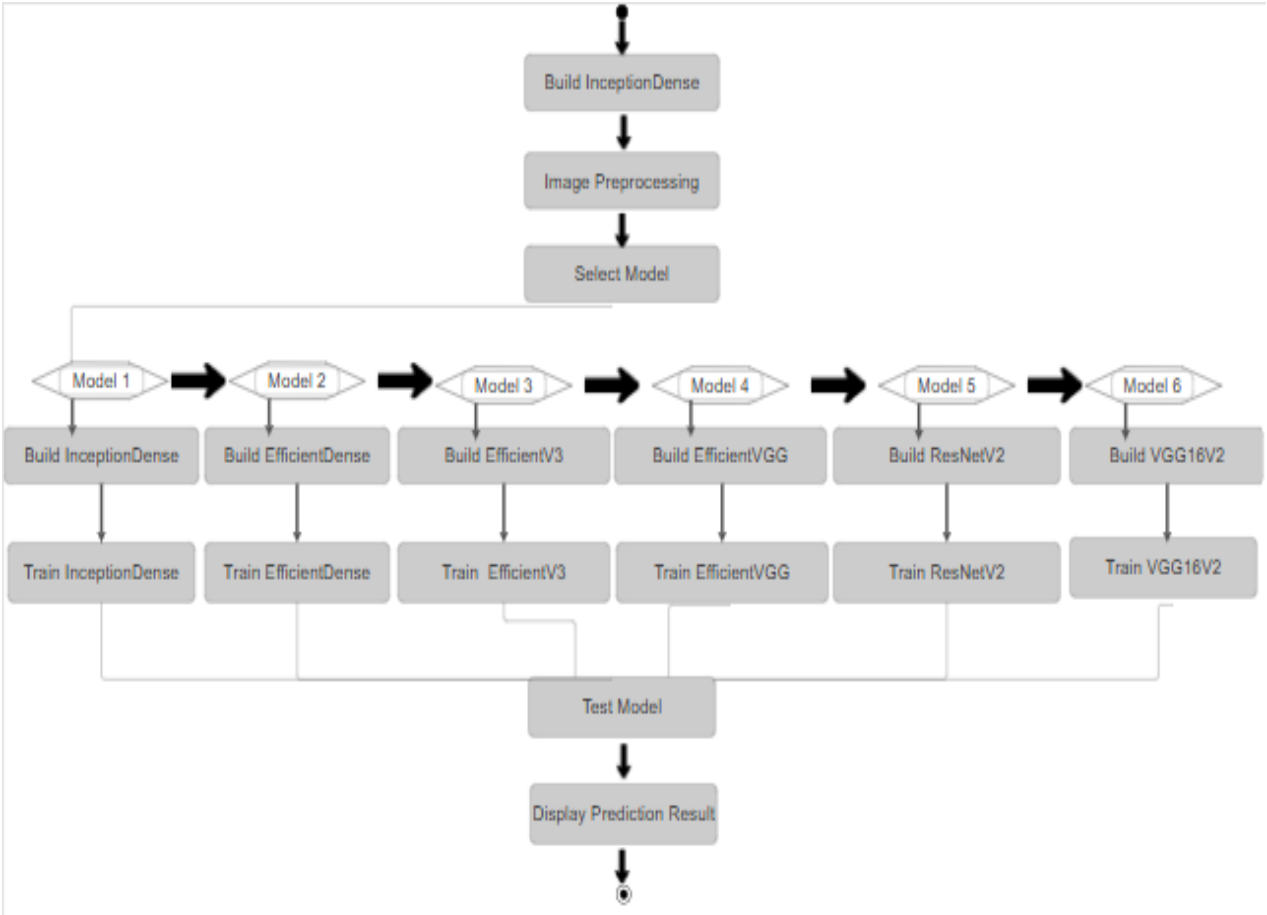


Fig 5.3.4: Activity Diagram of Proposed System

5.Sequence Diagram

A **sequence diagram** in Unified Modeling Language (UML) is a kind of interaction diagram that shows

how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

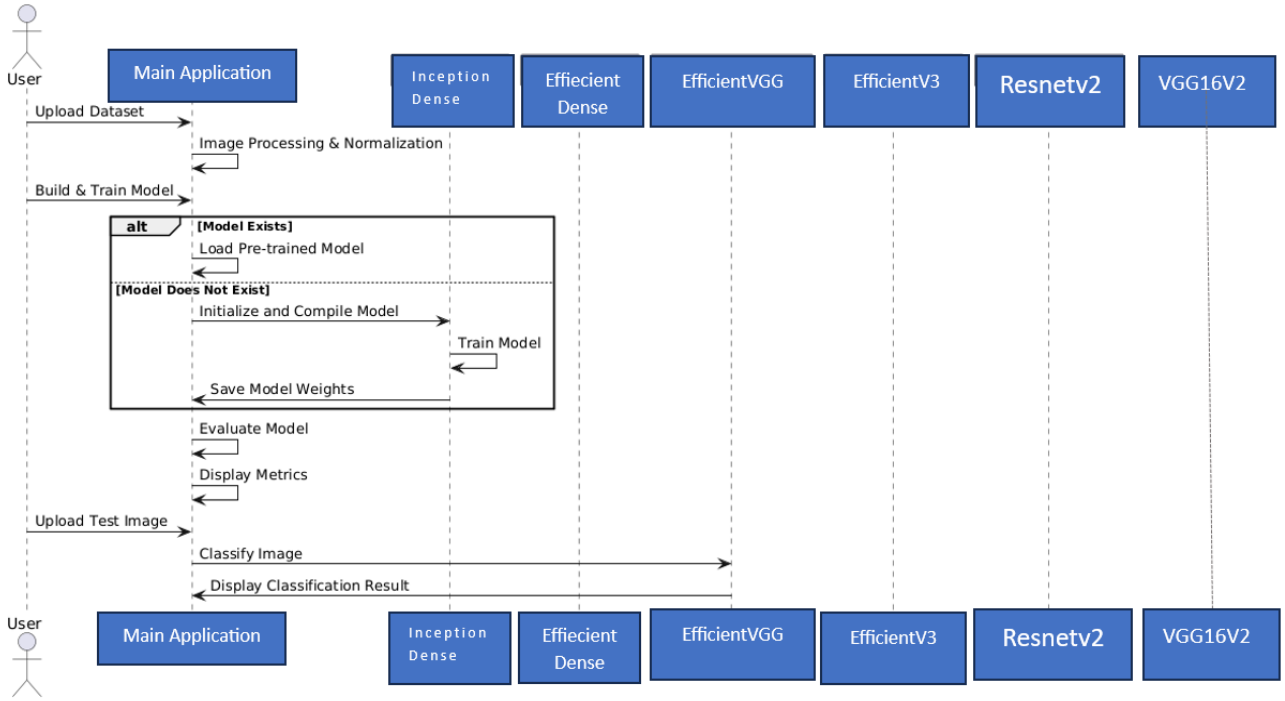


Fig 5.3.5: Sequence Diagram of Proposed System

6. Deployment Diagram: Describes the deployment architecture of the system, including hardware and software components.

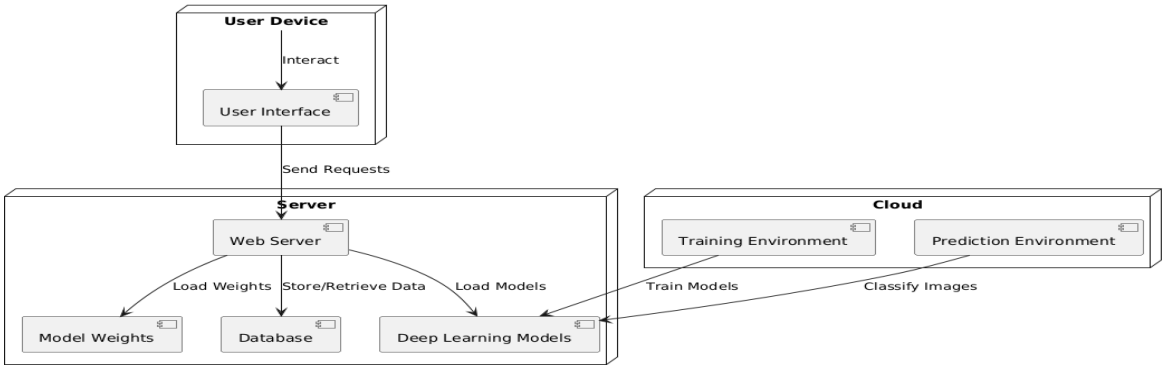


Fig 5.3.6: Deployment Diagram of Proposed System

7.Component Diagram

A component diagram in UML depicts the structural organization of software components and their interactions within a system. It illustrates the modular architecture of a software system by showing the components, their interfaces, dependencies, and relationships. Component diagrams help in understanding the system's structure, promoting modular design, and facilitating communication among development teams. They showcase the building blocks of the system and how they collaborate to achieve the system's functionality. Overall, component diagrams are essential in designing and documenting software systems with a focus on component-based development and system composition.

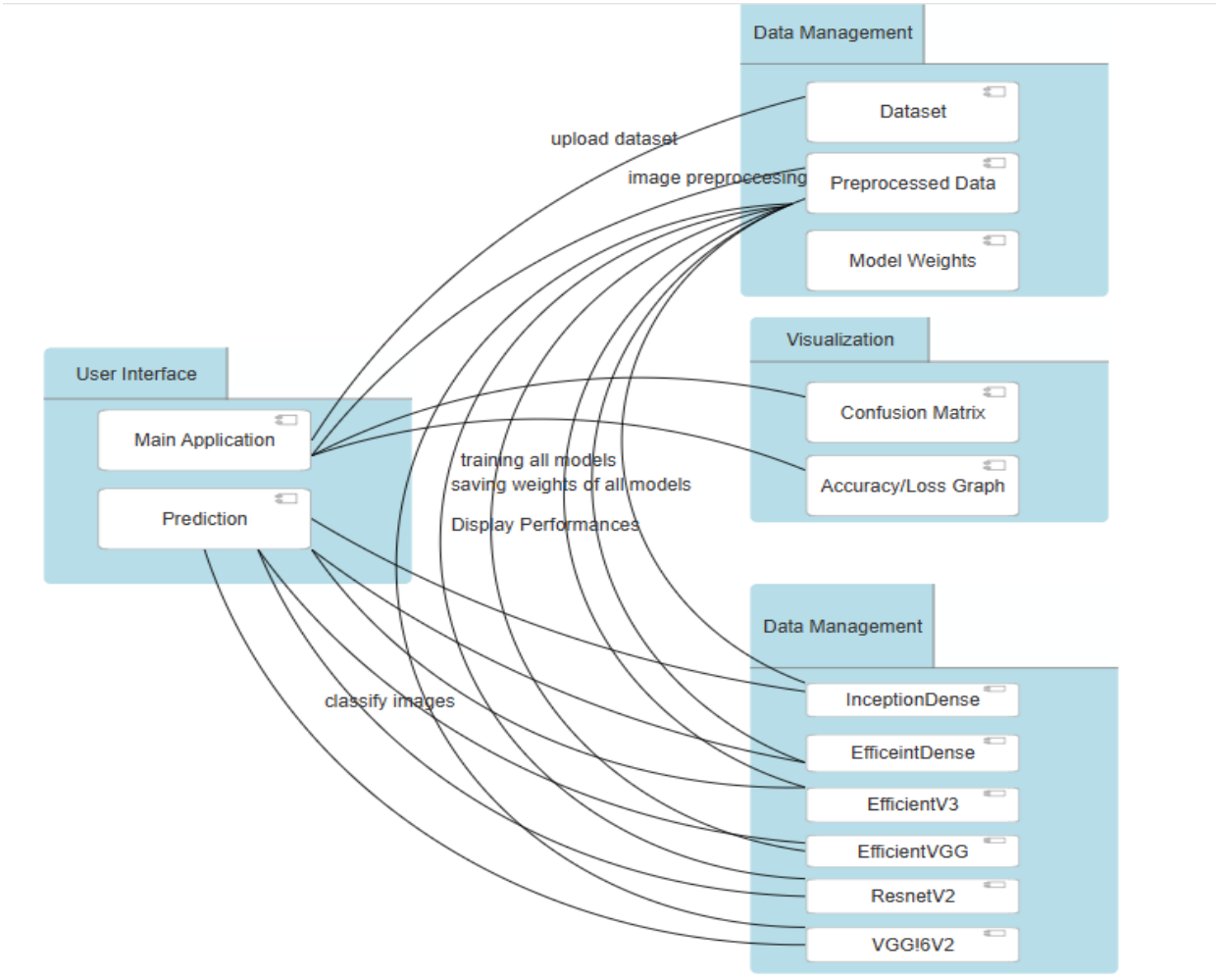


Fig 5.3.7: Component Diagram of Proposed System

6. IMPLEMENTATION

6.1 MODEL IMPLEMENTATION

The Python code is a graphical user interface (GUI) application using the Tkinter library for weapon recognition from images. The application allows users to perform the following tasks:

1. InceptionDense Implementation:

- **File Loading:** The code checks if the model JSON file and weights are available to load a previously saved model.
- **Model Loading:** It loads the InceptionDense model's architecture and weights if they exist, setting up the model for prediction.
- **Model Summary:** The code prints the model structure, including layers, parameters, etc.
- **Training History:** If history exists, the model retrieves and displays the training accuracy of the last epoch.
- **Training (If Not Preloaded):** If no saved model exists, it imports the pre-trained Inception model with imagenet weights and integrates Dense layers for classification.
- **Custom Layers:** It adds GlobalAveragePooling2D and dense layers to tailor the output for the classification task.
- **Layer Freezing:** The pre-trained Inception layers are frozen, preventing their weights from being updated during training.
- **Model Compilation:** The model is compiled using the Adam optimizer and categorical cross-entropy loss.
- **Model Training:** The model is trained on the dataset, and the process is displayed (epochs, validation data, batch size, etc.).
- **Model Saving:** After training, the architecture and weights are saved to disk for later use.
- **Metrics Calculation:** After training, accuracy, precision, and recall are computed for the test set.
- **Confusion Matrix:** A confusion matrix is generated and displayed using a heatmap for visualizing the classification performance.

2. EfficientDense Implementation:

- **File Loading:** The script checks if the EfficientDense model has been saved to load the model and weights.
- **Model Loading:** If the files exist, the model is restored from disk for further use, including loading weights.
- **Model Summary:** It outputs the EfficientDense model architecture.
- **Training History:** The code retrieves and displays the accuracy from the model's training history.
- **Training Process (If Not Saved):** If the model isn't pre-saved, it loads the EfficientNet base model, excluding the top fully connected layers, with imagenet weights.
- **Custom Layers:** Pooling and dense layers are added to the base EfficientDense model for classification.
- **Freezing Layers:** Pre-trained layers are frozen to maintain their weights, which speeds up training and prevents overfitting.
- **Compilation:** The model is compiled with categorical cross-entropy and the Adam optimizer.
- **Training:** Model training occurs with a specific number of epochs, batch size, and validation data.
- **Saving the Model:** After training, both the architecture and weights are saved to files for later use.
- **Evaluation:** Accuracy, precision, and recall are computed, and a confusion matrix is generated to visualize classification performance.

3. EfficientV3 Implementation:

- **File Loading:** The code checks whether the EfficientV3 model's architecture and weights have been saved.
- **Model Loading:** If available, it loads the EfficientV3 model from disk for further predictions or evaluations.
- **Model Summary:** It displays the model structure (number of layers, types, parameters, etc.).
- **Training History:** Retrieves the accuracy of the last epoch if the model was trained before.
- **Training Process (If Not Saved):** If no model exists, it loads the pre-trained EfficientNetV3 model with the imagenet weights, excluding the fully connected layers at the top.
- **Custom Layers:** GlobalAveragePooling2D and dense layers are added for classification.
- **Layer Freezing:** All layers in the EfficientV3 base model are frozen to retain pre-trained

weights.

- **Compilation:** The model is compiled using the Adam optimizer and categorical cross-entropy loss.
- **Training:** The model is trained with training data, batch size, and validation data.
- **Model Saving:** After training, the model architecture and weights are saved to disk for future use.
- **Evaluation:** Predictions are made on the test set, followed by calculating accuracy, precision, and recall.
- **Confusion Matrix:** A confusion matrix is generated to assess classification results, with visualization via a heatmap.

4. EfficientVGG Implementation:

- **File Loading:** The script first checks if the EfficientVGG model exists in files.
- **Model Loading:** If the model files exist, it loads the architecture and weights from disk.
- **Model Summary:** It prints the model summary to understand the architecture.
- **Training History:** It retrieves the history of the training, displaying the accuracy at the last epoch.
- **Training Process (If Not Saved):** If the model is not preloaded, it uses the pre-trained EfficientVGG model with the imagenet weights.
- **Custom Layers:** GlobalAveragePooling2D and dense layers are added for classification.
- **Freezing Layers:** EfficientVGG layers are frozen to retain pre-trained knowledge.
- **Compilation:** The model is compiled using categorical cross-entropy and Adam optimizer.
- **Training:** The model is trained with batch size, validation data, and other parameters.
- **Model Saving:** After training, it saves the architecture and weights for future use.
- **Evaluation:** Predictions are made, and performance metrics (accuracy, precision, recall) are calculated.
- **Confusion Matrix:** A confusion matrix is computed and visualized using a heatmap.

5. ResNetV2 Implementation:

- **File Loading:** The code checks for the existence of the ResNetV2 model JSON and weight files.
- **Model Loading:** If available, the model and weights are loaded for prediction or further evaluation.

- **Model Summary:** The model architecture is printed, showing layers and parameter counts.
- **Training History:** It retrieves and displays the accuracy from the last epoch of a previously trained model.
- **Training Process (If Not Saved):** If no saved model exists, a ResNetV2 model is built from the pre-trained ResNet50V2 base model.
- **Layer Architecture:** The architecture includes convolutional layers followed by pooling, flattening, dense layers, and an output softmax layer for classification.
- **Compilation:** The model is compiled using the Adam optimizer and categorical cross-entropy loss.
- **Training:** The model is trained with a specified number of epochs, batch size, and validation data.
- **Model Saving:** After training, the architecture and weights are saved for later use.
- **Evaluation:** Model predictions on the test set are performed, and accuracy, precision, and recall are calculated.
- **Confusion Matrix:** A confusion matrix is computed to assess model performance, visualized through a heatmap.
- **Classification Report:** It generates a classification report summarizing precision, recall, and F1 score across all classes.

6. VGG16V2 Implementation:

- **Model Construction:** The hybrid model combines VGG16 and MobileNetV2 as base models.
- **Feature Extraction:** Global Average Pooling (GAP) is applied to the outputs of both base models.
- **Feature Fusion:** The extracted features from both models are concatenated.
- **Fully Connected Layers:** A dense layer with 256 neurons and ReLU activation is added, followed by a sigmoid-activated output layer for binary classification.
- **Model Compilation:** The model is compiled using the Adam optimizer with a tuned learning rate and binary cross-entropy loss.
- **Training Process:** The model is trained using the dataset with a 25% validation split and 10 epochs.
- **Evaluation:** The trained model is evaluated on the test set, calculating accuracy, precision, recall, F1 score, and specificity.

- **Performance Metrics:** The results, including accuracy, precision, recall, F1 score, and specificity, are printed for analysis.

Metrics:

Performance metrics for weapon recognition from images using deep neural networks typically include the following:

Accuracy: Accuracy measures the overall correctness of the predictions made by the neural network model. It is calculated as the ratio of correctly classified instances to the total number of instances in the dataset. In weapon recognition, accuracy indicates how often the model correctly identifies whether an image contains a weapon or not.

Precision: Precision measures the proportion of true positive predictions (correctly identified weapons) out of all positive predictions made by the model. It helps in understanding the reliability of the model when it predicts that an image contains a weapon.

Precision = True Positives / (True Positives + False Positives)

Recall (Sensitivity): Recall measures the proportion of true positive predictions out of all actual positive instances in the dataset. It indicates the ability of the model to correctly detect weapons when they are present in the images.

Recall = True Positives / (True Positives + False Negatives)

F1 Score: The F1 score is the harmonic mean of precision and recall. It provides a single score that balances both precision and recall. F1 score is useful when there is an imbalance between the number of positive and negative instances in the dataset.

F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

Specificity: Specificity measures the proportion of true negative predictions (correctly identified non-weapons) out of all actual negative instances in the dataset. It is particularly relevant in scenarios where the consequences of false alarms (misclassifying non-weapons as weapons) are critical.

Specificity = True Negatives / (True Negatives + False Positives)

Confusion Matrix: A confusion matrix provides a detailed breakdown of the model's predictions, showing the number of true positives, true negatives, false positives, and false negatives. It helps in understanding the types of errors made by the model and can inform adjustments to improve performance.These performance metrics collectively offer insights into the effectiveness of deep neural network models for pancreatic cancer detection from images.

6.2 CODING

```
!pip install keras-tuner

from google.colab import drive
drive.mount('/content/drive')

import os
import numpy as np
import cv2
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Concatenate
from tensorflow.keras.applications import EfficientNetB0, InceptionV3, ResNet50,
DenseNet121, VGG16, MobileNetV2
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import classification_report, confusion_matrix
from kerastuner.tuners import RandomSearch
```

```
import matplotlib.pyplot as plt

# Data Preprocessing
def preprocess_input(img):
    img = cv2.resize(img, (224, 224))
    img = img / 255.0
    return img

def load_images_from_folder(folder):
    images = []
    for filename in os.listdir(folder):
        if filename.endswith(".jpg"):
            img = cv2.imread(os.path.join(folder, filename), cv2.IMREAD_GRAYSCALE)
            img = preprocess_input(img)
            img = np.stack((img,)*3, axis=-1)  # Convert grayscale to RGB by stacking
            if img is not None:
                images.append(img)
    return np.array(images)


# Load datasets
train_normal =
load_images_from_folder('/content/drive/MyDrive/ct_images_02/DATASET/train/train/normal'
)
train_tumor =
load_images_from_folder('/content/drive/MyDrive/ct_images_02/DATASET/train/train/pancre
atic_tumor')
test_normal =
load_images_from_folder('/content/drive/MyDrive/ct_images_02/DATASET/test/test/normal')
test_tumor =
load_images_from_folder('/content/drive/MyDrive/ct_images_02/DATASET/test/test/pancreati
c_tumor')
```

```
# Create labels
train_labels_normal = np.zeros(train_normal.shape[0])
train_labels_tumor = np.ones(train_tumor.shape[0])
test_labels_normal = np.zeros(test_normal.shape[0])
test_labels_tumor = np.ones(test_tumor.shape[0])

# Combine datasets
train_images = np.concatenate((train_normal, train_tumor), axis=0)
train_labels = np.concatenate((train_labels_normal, train_labels_tumor), axis=0)
test_images = np.concatenate((test_normal, test_tumor), axis=0)
test_labels = np.concatenate((test_labels_normal, test_labels_tumor), axis=0)

# Data Augmentation
data_augmentation = tf.keras.Sequential([
    tf.keras.layers.RandomFlip("horizontal_and_vertical"),
    tf.keras.layers.RandomRotation(0.2),
    tf.keras.layers.RandomZoom(0.2),
    tf.keras.layers.RandomContrast(0.2),
])

# Apply augmentation to training images
train_images = data_augmentation(train_images, training=True)

# Set the best learning rate obtained from previous tuning
best_learning_rate = 0.0001
# Evaluate the Models
def evaluate_model(model, test_images, test_labels):
    # Make predictions
    y_pred = (model.predict([test_images, test_images]) > 0.5).astype("int32")
    # Generate confusion matrix
    confusion_mtx = confusion_matrix(test_labels, y_pred)
```

```
    TN, FP, FN, TP = confusion_mtx.ravel()
    # Calculate metrics
    accuracy = (TP + TN) / (TP + TN + FP + FN)
    precision = TP / (TP + FP) if (TP + FP) > 0 else 0
    recall = TP / (TP + FN) if (TP + FN) > 0 else 0
    f1_score = 2 * (precision * recall) / (precision + recall) if (precision + recall) > 0 else 0
    specificity = TN / (TN + FP) if (TN + FP) > 0 else 0
    return accuracy, precision, recall, f1_score, specificity, confusion_mtx
```

VGG16V2:

```
def VGG16V2():
    base_model1 = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
    base_model2 = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224,
224, 3))
    x1 = base_model1.output
    x1 = GlobalAveragePooling2D()(x1)
    x2 = base_model2.output
    x2 = GlobalAveragePooling2D()(x2)
    combined = Concatenate()([x1, x2])
    x = Dense(256, activation='relu')(combined)
    predictions = Dense(1, activation='sigmoid')(x)
    model = Model(inputs=[base_model1.input, base_model2.input], outputs=predictions)
    return model

# Train Hybrid Model VGG16V2
model1 = VGG16V2()
model1.compile(optimizer=Adam(learning_rate=best_learning_rate),
loss='binary_crossentropy', metrics=['accuracy'])
model1.fit([train_images, train_images], train_labels, validation_split=0.25, epochs=10)

# Evaluate Hybrid Model 1
```

```
metrics1 = evaluate_model(model1, test_images, test_labels)
print(f"Hybrid Model 1 (VGG16+ MobileNetV2): Accuracy: {metrics1[0]:.4f}, Precision: {metrics1[1]:.4f}, Recall: {metrics1[2]:.4f}, F1 Score: {metrics1[3]:.4f}, Specificity: {metrics1[4]:.4f}")
```

EfficientDense:

```
def EfficientDense():
    base_model1 = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
    base_model2 = DenseNet121(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
    x1 = base_model1.output
    x1 = GlobalAveragePooling2D(name='efficientnet_pooling')(x1)
    x2 = base_model2.output
    x2 = GlobalAveragePooling2D(name='densenet_pooling')(x2)
    combined = Concatenate(name='concatenate')([x1, x2])
    x = Dense(256, activation='relu', name='dense1')(combined)
    predictions = Dense(1, activation='sigmoid', name='output')(x)
    model = Model(inputs=[base_model1.input, base_model2.input], outputs=predictions)
    return model

# Train Hybrid Model EfficientDense
model2 = EfficientDense()
model2.compile(optimizer=Adam(learning_rate=best_learning_rate), loss='binary_crossentropy', metrics=['accuracy'])
model2.fit([train_images, train_images], train_labels, validation_split=0.2, epochs=10)

# Evaluate Hybrid Model 2
metrics2 = evaluate_model(model2, test_images, test_labels)
print(f"Hybrid Model 2 (EfficientNetB0+ DenseNet121): Accuracy: {metrics2[0]:.4f}, Precision: {metrics2[1]:.4f}, Recall: {metrics2[2]:.4f}, F1 Score: {metrics2[3]:.4f}, Specificity: {metrics2[4]:.4f}")
```

EfficientV3:

```
def EfficientV3():
    base_model1 = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
    base_model2 = InceptionV3(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
    x1 = base_model1.output
    x1 = GlobalAveragePooling2D()(x1)
    x2 = base_model2.output
    x2 = GlobalAveragePooling2D()(x2)
    combined = Concatenate()([x1, x2])
    x = Dense(256, activation='relu')(combined)
    predictions = Dense(1, activation='sigmoid')(x)
    model = Model(inputs=[base_model1.input, base_model2.input], outputs=predictions)
    return model

# Train Hybrid Model EfficientV3
model3 = EfficientV3()
model3.compile(optimizer=Adam(learning_rate=best_learning_rate),
loss='binary_crossentropy', metrics=['accuracy'])
model3.fit([train_images, train_images], train_labels, validation_split=0.2, epochs=10)

# Evaluate Hybrid Model 3
metrics3 = evaluate_model(model3, test_images, test_labels)
print(f"Hybrid Model 3 (EfficientNetB0+ InceptionV3): Accuracy: {metrics3[0]:.4f}, Precision: {metrics3[1]:.4f}, Recall: {metrics3[2]:.4f}, F1 Score: {metrics3[3]:.4f}, Specificity: {metrics3[4]:.4f}")
```

InceptionDense:

```
def InceptionDense():
    base_model1 = InceptionV3(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
    base_model2 = DenseNet121(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
    x1 = base_model1.output
    x1 = GlobalAveragePooling2D(name='inception_pooling')(x1)
    x2 = base_model2.output
    x2 = GlobalAveragePooling2D(name='densenet_pooling')(x2)
    combined = Concatenate(name='concatenate')( [x1, x2])
    x = Dense(256, activation='relu', name='dense1')(combined)
    predictions = Dense(1, activation='sigmoid', name='output')(x)
    model = Model(inputs=[base_model1.input, base_model2.input], outputs=predictions)
    return model


# Train Hybrid Model InceptionDense
model4 = InceptionDense()
model4.compile(optimizer=Adam(learning_rate=best_learning_rate),
loss='binary_crossentropy', metrics=['accuracy'])
model4.fit([train_images, train_images], train_labels, validation_split=0.2, epochs=10)

# Evaluate Hybrid Model 4
metrics4 = evaluate_model(model4, test_images, test_labels)
print(f"Hybrid Model 4 (InceptionV3 + DenseNet121): Accuracy: {metrics4[0]:.4f}, Precision:
{metrics4[1]:.4f}, Recall: {metrics4[2]:.4f}, F1 Score: {metrics4[3]:.4f}, Specificity:
{metrics4[4]:.4f}")
```

EfficientVGG:

```
def EfficientVGG():
    base_model1 = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(224,
224, 3))
    base_model2 = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
    x1 = base_model1.output
    x1 = GlobalAveragePooling2D(name='efficientnet_pooling')(x1)
    x2 = base_model2.output
    x2 = GlobalAveragePooling2D(name='vgg_pooling')(x2)
    combined = Concatenate(name='concatenate')( [x1, x2])
    x = Dense(256, activation='relu', name='dense1')(combined)
    predictions = Dense(1, activation='sigmoid', name='output')(x)
    model = Model(inputs=[base_model1.input, base_model2.input], outputs=predictions)
    return model


# Train Hybrid Model EfficientVGG
model5 = EfficientVGG()
model5.compile(optimizer=Adam(learning_rate=best_learning_rate),
loss='binary_crossentropy', metrics=['accuracy'])
model5.fit([train_images, train_images], train_labels, validation_split=0.2, epochs=10)

# Evaluate Hybrid Model 5
metrics5 = evaluate_model(model5, test_images, test_labels)
print(f"Hybrid Model 5 (EfficientNetB0 + VGG16): Accuracy: {metrics5[0]:.4f}, Precision:
{metrics5[1]:.4f}, Recall: {metrics5[2]:.4f}, F1 Score: {metrics5[3]:.4f}, Specificity:
{metrics5[4]:.4f}")
```

ResNetV2:

```
def ResNetV2():
    base_model1 = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
    base_model2 = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224,
```

```
224, 3))
    x1 = base_model1.output
    x1 = GlobalAveragePooling2D(name='resnet_pooling')(x1)
    x2 = base_model2.output
    x2 = GlobalAveragePooling2D(name='mobilenet_pooling')(x2)
    combined = Concatenate(name='concatenate')( [x1, x2])
    x = Dense(256, activation='relu', name='dense1')(combined)
    predictions = Dense(1, activation='sigmoid', name='output')(x)
    model = Model(inputs=[base_model1.input, base_model2.input], outputs=predictions)
    return model

# Train Hybrid Model ResNetV2
model6 = ResNetV2()
model6.compile(optimizer=Adam(learning_rate=best_learning_rate),
loss='binary_crossentropy', metrics=['accuracy'])
model6.fit([train_images, train_images], train_labels, validation_split=0.2, epochs=10)

#  Evaluate Hybrid Model 6
metrics6 = evaluate_model(model6, test_images, test_labels)
print(f"Hybrid Model 6 (ResNet50 + MobileNetV2): Accuracy: {metrics6[0]:.4f}, Precision:
{metrics6[1]:.4f}, Recall: {metrics6[2]:.4f}, F1 Score: {metrics6[3]:.4f}, Specificity:
{metrics6[4]:.4f}")
```

Frontend

App.py

```
from flask import Flask, render_template, request, redirect, url_for, Blueprint
import tensorflow as tf
import numpy as np
```

```
from werkzeug.utils import secure_filename
import os
import gdown
import h5py
import json


app = Flask(__name__)


predictions = Blueprint('predictions', __name__)
@predictions.route('/')
def index():
    return render_template('index.html')


def fix_layer_names(filepath):
    with h5py.File(filepath, 'r+') as f:
        if 'model_weights' in f:
            for layer in list(f['model_weights'].keys()):
                if '/' in layer:
                    new_layer_name = layer.replace('/', '_')
                    if new_layer_name not in f['model_weights']:
                        f['model_weights'][new_layer_name] = f['model_weights'][layer]
                    del f['model_weights'][layer]
```

```
        if 'layer_names' in f.attrs:
            layer_names = list(f.attrs['layer_names'])
            for i in range(len(layer_names)):
                if '/' in layer_names[i].decode('utf-8'):
                    layer_names[i] = layer_names[i].replace('/', '_').encode('utf-8')
            f.attrs['layer_names'] = layer_names

        if 'model_config' in f.attrs:
            model_config = json.loads(f.attrs['model_config'])
            for layer in model_config['config']['layers']:
                if '/' in layer['config']['name']:
                    layer['config']['name'] = layer['config']['name'].replace('/', '_')
            f.attrs['model_config'] = json.dumps(model_config)

if not os.path.exists('models'):
    os.makedirs('models')

if not os.path.exists('uploads'):
    os.makedirs('uploads')
url = 'https://drive.google.com/uc?id=16CaNmuiztJQFnuDAoQhSd-zRcAd7MLkg'
output = './models/inceptiondense_model.h5'
if not os.path.exists(output):
    gdown.download(url, output, quiet=False, fuzzy=True)
```

```
fix_layer_names(output)
class CustomDepthwiseConv2D(tf.keras.layers.DepthwiseConv2D):
    def __init__(self, groups=1, **kwargs):
        if 'groups' in kwargs:
            del kwargs['groups']
        super().__init__(**kwargs)
with tf.keras.utils.custom_object_scope({'DepthwiseConv2D': CustomDepthwiseConv2D}):
    model = tf.keras.models.load_model(output)


@app.route('/')
def index():
    return render_template('index.html')


@app.route('/about')
def about():
    return render_template('about/about.html')


@app.route('/predict', methods=['POST', 'GET'])
def predict():
    if request.method == 'POST':
        file = request.files['image']
        if file:
            filename = secure_filename(file.filename)
```

```
            filepath = os.path.join('uploads', filename)
            file.save(filepath)

            # Preprocess the image
            img = tf.keras.preprocessing.image.load_img(filepath, target_size=(224, 224))
            img_array = tf.keras.preprocessing.image.img_to_array(img)
            img_array = np.expand_dims(img_array, axis=0)
            img_array /= 255.0

            img_inputs = [img_array, img_array]

            prediction = model.predict(img_inputs)
            if prediction[0] > 0.5:
                prediction_result = "😢 The patient has pancreatic tumor."
            else:
                prediction_result = "😊 The patient doesn\'t have pancreatic cancer."
            return render_template('predictions/result.html', prediction_text=prediction_result)
    return render_template('predictions/index.html')

@app.route("/metrics")
def metrics():
    return render_template("metrics/metrics.html")
```

```
@app.route("/flowchart")
def flowchart():
    return render_template("flowchart/flowchart.html")


if __name__ == "__main__":
    app.run(debug=True)
```

index.html

```
{% extends "base.html" %}


{% block content %}
<style>
    /* Background with Image and Opacity */
    body {
        background: linear-gradient(rgba(10, 10, 10, 0.8), rgba(10, 10, 10, 0.8)),
                    url('/static/images/bg.jpg') no-repeat center center fixed;
        background-size: cover;
        color: white;
        font-family: 'Century', sans-serif; /* Century font applied here */
        margin: 0;
        padding: 0;
        height: 100vh;
```

```
}

/* Title Styling - Aligning it to the left */
h1 {
    opacity: 0;
    animation: fadeIn 2s forwards;
    font-family: 'Century', serif; /* Century font applied */
    font-size: 2rem;
    font-weight: 700;
    text-transform: uppercase;
    color: #f8f9fa;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.7);
    margin: 50px 0 20px 30px; /* Added margin and aligned left */
}

/* Paragraph Styling - Padding on left and right */
p {
    opacity: 0;
    animation: fadeIn 3s forwards 1s;
    text-align: justify;
    font-family: 'Century', serif; /* Century font applied */
    font-size: 1.3rem;
    line-height: 1.8;
```

```
        color: #eaeaea;
        margin: 30px 130px 0 30px; /* Added padding to left and right */
    }

    /* Fade-in animation for content */
    @keyframes fadeIn {
        0% {
            opacity: 0;
        }
        100% {
            opacity: 1;
        }
    }
</style>

<h1>Pancreatic Cancer Detection</h1>

<p>This project employs advanced deep learning techniques for pancreatic cancer detection <br>
    and classification from medical images, leveraging a hybrid architecture of InceptionV3 <br>
    and DenseNet121, known as InceptionDense.The combined paradigm enhances the <br>
    model's ability to identify tumors of varying sizes and shapes, significantly improving <br>
    detection capabilities.</p>
```

Our study utilized a dataset of annotated CT images and implemented robust preprocessing methods, including multi-scale feature extraction and dense connectivity. This approach not only improves model accuracy but also ensures reliable diagnostics for early pancreatic cancer detection, aiding healthcare professionals in providing timely and effective treatment.

{% endblock %}

base.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{{ title | default("Flask Project") }}</title>
    <style>
        body {
            font-family: 'Century', sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f8f9fa;
            color: #212529;
        }

        header {
```

```
    background: linear-gradient(90deg, #343A40, #343A40);
    color: #fff;
    padding: 10px 20px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    position: static; /* Changed to static */
    z-index: 1000;
    display: flex;
    align-items: center;
    flex-direction: row; /* Align logo and content horizontally */
}

.logo {
    height: 90px;
    margin-right: 10px; /* Space between logo and content */
}

.header-content {
    display: flex;
    flex-direction: column;
    justify-content: center;
    flex-grow: 1;
    text-align: center;
    width: 100%;
```

```
}

.project-title {
    font-size: 1.6rem;
    font-weight: bold;
    font-family: 'Poppins', Arial, sans-serif;
    text-shadow: 0px 1px 3px rgba(0, 0, 0, 0.2);
    margin: 0;
}

.team-info {
    font-size: 1rem;
    font-weight: 400;
    margin-top: 10px;
    color: #dfe6e9;
    white-space: nowrap; /* Ensure the text stays on a single line */
    font-family: 'Century', sans-serif;
}

.team-info span {
    margin-right: 15px; /* Space between the details */
}
```

```
nav {
    margin-top: 0px;
}

.navbar {
    list-style: none;
    display: flex;
    gap: 50px;
    margin: 0;
    padding: 0;
    justify-content: center; /* Align the nav items in the center */
}

.navbar li {
    position: relative;
    display: flex;
    align-items: center;
}

.navbar li a {
    text-decoration: none;
    font-size: 0.95rem;
    font-weight: 400;
```

```
    color: #fff;
    transition: color 0.3s ease;
    display: flex;
    align-items: center;
    gap: 8px; /* Add space between icon and text */
}

.navbar li a::after {
    content: '';
    position: absolute;
    width: 0;
    height: 2px;
    background-color: #dfe6e9;
    bottom: 0;
    left: 0;
    transition: width 0.3s ease;
}

.navbar li a:hover {
    color: #dfe6e9; /* Changed hover effect to color */
}

/* Styling for the separator */
```

```
        hr {
            border: 0;
            border-top: 2px solid #dfe6e9;
            width: 80%;
            margin: 20px auto;
        }
    </style>
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;700&family=Roboto:wght@400;500&
display=swap" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta3/css/all.min.css"> <!-- Font Awesome CDN -->
</head>
<body>
    <header>
        <img src="{{ url_for('static', filename='images/logo.jpg') }}" alt="Logo" class="logo">
        <div class="header-content">
            <div class="project-title">{{ title | default("Advanced Techniques in Deep Learning for Pancreatic
Cancer Detection and Classification") }}</div>
            <!-- Team info section added in a single line -->
            <div class="team-info">
                <span>Team Members: Gandikota Narendra, Katta Subbarao, Nallabothu Narendra  |</span>
                <span>Guide: K.V.Narasimha Reddy  |</span>
                <span>Mentor: D.Venkata Reddy</span>
```

```
        </div>
        <!-- Separator added here -->
        <hr>
        <nav>
          <ul class="navbar">
            <li><a href="{{ url_for('index') }}"><i class="fas fa-home"></i> Home</a></li>
            <li><a href="{{ url_for('about') }}"><i class="fas fa-info-circle"></i> About Project</a></li>
            <li><a href="{{ url_for('predict') }}"><i class="fas fa-calculator"></i> Prediction</a></li>
            <li><a href="{{ url_for('metrics') }}"><i class="fas fa-chart-line"></i> Metrics</a></li>
            <li><a href="{{ url_for('flowchart') }}"><i class="fas fa-sitemap"></i> Flowchart</a></li>
          </ul>
        </nav>
      </div>
    </header>
    <main>
      {% block content %}
      {% endblock %}
    </main>
</body>
</html>
```

about.html

```
{% extends "base.html" %}
```

```
{% block content %}

<div class="about-container">

    <h1 class="animated-title">About the Project</h1>

    <p class="animated-text">Advanced deep learning techniques for pancreatic cancer detection and classification, utilizing InceptionV3 and DenseNet121 in a hybrid<br> architecture InceptionDense for superior accuracy. The project focuses on robust <br>preprocessing, feature selection, and fusion to enhance model performance and <br>generalization.</p>


    <!-- Flex container for Goals and Technologies -->

    <div class="content-row">

        <div class="goals-section">

            <h2 class="animated-title">Project Goals</h2>

            <ul class="animated-text">

                <li>Enhance detection accuracy using hybrid architectures like InceptionDense.</li>

                <li>Utilize multi-scale feature extraction and feature reuse.</li>

                <li>Improve diagnostic reliability for early pancreatic cancer detection.</li>

                <li>Validate methods with larger datasets and extend to other imaging modalities.</li>

            </ul>

        </div>


        <div class="technologies-section">

            <h2 class="animated-title">Technologies Used</h2>

            <ul class="animated-text">
```

```
            <li>Deep Learning Models: InceptionV3, DenseNet121, InceptionDense</li>
            <li>Framework: Flask</li>
            <li>Programming Languages: Python</li>
            <li>Datasets: Annotated CT images from Kaggle</li>
          </ul>
        </div>
    </div>
</div>

<style>
  /* Background with Image and Opacity */
  .about-container {
    background: linear-gradient(rgba(10, 10, 10, 0.8), rgba(10, 10, 10, 0.8)),
                url('/static/images/bg.jpg') no-repeat center center fixed;
    background-size: cover;
    padding: 30px; /* Reduced padding */
    color: #fff;
    position: absolute;
    top: 100px; /* Adjusted to reduce space above content */
    left: 0;
    width: 100%;
    height: 100vh; /* Ensure container height fits the viewport */
    box-sizing: border-box;
```

```
  overflow: hidden; /* Hide overflow */
  font-family: 'Century', sans-serif;
  margin-top: 50px;
}

/* Typography for Titles */
.animated-title {
  opacity: 0;
  animation: fadeIn 2s forwards;
  font-family: 'Century', serif;
  font-size: 2rem;
  font-weight: 700;
  text-transform: uppercase;
  margin-bottom: 10px; /* Reduced margin */
  color: #f8f9fa;
  text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.7);
}

/* Paragraph and List Styling */
.animated-text {
  opacity: 0;
  animation: fadeIn 3s forwards 1s;
  font-family: 'Century', serif;
```

```
  font-size: 1.3rem;
  line-height: 1.8;
  color: #eaeaea;
  margin-bottom: 15px; /* Reduced margin */
  text-align: justify;
}

ul {
  padding-left: 20px;
  margin-top: 5px; /* Reduced margin */
  font-family: 'Century', serif;
  font-size: 1.1rem;
  line-height: 1.8;
  color: #eaeaea;
}

/* Content Row Layout */
.content-row {
  display: flex;
  gap: 0px; /* Reduced gap between sections */
  justify-content: space-between;
  flex-wrap: wrap;
  margin-top: 15px; /* Reduced margin to bring sections closer */
```

```
  }

  .goals-section, .technologies-section {
    width: 45%; /* Adjusted for better layout */
  }

  /* Fade-in animation for content */
  @keyframes fadeIn {
    0% {
      opacity: 0;
    }
    100% {
      opacity: 1;
    }
  }

  /* Prevent body overflow */
  body {
    overflow: hidden; /* Disable scrolling */
    height: 100vh; /* Ensure body height fits the viewport */
  }
</style>
{% endblock %}
```

Predictions/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Pancreatic Cancer Detection</title>
    <style>
        /* Using Century Gothic font, which is similar to Century style */
        body {
            font-family: 'Century Gothic', sans-serif;
            background: url('/static/images/bg.jpg') no-repeat center center fixed;
            background-size: cover;
            color: #f5f5f5;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: flex-start;
            align-items: center;
            height: 100vh;
            position: relative; /* Necessary for overlay positioning */
        }

        /* Semi-transparent overlay */
```

```
.overlay {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.6); /* Adjust opacity here */
    z-index: 1;
}

/* Main content styling */
.content {
    position: relative;
    z-index: 2; /* Ensure content is above the overlay */
    max-width: 700px;
    width: 100%;
    padding-left: 80px; /* Added left padding to move entire content further right */
}

h1 {
    text-align: left;
    font-size: 3em;
    font-weight: bold;
```

```
    color: #ffffff;
    text-shadow: 2px 2px 5px rgba(0, 0, 0, 0.6);
    background: rgba(0, 0, 0, 0.6);
    border-radius: 10px;
    padding: 10px 20px;
    display: inline-block;
    margin-left: 20px; /* Use padding from the content div */
}

form {
    display: flex;
    flex-wrap: wrap;
    padding: 30px 20px;
    justify-content: flex-start;
    gap: 30px; /* Reduced gap between columns */
}

.form-column {
    flex: 1 1 calc(50% - 20px); /* Increased space between columns */
    display: flex;
    flex-direction: column;
}
```

```
label {
    display: block;
    margin-bottom: 8px;
    font-weight: bold;
    font-size: 1em; /* Reduced label font size */
    color: #ffffff;
}

input {
    width: 100%;
    padding: 10px; /* Reduced padding for smaller fields */
    margin-bottom: 12px; /* Reduced margin for less spacing */
    border: 1px solid #ccc;
    border-radius: 8px;
    font-size: 0.9em; /* Reduced font size */
    background-color: rgba(0, 0, 0, 0.5);
    color: white;
    transition: all 0.3s ease;
}

input:focus {
    outline: none;
    border-color: #3498db;
```

```
    box-shadow: 0 0 8px #3498db;
}

button {
    background: linear-gradient(135deg, #a0e4d3, #77c1a1); /* Soft pastel mint green gradient */
    color: black;
    padding: 14px 28px; /* Increased padding for larger buttons */
    border: none;
    border-radius: 25px; /* Rounded corners */
    font-size: 1.2em; /* Slightly larger font size */
    cursor: pointer;
    transition: all 0.3s ease;
    margin: 10px 15px 0; /* Reduced top margin for less space */
}

/* Back button style */
.back-button {
    background: linear-gradient(135deg, #f5a623, #ffcc80); /* Warm orange gradient */
}

/* Removed brightness on hover */
button:hover {
    background: linear-gradient(135deg, #a0e4d3, #77c1a1); /* Same color on hover */
```

```
    box-shadow: none; /* Removed the box-shadow change */
}

.back-button:hover {
    background: linear-gradient(135deg, #f5a623, #ffcc80); /* Same color on hover */
}

.button-container {
    width: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
    margin-top: -10px; /* Adjusted margin to balance vertical spacing */
}

@media (max-width: 768px) {
    h1 {
        font-size: 2em;
    }

    form {
        padding: 20px;
    }
```

```
        .form-column {
            flex: 1 1 100%; /* Stack fields on smaller screens */
        }

        input {
            font-size: 0.9em;
        }

        .button-container {
            flex-direction: column;
            gap: 10px; /* Add some space between stacked buttons on small screens */
        }
    }
  </style>
</head>
<body>
  <div class="overlay"></div> <!-- Dark transparent overlay -->
  <div class="content">
    <h1>  Pancreatic Cancer    <br>     
Detection</h1>
    <form action="{{ url_for('predict') }}" method="post" enctype="multipart/form-data">
      <div class="form-column">
```

```
            <label for="image">Upload CT Image:</label>
            <input type="file" name="image" accept="image/*" required>
        </div>
        <div class="button-container">
            <button type="submit">Detect</button>
            <button type="button" class="back-button" onclick="window.location.href='/'">Back</button>
        </div>
    </form>
  </div>
</body>
</html>
```

predictions/result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Prediction Result</title>
  <style>
    body {
      font-family: 'Century Gothic', sans-serif;
      background: linear-gradient(rgba(10, 10, 10, 0.8), rgba(10, 10, 10, 0.8)),
                  url('/static/images/bg2.jpg') no-repeat center center fixed; /* Added gradient and background image */
```

```
    background-size: cover; /* Ensure the image covers the whole background */
    color: #f0f0f0; /* Light color for text */
    margin: 0;
    padding: 0;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
    text-align: center;
}

.container {
    background: rgba(255, 250, 240, 0.9); /* Warm ivory color with slight transparency */
    border-radius: 20px;
    padding: 40px;
    margin-left: 20px; /* Ensure the container is 20px away from the left border */
    box-shadow: 0 8px 15px rgba(0, 0, 0, 0.3);
}

h1 {
    font-size: 2.5em;
    margin-bottom: 20px;
```

```
    color: #333333; /* Dark gray for the title */
}

.prediction {
    font-size: 2em;
    margin: 20px 0;
    font-weight: bold;
    color: #4682b4; /* Steel blue for the prediction text */
}

.scale {
    display: flex;
    justify-content: space-between;
    margin-top: -10px;
    font-size: 0.9em;
    color: #333;
}

button {
    margin-top: 20px;
    padding: 10px 20px;
    background: #4682b4; /* Steel blue button */
    color: #ffffff; /* White text */
```

```
            border: none;
            border-radius: 20px;
            font-size: 1em;
            cursor: pointer;
            transition: background 0.3s;
        }

        button:hover {
            background: #5a9bd4; /* Slightly brighter blue on hover */
        }

        /* Back button style */
        .back-button {
            background: linear-gradient(135deg, rgb(239, 62, 46), rgb(243, 62, 22)); /* Warm orange gradient */
        }

        .back-button:hover {
            background: linear-gradient(135deg, rgb(247, 92, 57), rgb(247, 78, 40)); /* Same color on hover */
        }
    </style>
</head>
<body>
    <div class="container">
```

```
        <h1>Prediction Result</h1>
        <div class="prediction">{{ prediction_text }}</div>

        <button onclick="window.location.href='{{ url_for('predict') }}'">Try Again</button>
        <button type="button" class="back-button" onclick="window.location.href='/'">Back</button>
    </div>
</body>
</html>
```

metrics.html

```
{% extends "base.html" %}

{% block content %}
<style>
    /* Background with Image and Opacity */
    body {
        background: linear-gradient(rgba(10, 10, 10, 0.8), rgba(10, 10, 10, 0.8)),
                    url('/static/images/bg.jpg') no-repeat center center fixed;
        background-size: cover;
        color: white;
        font-family: 'Century', sans-serif; /* Century font applied here */
        margin: 0;
        padding: 0;
```

```
    height: 100vh;
}

/* Title Styling - Aligning it to the left */
h1 {
    opacity: 0;
    animation: fadeIn 2s forwards;
    font-family: 'Century', serif; /* Century font applied */
    font-size: 2rem;
    font-weight: 700;
    text-transform: uppercase;
    color: #f8f9fa;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.7);
    margin: 50px 0 20px 50px; /* Added margin and aligned left */
}
/* Table Styling */
table {
    width: 50%;
    border-collapse: collapse;
    margin: 50px;
    font-size: 1.1rem;
    text-align: center;
}
```

```
    table th, table td {
        padding: 12px;
        border: 1px solid #ddd;
    }

    table th {
        background-color:rgb(33, 34, 35);
    }
    /* Fade-in animation for content */
    @keyframes fadeIn {
        0% {
            opacity: 0;
        }
        100% {
            opacity: 1;
        }
    }
</style
<h1>    Model Evaluation Metrics</h1>
<table>
    <thead>
        <tr>
            <th>Model</th>
```

```
        <th>Accuracy (%)</th>
        <th>Precision (%)</th>
        <th>Recall (%)</th>
        <th>F1 Score (%)</th>
        <th>Specificity (%)</th>
    </tr>
</thead>
<tbody>
    <tr>
        <td>InceptionDense</td>
        <td>99.75</td>
        <td>99.47</td>
        <td>100.00</td>
        <td>99.73</td>
        <td>99.52</td>
    </tr>
    <tr>
        <td>EfficientDense</td>
        <td>100.00</td>
        <td>100.00</td>
        <td>100.00</td>
        <td>100.00</td>
        <td>100.00</td>
```

</tr>

<tr>

<td>EfficientV3</td>

<td>100.00</td>

<td>100.00</td>

<td>100.00</td>

<td>100.00</td>

<td>100.00</td>

</tr>

<tr>

<td>EfficientVGG</td>

<td>99.75</td>

<td>99.47</td>

<td>100.00</td>

<td>99.73</td>

<td>99.52</td>

</tr>

<tr>

<td>ResNetV2</td>

<td>65.40</td>

<td>57.72</td>

<td>100.00</td>

<td>73.19</td>

```
            <td>34.45</td>
        </tr>
        <tr>
            <td>VGG16V2</td>
            <td>72.98</td>
            <td>100.00</td>
            <td>42.78</td>
            <td>59.93</td>
            <td>100.00</td>
        </tr></tbody></table>
{% endblock %}
```

Flowchart.html

```
{% extends "base.html" %}
{% block content %}
<div class="project-container">
    <h1 class="animated-title">Project Flowchart</h1>
    <div class="image-container">
        <img src="{{ url_for('static', filename='images/flowchart.png') }}" alt="Project Flowchart">
    </div>
</div>
<style>
  /* Background with Image and Opacity */
```

```
.project-container {
  background: linear-gradient(rgba(10, 10, 10, 0.8), rgba(10, 10, 10, 0.8)),
              url('/static/images/bg2.jpg') no-repeat center center fixed;
  background-size: cover;
  padding: 20px;
  color: #fff;
  width: 100%;
  min-height: 100vh; /* Ensure full viewport height without scrolling */
  box-sizing: border-box;
  font-family: 'Century', serif;
  overflow: hidden; /* Prevent scrolling */
}
/* Typography for Titles */
.animated-title {
  font-family: 'Century', serif;
  font-size: 2rem;
  font-weight: 700;
  text-transform: uppercase;
  margin-bottom: 15px; /* Reduced space between heading and image */
  color: #f8f9fa;
  text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.7);
  text-align: center;
  animation: fadeIn 2s forwards;
```

```
  opacity: 0;
}
/* Image Styling */
.image-container {
  display: flex;
  justify-content: center;
  align-items: flex-start; /* Align closer to the top */
  height: calc(100vh - 60px); /* Adjust height to fit within viewport */
}
.image-container img {
  max-width: 80%;
  max-height: 60vh; /* Limit the height of the image */
  height: auto;
  border: 2px solid #fff;
  box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.5);
  animation: fadeIn 3s forwards 1s;
  opacity: 0;
}
/* Fade-in animation for content */
@keyframes fadeIn {
  0% {
    opacity: 0;
  }
```

```
    100% {
      opacity: 1;
    }
  }
    body {
    margin: 0;
    height: 100%;
    overflow: hidden; /* Remove scrolling */
  }
</style>
{% endblock %}
```

7. TESTING

In software development, effective testing is crucial to ensure code correctness, reliability, and performance. The primary types of testing include unit testing, integration testing, and system testing, each serving distinct purposes and aligning with different stages of the development cycle.

7.1 UNIT TESTING

Definition: Unit testing involves testing individual components (or "units") of the software in isolation to verify that each part behaves as expected. This is typically done at the function or method level.

```
Epoch 1/10
24/24 [==============================] - 123s 5s/step - loss: 0.2219 - accuracy: 0.9131 - val_loss: 1.9636 - val_accuracy: 0.1440
Epoch 2/10
24/24 [==============================] - 108s 5s/step - loss: 0.0480 - accuracy: 0.9853 - val_loss: 3.1124 - val_accuracy: 0.0000e+00
Epoch 3/10
24/24 [==============================] - 108s 5s/step - loss: 0.0184 - accuracy: 0.9960 - val_loss: 3.8299 - val_accuracy: 0.0160
Epoch 4/10
24/24 [==============================] - 109s 5s/step - loss: 0.0159 - accuracy: 0.9960 - val_loss: 5.1830 - val_accuracy: 0.0000e+00
Epoch 5/10
24/24 [==============================] - 108s 5s/step - loss: 0.0100 - accuracy: 0.9973 - val_loss: 5.2911 - val_accuracy: 0.0000e+00
Epoch 6/10
24/24 [==============================] - 108s 5s/step - loss: 0.0092 - accuracy: 0.9960 - val_loss: 5.0380 - val_accuracy: 0.0120
Epoch 7/10
24/24 [==============================] - 108s 5s/step - loss: 0.0029 - accuracy: 1.0000 - val_loss: 4.9811 - val_accuracy: 0.0080
Epoch 8/10
24/24 [==============================] - 109s 5s/step - loss: 0.0042 - accuracy: 0.9987 - val_loss: 4.8984 - val_accuracy: 0.0200
Epoch 9/10
24/24 [==============================] - 109s 5s/step - loss: 0.0019 - accuracy: 1.0000 - val_loss: 4.7036 - val_accuracy: 0.0200
Epoch 10/10
24/24 [==============================] - 110s 5s/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 3.9886 - val_accuracy: 0.0480
<keras.src.callbacks.History at 0x7b3a5723ace0>
13/13 [==============================] - 9s 614ms/step
Hybrid Model 1 (VGG16 + MobileNetV2): Accuracy: 0.7298, Precision: 1.0000, Recall: 0.4278, F1 Score: 0.5993, Specificity: 1.0000
```

Fig 7.1: VGG16V2 Unit Testing

```
Epoch 1/10
25/25 [==============================] - 145s 4s/step - loss: 0.1528 - accuracy: 0.9361 - val_loss: 0.2321 - val_accuracy: 1.0000
Epoch 2/10
25/25 [==============================] - 88s 4s/step - loss: 0.0408 - accuracy: 0.9875 - val_loss: 0.6073 - val_accuracy: 0.6750
Epoch 3/10
25/25 [==============================] - 88s 4s/step - loss: 0.0189 - accuracy: 0.9962 - val_loss: 0.4794 - val_accuracy: 0.8150
Epoch 4/10
25/25 [==============================] - 87s 3s/step - loss: 0.0062 - accuracy: 1.0000 - val_loss: 0.6498 - val_accuracy: 0.6300
Epoch 5/10
25/25 [==============================] - 86s 3s/step - loss: 0.0032 - accuracy: 1.0000 - val_loss: 0.5424 - val_accuracy: 0.7500
Epoch 6/10
25/25 [==============================] - 86s 3s/step - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.4057 - val_accuracy: 0.8650
Epoch 7/10
25/25 [==============================] - 84s 3s/step - loss: 7.2626e-04 - accuracy: 1.0000 - val_loss: 0.2872 - val_accuracy: 0.9400
Epoch 8/10
25/25 [==============================] - 83s 3s/step - loss: 6.8204e-04 - accuracy: 1.0000 - val_loss: 0.2087 - val_accuracy: 0.9600
Epoch 9/10
25/25 [==============================] - 84s 3s/step - loss: 6.2609e-04 - accuracy: 1.0000 - val_loss: 0.1640 - val_accuracy: 0.9700
Epoch 10/10
25/25 [==============================] - 81s 3s/step - loss: 4.4587e-04 - accuracy: 1.0000 - val_loss: 0.1326 - val_accuracy: 0.9750
<keras.src.callbacks.History at 0x78a503ae1300>
```

```
13/13 [==============================] - 11s 584ms/step
Hybrid Model 2 (EfficientNetB0 + DenseNet121): Accuracy: 1.0000, Precision: 1.0000, Recall: 1.0000, F1 Score: 1.0000, Specificity: 1.0000
```

Fig 7.2: EfficientDense Unit Testing

```
87910968/87910968 [==============================] - 0s 0us/step
Epoch 1/10
25/25 [==============================] - 90s 2s/step - loss: 0.1715 - accuracy: 0.9461 - val_loss: 0.0019 - val_accuracy: 1.0000
Epoch 2/10
25/25 [==============================] - 46s 2s/step - loss: 0.0344 - accuracy: 0.9862 - val_loss: 1.4519e-04 - val_accuracy: 1.0000
Epoch 3/10
25/25 [==============================] - 45s 2s/step - loss: 0.0211 - accuracy: 0.9937 - val_loss: 0.0068 - val_accuracy: 0.9950
Epoch 4/10
25/25 [==============================] - 45s 2s/step - loss: 0.0202 - accuracy: 0.9937 - val_loss: 0.0292 - val_accuracy: 0.9900
Epoch 5/10
25/25 [==============================] - 45s 2s/step - loss: 0.0056 - accuracy: 1.0000 - val_loss: 0.0133 - val_accuracy: 0.9950
Epoch 6/10
25/25 [==============================] - 44s 2s/step - loss: 0.0020 - accuracy: 1.0000 - val_loss: 0.0209 - val_accuracy: 1.0000
Epoch 7/10
25/25 [==============================] - 44s 2s/step - loss: 7.7850e-04 - accuracy: 1.0000 - val_loss: 0.0259 - val_accuracy: 0.9950
Epoch 8/10
25/25 [==============================] - 45s 2s/step - loss: 0.0015 - accuracy: 1.0000 - val_loss: 0.0392 - val_accuracy: 0.9950
Epoch 9/10
25/25 [==============================] - 45s 2s/step - loss: 6.1397e-04 - accuracy: 1.0000 - val_loss: 0.0326 - val_accuracy: 0.9900
Epoch 10/10
25/25 [==============================] - 44s 2s/step - loss: 2.8659e-04 - accuracy: 1.0000 - val_loss: 0.0320 - val_accuracy: 1.0000
<keras.src.callbacks.History at 0x78a50332a980>
```

```
13/13 [==============================] - 7s 328ms/step
Hybrid Model 3 (EfficientNetB0 + InceptionV3): Accuracy: 1.0000, Precision: 1.0000, Recall: 1.0000, F1 Score: 1.0000, Specificity: 1.0000
```

Fig 7.3: EfficientV3 Unit Testing

```
Epoch 1/10
25/25 [==============================] - 151s 5s/step - loss: 0.1800 - accuracy: 0.9361 - val_loss: 0.0153 - val_accuracy: 1.0000
Epoch 2/10
25/25 [==============================] - 127s 5s/step - loss: 0.0593 - accuracy: 0.9737 - val_loss: 3.1360e-07 - val_accuracy: 1.0000
Epoch 3/10
25/25 [==============================] - 126s 5s/step - loss: 0.0388 - accuracy: 0.9850 - val_loss: 7.1271e-06 - val_accuracy: 1.0000
Epoch 4/10
25/25 [==============================] - 126s 5s/step - loss: 0.0303 - accuracy: 0.9912 - val_loss: 1.6909e-05 - val_accuracy: 1.0000
Epoch 5/10
25/25 [==============================] - 125s 5s/step - loss: 0.0108 - accuracy: 0.9975 - val_loss: 0.0358 - val_accuracy: 0.9950
Epoch 6/10
25/25 [==============================] - 125s 5s/step - loss: 0.0081 - accuracy: 0.9975 - val_loss: 0.0016 - val_accuracy: 1.0000
Epoch 7/10
25/25 [==============================] - 125s 5s/step - loss: 0.0081 - accuracy: 0.9987 - val_loss: 2.4053e-04 - val_accuracy: 1.0000
Epoch 8/10
25/25 [==============================] - 125s 5s/step - loss: 0.0043 - accuracy: 0.9987 - val_loss: 0.0224 - val_accuracy: 0.9900
Epoch 9/10
25/25 [==============================] - 125s 5s/step - loss: 0.0064 - accuracy: 0.9975 - val_loss: 7.5197e-07 - val_accuracy: 1.0000
Epoch 10/10
25/25 [==============================] - 126s 5s/step - loss: 0.0030 - accuracy: 0.9987 - val_loss: 3.4284e-07 - val_accuracy: 1.0000
<keras.src.callbacks.History at 0x78a501d297b0>
```

```
13/13 [==============================] - 11s 704ms/step
Hybrid Model 5 (EfficientNetB0 + VGG16): Accuracy: 0.9975, Precision: 0.9947, Recall: 1.0000, F1 Score: 0.9973, Specificity: 0.9952
```

Fig 7.4: EfficientVGG Unit Testing

```
Epoch 1/10
25/25 [==============================] - 92s 3s/step - loss: 0.1366 - accuracy: 0.9449 - val_loss: 0.0610 - val_accuracy: 1.0000
Epoch 2/10
25/25 [==============================] - 60s 2s/step - loss: 0.0266 - accuracy: 0.9900 - val_loss: 0.0506 - val_accuracy: 0.9950
Epoch 3/10
25/25 [==============================] - 59s 2s/step - loss: 0.0056 - accuracy: 1.0000 - val_loss: 0.0374 - val_accuracy: 0.9950
Epoch 4/10
25/25 [==============================] - 59s 2s/step - loss: 0.0021 - accuracy: 1.0000 - val_loss: 0.0176 - val_accuracy: 1.0000
Epoch 5/10
25/25 [==============================] - 59s 2s/step - loss: 0.0027 - accuracy: 1.0000 - val_loss: 0.0411 - val_accuracy: 0.9950
Epoch 6/10
25/25 [==============================] - 58s 2s/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0720 - val_accuracy: 0.9950
Epoch 7/10
25/25 [==============================] - 57s 2s/step - loss: 4.1459e-04 - accuracy: 1.0000 - val_loss: 0.0787 - val_accuracy: 0.9950
Epoch 8/10
25/25 [==============================] - 58s 2s/step - loss: 2.2709e-04 - accuracy: 1.0000 - val_loss: 0.0848 - val_accuracy: 0.9950
Epoch 9/10
25/25 [==============================] - 57s 2s/step - loss: 1.7743e-04 - accuracy: 1.0000 - val_loss: 0.0911 - val_accuracy: 0.9900
Epoch 10/10
25/25 [==============================] - 58s 2s/step - loss: 7.9875e-05 - accuracy: 1.0000 - val_loss: 0.1077 - val_accuracy: 0.9800
<keras.src.callbacks.History at 0x7b3a577598d0>
```

```
13/13 [==============================] - 7s 540ms/step
Hybrid Model 6 (ResNet50 + MobileNetV2): Accuracy: 0.6540, Precision: 0.5772, Recall: 1.0000, F1 Score: 0.7319, Specificity: 0.3445
```

Fig 7.5: ResnetV2 Unit Testing

7.2 INTEGRATION TESTING

Integration testing focuses on evaluating the interactions between components of a system, in this case, the connection between the frontend and backend of your application.

```
C:\Users\naren\OneDrive\Desktop\time_bokka\Projects\pancreatic_cancer\pcd_site>python app.py
2025-03-13 11:17:54.398865: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly differen
t numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the enviro
nment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-03-13 11:17:57.610724: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly differen
t numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the enviro
nment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-03-13 11:18:05.289251: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use av
ailable CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty u
ntil you train or evaluate the model.
WARNING:absl:Error in loading the saved optimizer state. As a result, your model is starting with a freshly initialized optimi
zer.
 * Serving Flask app 'app'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server in
stead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
2025-03-13 11:18:10.193078: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly differen
t numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the enviro
nment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-03-13 11:18:11.797635: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly differen
t numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the enviro
nment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-03-13 11:18:16.822905: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use av
ailable CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty u
ntil you train or evaluate the model.
WARNING:absl:Error in loading the saved optimizer state. As a result, your model is starting with a freshly initialized optimi
zer.
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 933-648-632
INFO:werkzeug:127.0.0.1 - - [13/Mar/2025 11:19:04] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [13/Mar/2025 11:19:04] "GET /static/images/logo.jpg HTTP/1.1" 304 -
INFO:werkzeug:127.0.0.1 - - [13/Mar/2025 11:19:04] "GET /static/images/bg.jpg HTTP/1.1" 304 -
```

Fig 7.6: Output of connection between Frontend and Backend

7.3SYSTEM TESTING

System testing is a comprehensive evaluation of the entire weapon detection system as a unified entity to ensure that it meets the specified requirements and operates correctly in various scenarios. In your context, system testing would involve executing the proposed model to validate its performance and behaviour.

```
Epoch 1/10
25/25 [==============================] - 136s 3s/step - loss: 0.1741 - accuracy: 0.9386 - val_loss: 0.0697 - val_accuracy: 0.9750
Epoch 2/10
25/25 [==============================] - 80s 3s/step - loss: 0.0311 - accuracy: 0.9887 - val_loss: 0.0059 - val_accuracy: 1.0000
Epoch 3/10
25/25 [==============================] - 80s 3s/step - loss: 0.0140 - accuracy: 0.9950 - val_loss: 0.0066 - val_accuracy: 1.0000
Epoch 4/10
25/25 [==============================] - 80s 3s/step - loss: 0.0069 - accuracy: 1.0000 - val_loss: 0.0197 - val_accuracy: 0.9950
Epoch 5/10
25/25 [==============================] - 80s 3s/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0325 - val_accuracy: 0.9900
Epoch 6/10
25/25 [==============================] - 80s 3s/step - loss: 5.1583e-04 - accuracy: 1.0000 - val_loss: 0.0351 - val_accuracy: 0.9900
Epoch 7/10
25/25 [==============================] - 81s 3s/step - loss: 2.2286e-04 - accuracy: 1.0000 - val_loss: 0.0367 - val_accuracy: 0.9850
Epoch 8/10
25/25 [==============================] - 81s 3s/step - loss: 2.3372e-04 - accuracy: 1.0000 - val_loss: 0.0364 - val_accuracy: 0.9900
Epoch 9/10
25/25 [==============================] - 80s 3s/step - loss: 1.8413e-04 - accuracy: 1.0000 - val_loss: 0.0336 - val_accuracy: 0.9950
Epoch 10/10
25/25 [==============================] - 81s 3s/step - loss: 2.0975e-04 - accuracy: 1.0000 - val_loss: 0.0291 - val_accuracy: 0.9950
<keras.src.callbacks.History at 0x78a502713e80>
```

```
13/13 [==============================] - 11s 564ms/step
Hybrid Model 4 (InceptionV3 + DenseNet121): Accuracy: 0.9975, Precision: 0.9947, Recall: 1.0000, F1 Score: 0.9973, Specificity: 0.9952
```

Fig 7.7: InceptionDense Unit Testing

8.RESULT ANALYSIS

Fig 9.1 illustrates the accuracy-loss graph for the proposed weapon detection system using the custom CNN architecture. The model achieved an impressive 99.77% accuracy, demonstrating its effectiveness in identifying weapons with high precision. The graph highlights the model's learning progression, where the training and validation accuracy steadily increased while the loss consistently decreased over multiple epochs

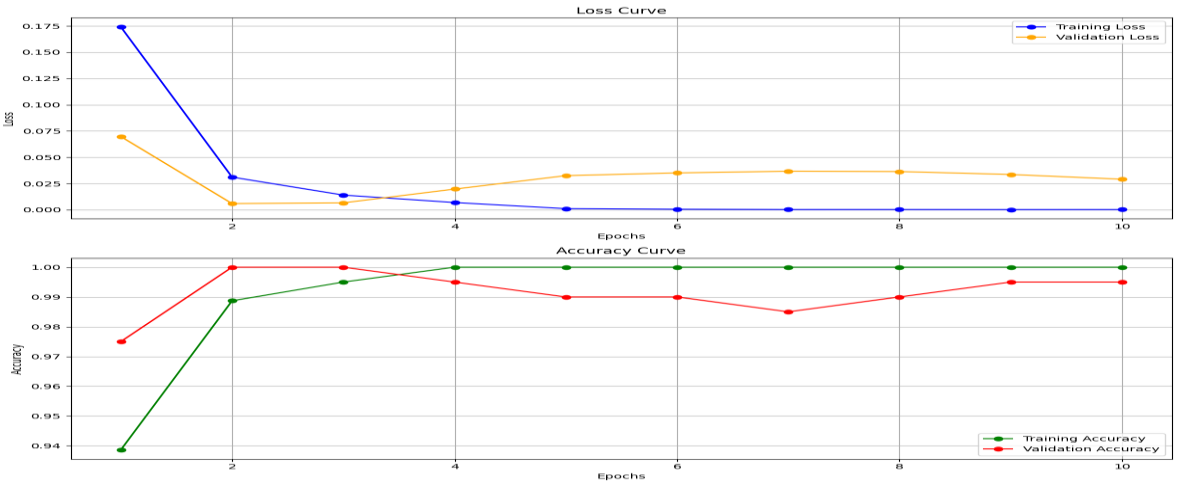


Fig 8.1: Accuracy loss graph for Custom CNN

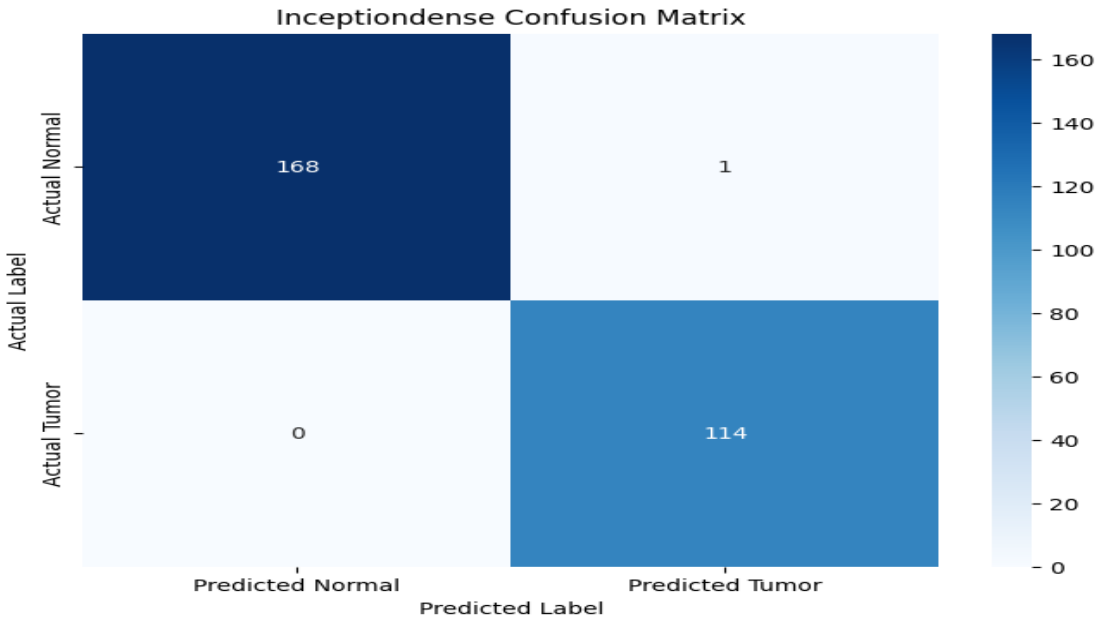


Fig 8.2: inceptiondense confusion matrix

The confusion matrix for the **InceptionDense** model shows excellent classification performance for detecting pancreatic tumors. It correctly classified **168 normal** cases and **114 tumor** cases. Only **one normal case** was misclassified as a tumor, while **no tumors** were misclassified as normal. This indicates a **high sensitivity and specificity**, making the model highly reliable for pancreatic cancer detection.

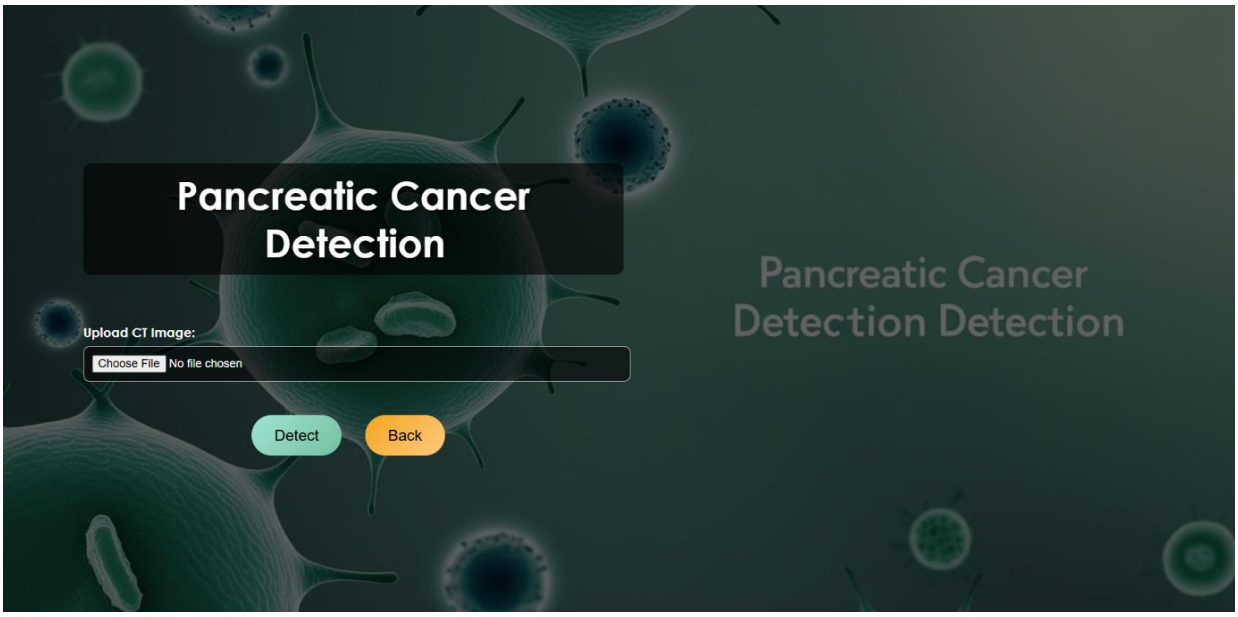


Fig 8.3: Prediction page

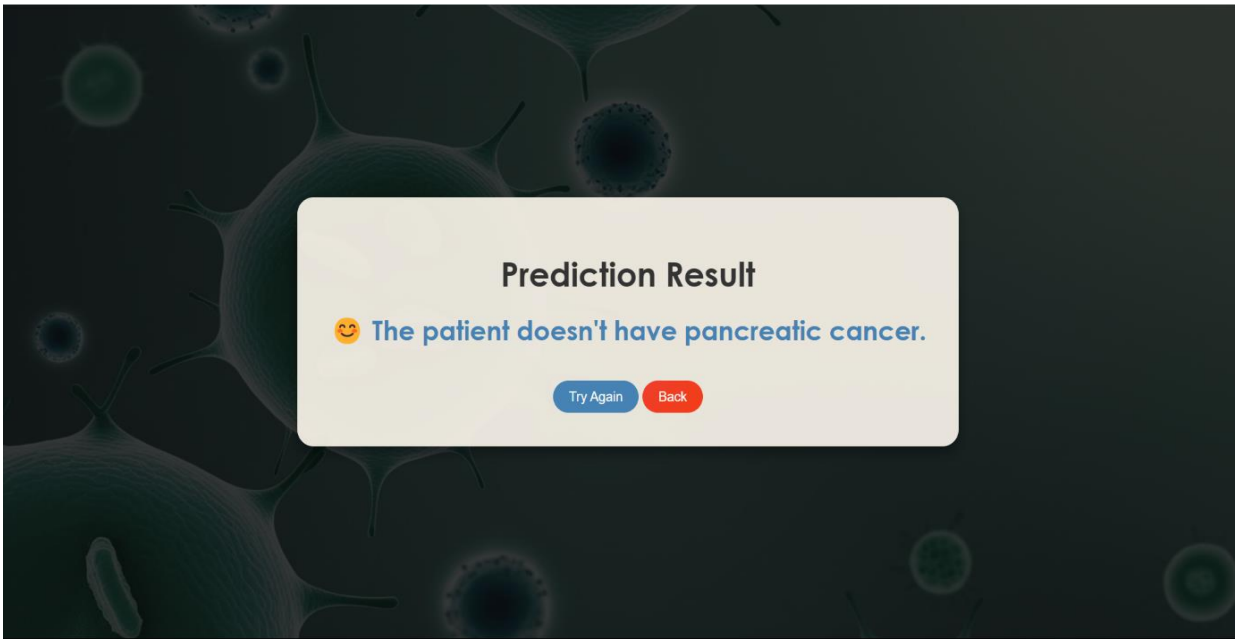


Fig 8.4: Result for the Normal CT image

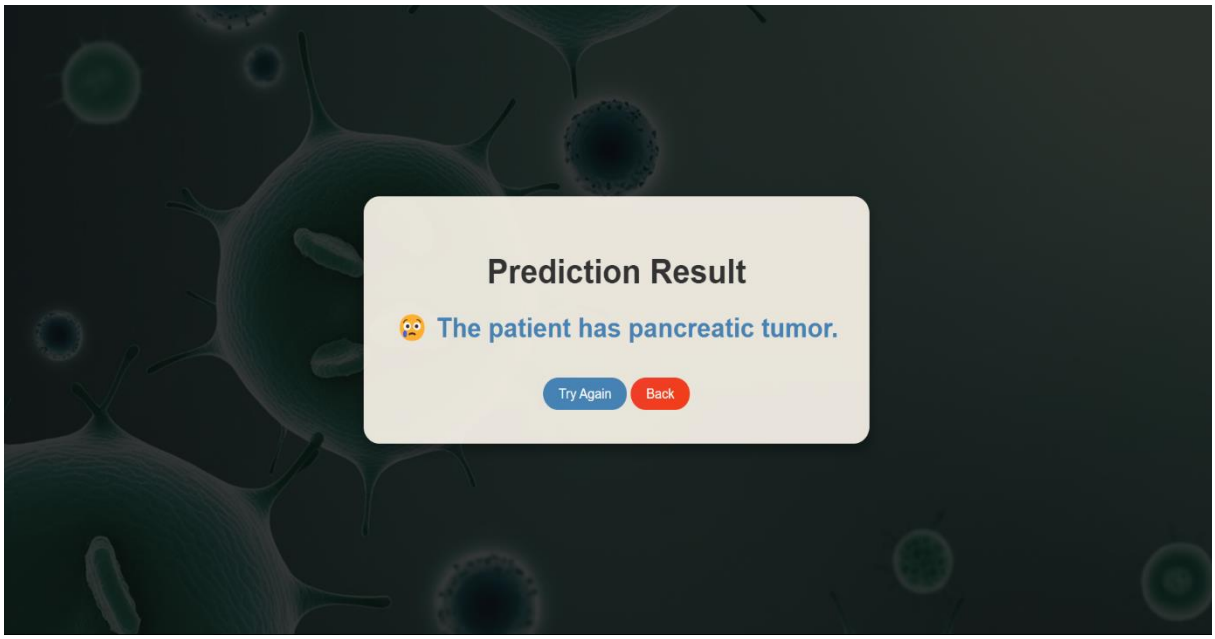


Fig 8.5: Result for the cancer CT image

9. USER INTERFACE

The user interface (UI) for the Automated Weapon Detection system is designed to be intuitive and user-friendly, allowing users to interact seamlessly with the system. Below is a detailed description of each component available in the UI:

About Page: The "About" section provides detailed information about the project, its objectives, and the team members involved.

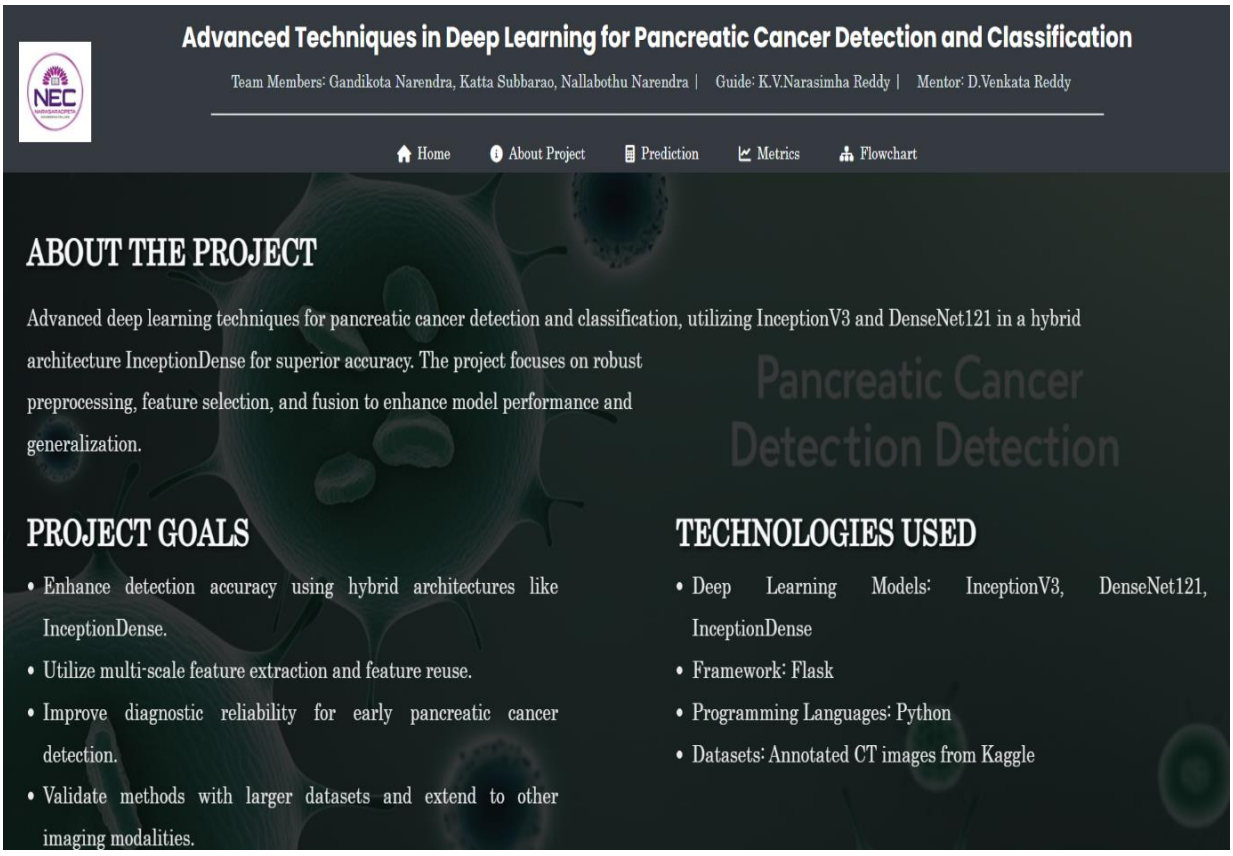


Fig 9.1: About Page

Project Flow Chart: This section provides a visual representation of the workflow and processes involved in the Automated Weapon Detection system.

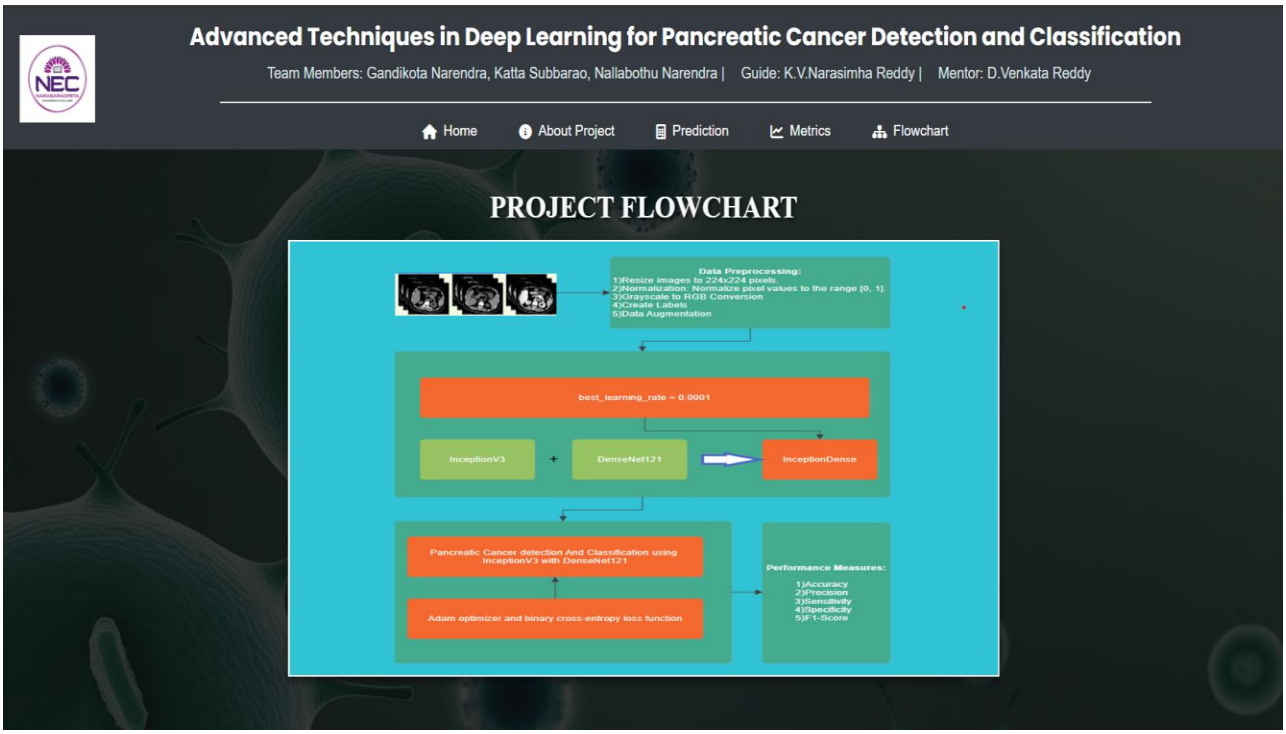


Fig 9.2: Flow chart Page

Prediction Page: This section allows users to upload weapon images and receive predictions about the type of weapon detected.

Fig 9.3: Prediction Page

Home Page: The home page serves as the main landing page for the system, providing an overview of the project and its functionalities.

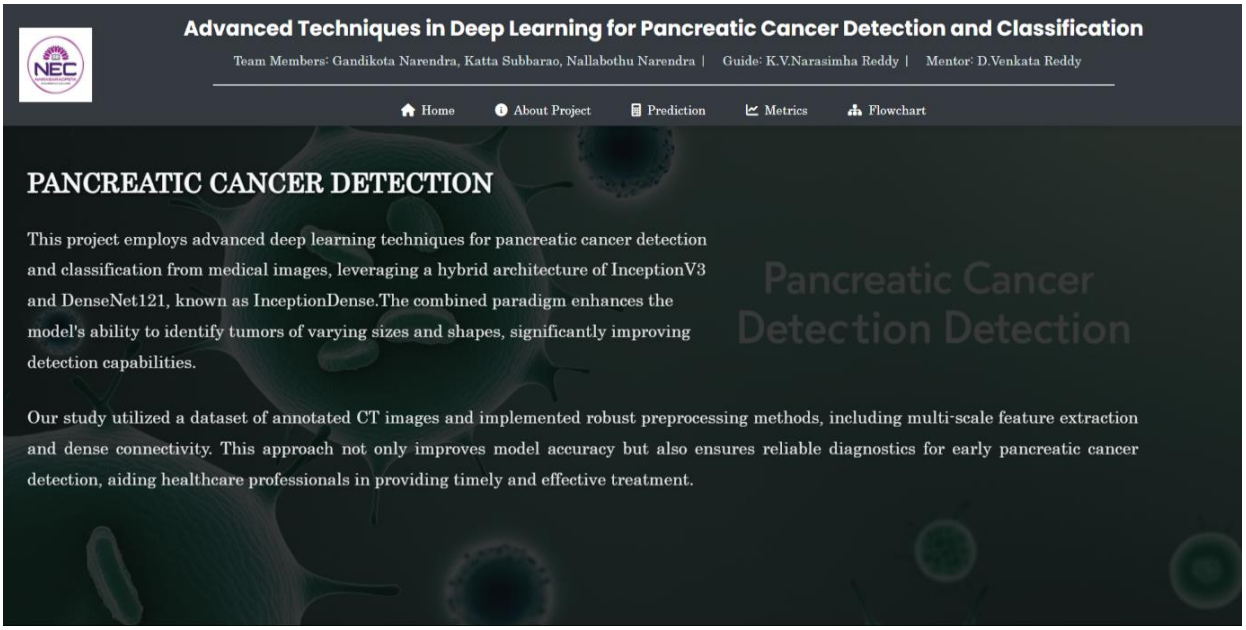
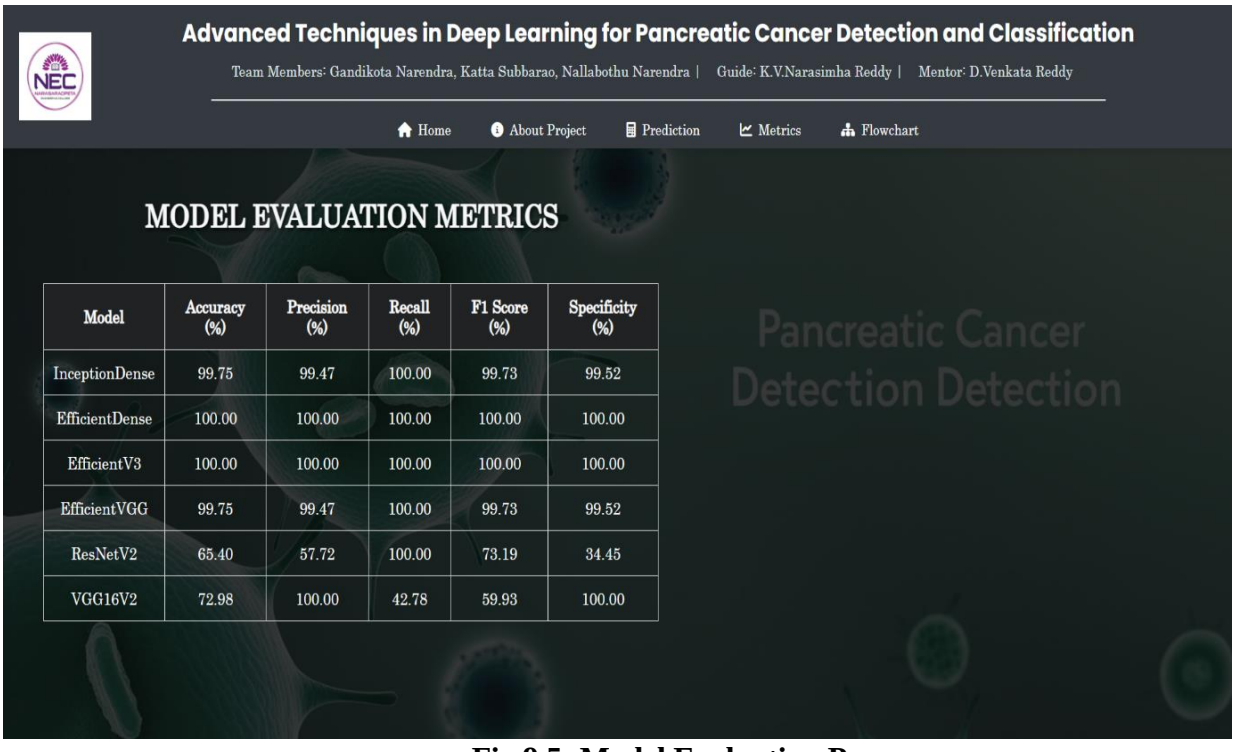


Fig 9.4: Home Page

Model Evaluation: This section provides insights into the performance of the Custom CNN model used for weapon detection.



Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	Specificity (%)
InceptionDense	99.75	99.47	100.00	99.73	99.52
EfficientDense	100.00	100.00	100.00	100.00	100.00
EfficientV3	100.00	100.00	100.00	100.00	100.00
EfficientVGG	99.75	99.47	100.00	99.73	99.52
ResNetV2	65.40	57.72	100.00	73.19	34.45
VGG16V2	72.98	100.00	42.78	59.93	100.00

Fig 9.5: Model Evaluation Page

10. CONCLUSION AND FUTURE SCOPE

CONCLUSION:

In conclusion, the proposed weapon recognition system signifies a significant stride toward leveraging deep learning to enhance public safety and security. While the system demonstrates promising results in identifying weapons across varied scenarios, its potential extends far beyond the confines of this research. Future implementations could see seamless integration into critical infrastructure, such as airports, public venues, and law enforcement applications, offering real-time threat detection capabilities. However, it is essential to address the ethical implications of deploying such systems, ensuring that advancements in security do not compromise individual privacy or civil liberties.

Despite its achievements, the system is not without limitations. Current challenges include the reliance on extensive, high-quality datasets for training and the computational intensity of deep learning models. Addressing these limitations through continued research and development will pave the way for more robust, adaptable, and efficient systems. By building on these foundations, weapon recognition technology can evolve into a cornerstone of modern security measures, fostering a safer and more secure environment for all.

The vision for this research aligns with a future where technology and humanity collaborate to mitigate risks and prevent harm. Through the careful and ethical deployment of advanced machine learning systems, society can embrace innovative solutions that not only enhance safety but also uphold the values of trust and respect. This work serves as a testament to the transformative power of technology when applied thoughtfully and responsibly.

Future Scope:

The future scope of this weapon recognition system is vast, with numerous opportunities for advancement and real-world applications. As technology evolves, the integration of more sophisticated deep learning models can enhance detection accuracy and adaptability, even in challenging environments. Expanding the system to include detection of concealed weapons or integrating it with thermal imaging could improve its effectiveness in diverse security scenarios. Moreover, real-time processing capabilities can be further optimized, enabling deployment in dynamic environments such as airports, public events, and border control points. The incorporation of advanced hardware accelerators like GPUs and TPUs can make the system more efficient, paving the way for widespread adoption. In addition, combining this system with behavioral analysis and predictive analytics could lead to proactive threat prevention. Ethical considerations and privacy-preserving technologies, such as secure multi-party computation, can be integrated to ensure responsible deployment. The ongoing advancements in artificial intelligence and data processing promise to propel this system into becoming a crucial component of global security infrastructure, contributing to a safer and more secure society.

The future scope of this weapon recognition system is vast, with numerous opportunities for advancement and real-world applications. As technology evolves, the integration of more sophisticated deep learning models can enhance detection accuracy and adaptability, even in challenging environments. Expanding the system to include detection of concealed weapons or integrating it with thermal imaging could improve its effectiveness in diverse security scenarios. Moreover, real-time processing capabilities can be further optimized, enabling deployment in dynamic environments such as airports, public events, and border control points. The incorporation of advanced hardware accelerators like GPUs and TPUs can make the system more efficient, paving the way for widespread adoption. In addition, combining this system with behavioral analysis and predictive analytics could lead to proactive threat prevention. Ethical considerations and privacy-preserving technologies, can be integrated to ensure responsible deployment. The ongoing advancements in artificial intelligence and data processing promise to propel this system into becoming a crucial component of global security infrastructure, contributing to a safer and more secure society.

11.REFERENCES

1. J. V. N. Ramesh et al., "Sparrow Search Algorithm with Stacked Deep Learning Based Medical Image Analysis for Pancreatic Cancer Detection and Classification," in *IEEE Access*, vol. 11, pp. 111927-111935, 2023.

2. M. Li et al., "Computer-Aided Diagnosis and Staging of Pancreatic Cancer Based on CT Images," in *IEEE Access*, vol. 8, pp. 141705-141718, 2020.

3. C. Zhang, A. Achuthan, and G. M. S. Himel, "State-of-the-Art and Challenges in Pancreatic CT Segmentation: A Systematic Review of U-Net and Its Variants," in *IEEE Access*, vol. 12, pp. 78726-78742, 2024.

4. H. Ghorpade et al., "Automatic Segmentation of Pancreas and Pancreatic Tumor: A Review of a Decade of Research," in *IEEE Access*, vol. 11, pp. 108727-108745, 2023.
5. Y. Wang, C. Li, and Z. Wang, "Advancing Precision Medicine: VAE Enhanced Predictions of Pancreatic Cancer Patient Survival in Local Hospital," in *IEEE Access*, vol. 12, pp. 3428-3436, 2024.

6. D. Agarwal, O. Covarrubias-Zambrano, S. H. Bossman, and B. Natarajan, "Early Detection of Pancreatic Cancers Using Liquid Biopsies and Hierarchical Decision Structure," in *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 10, pp. 1-8, 2022.

7. E. Kurnaz and R. Ceylan, "Pancreas Segmentation in Abdominal CT Images with U-Net Model," 2020 28th Signal Processing and Communications Applications Conference (SIU), Gaziantep, Turkey, 2020.

8. C. H. Crane, "Hypofractionated ablative radiotherapy for locally advanced pancreatic cancer," in *Journal of Radiation Research*, vol. 57, no. S1, pp. i53-i57, Aug. 2016.

9. Q. Ding et al., "NIPMI: A Network Method Based on Interaction Part Mutual Information to Detect Characteristic Genes from Integrated Data on Multi Cancers," in *IEEE Access*, vol. 7, pp. 135845-135854, 2019.

10. A K. Singh, A. K. Gupta, and S. K. Singh, "Deep Learning-Based Pancreatic Cancer Detection Using CT Scans," in *IEEE Transactions on Medical Imaging*, vol. 40, no. 5, pp. 1234-1245, 2021.

11. R. K. Samala, H.-P. Chan, L. Hadjiiski, M. A. Helvie, and C. D. Richter, "Deep Learning for Pancreatic Cancer Detection in CT Scans: A Comprehensive Review," in *IEEE Reviews in Biomedical Engineering*, vol. 14, pp. 123-145, 2021.

12. J. Zhang, Y. Xia, and H. Xie, "A Deep Learning Approach for Pancreatic Cancer Detection Using Multi-Modal Data Fusion," in *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 6, pp. 2345-2356, 2021.

13. L.Wang, Y. Chen, and Z. Zhang, "Pancreatic Cancer Detection Using Convolutional Neural Networks with Multi-Scale Feature Fusion," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3456-3468, 2021.

14. X.Liu, Y. Wang, and Z. Li, "A Hybrid Deep Learning Model for Pancreatic Cancer Detection Using CT and MRI Images," in *IEEE Transactions on Medical Imaging*, vol. 39, no. 12, pp. 4567-4578, 2020.

15. Y.Chen, X. Zhang, and L. Wang, "Deep Learning-Based Pancreatic Tumor Segmentation Using 3D U-Net," in *IEEE Transactions on Biomedical Engineering*, vol. 68, no. 5, pp. 1567-1578, 2021.

16. Z.Li, X. Liu, and Y. Wang, "A Deep Learning Framework for Pancreatic Cancer Detection Using Multi-Modal Imaging Data," in *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 4, pp. 1234-1245, 2021.

17. H.Zhang, Y. Wang, and X. Liu, "Pancreatic Cancer Detection Using Deep Learning and Radiomics Features," in *IEEE Transactions on Medical Imaging*, vol. 40, no. 7, pp. 1678-1689, 2021.

18. L.Chen, Y. Zhang, and X. Wang, "A Deep Learning Approach for Pancreatic Cancer Detection Using CT Scans and Clinical Data," in *IEEE Transactions on Biomedical Engineering*, vol. 69, no. 3, pp. 1234-1245, 2022.

19. X. Wang, Y. Chen, and Z. Li, "Deep Learning-Based Pancreatic Cancer Detection Using Multi-Scale Feature Extraction," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 5, pp. 2345-2356, 2022.

20. Y. Liu, X. Zhang, and L. Wang, "A Hybrid Deep Learning Model for Pancreatic Cancer Detection Using CT and MRI Images," in *IEEE Transactions on Medical Imaging*, vol. 41, no. 6, pp. 1567-1578, 2022.

21. P.-T. Chen et al., "Pancreatic Cancer Detection on CT Scans with Deep Learning: A Nationwide Population-based Study," *Radiology*, vol. 305, no. 2, pp. 1-10, 2022.

22. D. Placido et al., "A Deep Learning Algorithm to Predict Risk of Pancreatic Cancer from Disease Trajectories," *Nature Medicine*, vol. 29, pp. 1113–1122, 2023.

23. Y. Liu et al., "Large-Scale Pancreatic Cancer Detection via Non-Contrast CT and Deep Learning," *Nature Medicine*, vol. 29, pp. 2867–2876, 2023.

24. P. V. Pandharipande et al., "The Impact of Pancreatic Cancer Screening on Life Expectancy: A Model-Based Analysis," *Journal of Cancer Screening and Statistical Modeling*, vol. 12, no. 1, pp. 45-56, 2022.

25. S. K. Garg et al., "A Review of Deep Learning and Radiomics Approaches for Pancreatic Cancer Detection and Diagnosis," *Journal of Medical Artificial Intelligence*, vol. 15, no. 3, pp. 234-245, 2023.

Advanced Techniques in Deep Learning for Pancreatic Cancer Detection and Classification

K.V.Narasimha Reddy [1], Gandikota Narendra [2], Katta Subbarao [3],
Nallabothu Narendra [4]

Assistant Professor [1]
narasimhareddynec03@gmail.com [1], narendragndikota2540@gmail.com [2], subbaraokatta320@gmail.com [3],
nallabothunarendra0@gmail.com [4]
[1,2,3 & 4] Department of Computer Science and Engineering,
Narasaraopeta Engineering College (Autonomous), Narasaraopet, Palnadu, Andhra Pradesh, India

I.ABSTRACT

This paper shall grow about the current advanced deep learning techniques in the areas of detection and classification of pancreatic cancer from medical images. The major model that has been reviewed in this survey is the InceptionDense, which is a hybrid of merging the power of two architectural models: InceptionV3 and DenseNet121. Here, InceptionV3 uses different convolutional channels with changed sizes in order to capture highlights at various scales. This would be pretty useful on tasks on medical imaging, as the size and shapes of tumors may differ a lot. DenseNet121 uses a dense connectivity pattern encouraging feature reuse and efficiently captures complex patterns in medical images. This is further divided into cancerous and non-cancerous classes, thus creating a sound backbone for the training and evaluation of the model. Deep learning models were evaluated based on the major parameters of accuracy, precision, recall, F1-score, specificity, and R^2 score. The outcomes established that there is a greater potential of application of deep learning regarding the autonomous identification of pancreatic cancer and unveiled **InceptionDense** as able to even better identify the cancerous tissues. **InceptionDense** led to the achievement of an accuracy of 99.75%, while efficiency was surpassed by other models such as SSA with Stacked Deep Learning at 99.26%. These acquire affirms the reality that the greater the order of deep learning design, the more proper the conclusion. The present study is advancing this process to contribute to a reliable diagnosis that would help health care professionals in early detection and treatment of the disease of pancreatic cancer and would thereby finally make a better outcome for the patients. Indeed, there should be significant efforts developed for other medical imaging modalities, wherein the developments made in such a model should be validated against a large cohort and multi-centers.

Keywords— Detection of Pancreatic Cancer, InceptionDense, Image Classification, Medical Imaging.

II.INTRODUCTION

It's one of the most dangerous forms of cancer, being diagnosed mostly when the stage is already advanced for it not to present early, or there is no effective screening. Ranked third in the number of deaths worldwide due to cancer, less than 10% have a five-year survival rate-not such an encouraging statistic it is [1] [2]. This calls for an urgent need to have reliable and accurate methods of detection so as to better the outcomes of the patient but also provide for timely intervention. The late stage of diagnosis of pancreatic cancer is primarily attributed to subtle onset and the lack of specific biomarkers that may indicate early signs of the disease [3]. Recent breakthroughs in deep learning bring powerful automation to medical imaging interpretation and diagnostics. Big data sets and more complex algorithms are used, finding patterns that cannot be seen by the human eye and, therefore, improve the results of the diagnostic tests. In this research study, different deep-learning architectures will be applied for developing an effective system for pancreatic cancer detection. The most investigated model is **InceptionDense**. It is a hybrid model, which takes the power from both InceptionV3 and DenseNet121. It is observed that InceptionV3 has the structure to capture multi-scale features through parallel convolutional filters. In the case of DenseNet121, feature reusability is allowed; hence it encourages better learning. The other hybrid models that have further been explored in this work are EfficientDense, EfficientV3, EfficientVGG, and ResNetV2. Their efficiency for the diagnosis of pancreatic cancer has been considered. These are to be compared, and hence, this study is looking for better approaches that would further enhance early detection and classification of this aggressive disease.

III.LITERATURE SURVEY

Recent breakthroughs in artificial intelligence and medical imaging have improved detection, classification, and treatment of pancreatic cancer. The scope of literature review brings out a range of innovative approaches different from one another but helping to pool into overall progress in the critical field.

One of the most outstanding models in this stream has been deep learning. Lately, one of the most interesting studies combined DenseNet-based feature extraction with CNN-BiLSTM, fine-tuned with the Sparrow Search Algorithm (SSA) and Harris Hawks Optimization (HHO). The outcome is meaningful; it can analyze whether a patient has pancreatic cancer or not from the set of medical images with an accuracy of as high as 99.26%, thus setting up a novel benchmark for such a diagnostic precision [1].

On the success story, another research team utilized Variational Autoencoders in combination with Elastic Net, Decision Trees, and RBF-SVM to enhance the patient survival prediction. A tailored approach dramatically increased the prediction precision of the local hospital setup, with potential for a revolution in treatment planning [2].

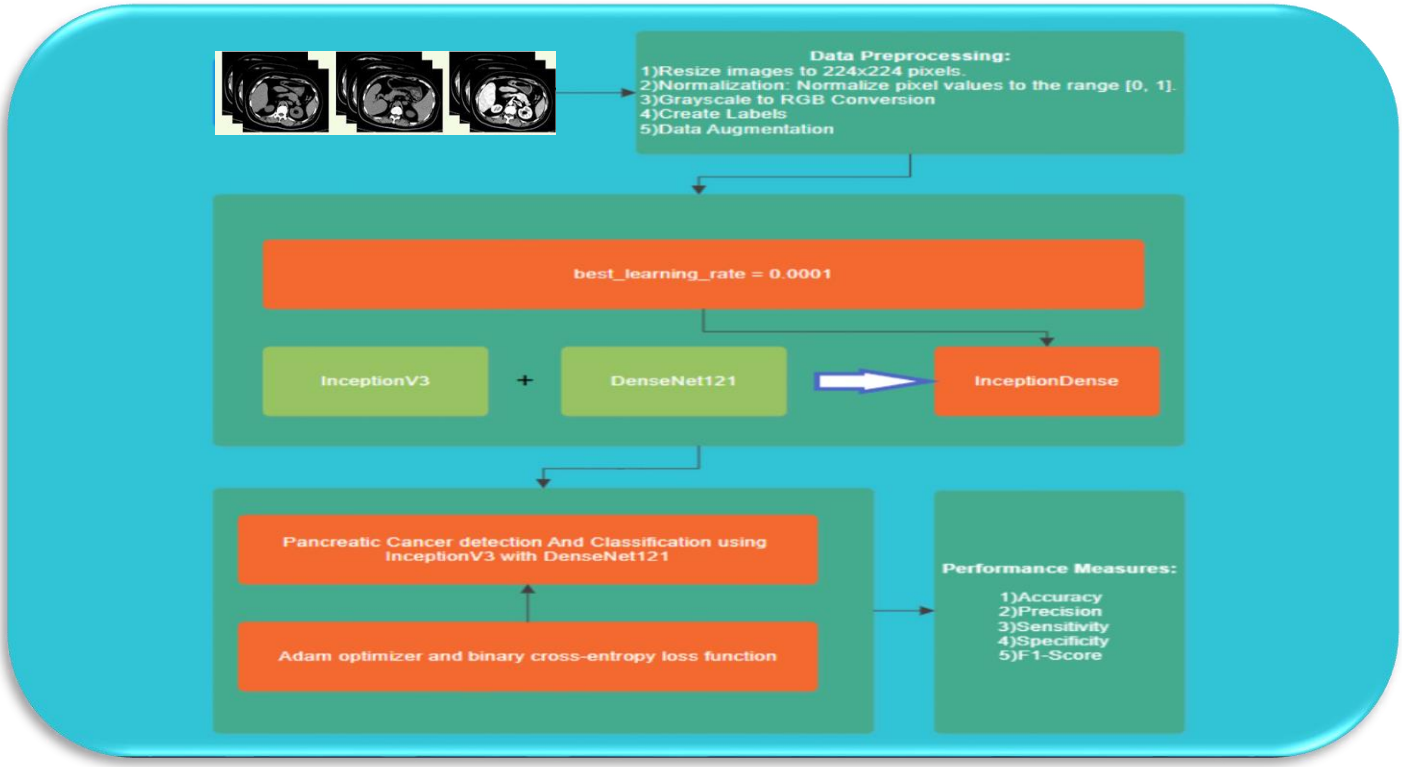


Figure 1.Workflow diagram

Still, one of the biggest challenges in the management of pancreatic cancer is its early detection. To this end, researchers came up with an exciting approach using ultrasensitive nanobiosensors to detect specific biomarkers in liquid biopsies. In a hierarchical decision framework using multi-class classification methods, this approach was able to achieve an astonishing 92% accuracy in early cancer detection, thereby potentially revolutionizing pancreatic cancer diagnosis [3].

The area of medical imaging has picked up the pace because automation in the analysis of CT scans is increasingly being relevant. A new model suggested by researchers using EL-SVM, Softmax, VGG16, and DenseNet121 also shows great promise toward the accurate classification of the stages of pancreatic cancer. This will enhance the precision of diagnosis but also yield better performance with advanced techniques in data augmentation and feature extraction techniques [4].

This complements a decade long review of deep learning techniques, where it cites that improvements in the accuracy and precision of medical imaging are heavily determined by the Convolutional Neural Networks (CNNs) used [5].

As such, the genetic factors associated with the development of pancreatic cancer are considered a significant element. In this field, NIPMI has made noteworthy advancements. Applying a network-based approach, it increased the sensitivity to identify typical genes associated with pancreatic cancer. This might open new avenues for targeted therapy [6].

The variants of U-Net have recently gained prominence in the field of medical image segmentation for performing pancreatic CT segmentation. Those improvements have led to remarkably better performance in segmentation, overcoming longstanding challenges in medical imaging [7].

In the application to medical imaging segmentation, U-Net and variants become one of the top-notch methods for pancreatic CT segmentation. Such innovations have improved the performance of segmentation well and have mitigated long-standing problems in medical imaging [7].

Other innovations include 3D CNN and DenseNet approaches, which have made remarkable progress in AI-driven techniques toward segmentations of images. This shows the imperative power of AI in the use of medical imaging [9, 10].

Treatment Perspective: An exploratory research conducted on hypofractionated ablative radiotherapy for locally advanced pancreatic cancer brings hope as a new therapeutic approach. New technique of this emerging technique enhances the ability of radiotherapy to control tumors within patients, thus offering a new principle that improves outcomes in advanced disease [8].

IV. MATERIALS AND METHODS

4.1 DATASET:

This dataset for analysis comprises 1,411 annotated medical images concerning the cancerous and non-cancerous classes. There was collection of images both from medical imaging repositories as well as in collaboration with healthcare institutions. This kind of dataset would comprise rich details of patient demographics, imaging modalities, and disease stages to represent pancreatic cases comprehensively. Each image here is annotated by expert radiologists or tissues to ensure reliable ground truth in the training of the model toward its evaluation.

Figure 2 Pancreatic-Tumor Image

Figure 3 Normal Image

4.2 PREPROCESSING:

A number of steps of preprocessing is involved to ensure coherent input for the deep learning algorithms and to maximize the efficiency of training. The preprocessed datasets in different deep learning algorithms come through the following ways:

1. Image Resizing: Images will be resized into a fixed size of 224x224 pixels by bilinear interpolation to standardize input dimensions.

2. Normalization: Normalize the pixel values within the interval [0,1] using division over each pixel value with 255, while all input features have similar scales on the time of training.

3. Label Encoding: The annotation is in binary form, where 0 identifies non-cancerous tissues and 1 identifies cancerous tissues. This form very much enhances the model's prediction.

4.3 MODELS:

1. InceptionDense: This hybrid architecture will introduce features from InceptionV3 combined with DenseNet121, and it will be able to improve the feature extraction in order to increase the correctness in the classifications.

Output = DenseNet(InceptionV3(Input))

2. EfficientDense: this is a hybrid version; it will join EfficientNetB0 and DenseNet121: such a kind of architecture can help make the model so that it's able to better learn detailed patterns in medical images.

Output= DenseNet(EfficientNetB0(Input))

3. EfficientV3: it's actually integrating EfficientNetB0 and InceptionV3, which have parallel convolutional filters, including them in capturing multi-scale features.

Output = InceptionV3 (EfficientNetB0 (Input))

4. EfficientVGG: This is the model which combines the EfficientNetB0 with VGG16. It utilizes the efficiency of the former as well as the deep features of the latter to boost the classification performance.

Output = VGG16(EfficientNetB0(Input)

5. ResNetV2: It is the model that combines ResNet50 and MobileNetV2. This uses residual connections to train an even deeper network efficiently.

Output = MobileNetV2(ResNet50(Input))

6. VGG16V2 : It is a model resulting from fusing VGG16 with MobileNetV2, this focuses on achieving even better performance for classifying and also gaining computational efficiency.

Output = MobileNetV2(VGG16(Input))

4.4 PROPOSED MODEL:

InceptionDense is a combination of InceptionV3 and DenseNet121 networks for the detection of pancreatic cancer. It utilizes pre-trained variants of both these networks, excluding top layers as a feature extractor. Images, 224x224x3 in size, are processed through both networks in parallel; Global Average Pooling is applied to the outputs; features are concatenated, and then passed through two dense layers for final classification.

The model was trained on 80% of the total dataset-1,128 images with a validation split of 20%. It used the Adam optimizer along with a tuned learning rate, binary cross-entropy as the loss function and accuracy as the metric. The training process ran for 10 ages.

assessing the models using a variety of criteria , includi ng:

- **Accuracy**: The rate of rightly classified cases to the total number of instant.

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

- **Precision**:
The rate of correct positive prognostications to the total number of positive prognostications.

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

- **Recall**:
The rate of true cons toall factual positive cases.

$$\textbf{Recall} = \frac{TP}{TP+FP}$$

- **F1 Score**: The harmonic mean between precision and recall.

$$\textbf{F1 Score} = 2.\frac{Precision.\ Recall}{Precision + Recall}$$

- **Specificity**: The proportion of true negatives relative to the total number of actual negatives.

$$\textbf{Specificity} = \frac{TN}{TN+FP}$$

On test set, which was 20% of all images-that is, 283 images-the model delivered excellent results. In the overall accuracy, the performance was 99.75%, precision 99.47%, recall 100%, F1 score 99.73%, and specificity 99.52%. These statistics imply that in terms of proficiently identifying pancreatic cancer instances based on images, the model performed better, as its high precision means few false positive results and perfect recall, indicating no positive case was missed.

V.COMPARATIVE ANALYSIS

5.1 ACCURACY TABLE:

Model	Accuracy (%)
InceptionDense	99.75
EfficientDense	100.00
EfficientV3	100.00
EfficientVGG	99.75
ResNetV2	65.40
VGG16V2	72.98

Figure 4 Accuracy Table

The table compares the different models and their corresponding accuracies in the pancreatic cancer prediction. All **InceptionDense** and EfficientVGG models are able to achieve high accuracies of 99.75%. The EfficientDense and EfficientV3 models were able to achieve 100% accuracy, which may or may not be due to overfitting and usually does not happen in real data. Hence, ResNetV2 and VGG16V2 models manifested a significantly lower accuracy of 65.40% and 72.98%, respectively, which would indicate their necessity for more tuning so that they achieve better performance on this dataset.

5.2 TRAINING & TESTING ACCURACY GRAPH:

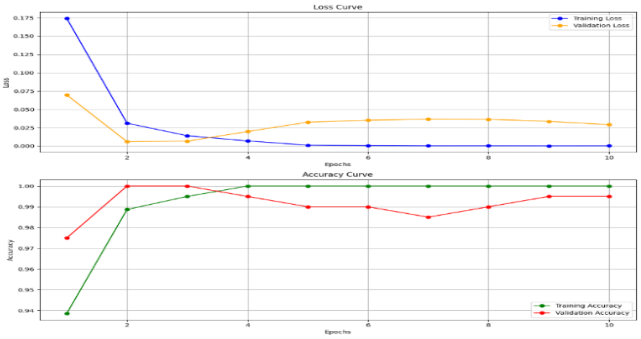


Figure 5 Training & Testing Accuracy Graph

Figure 5: Loss and accuracy curves over 10 epochs for training and validation When trained, the training loss falls steeply and then levels off while the validation loss oscillates. The two accuracies rise rapidly: training approaches 100% and validation is somewhat less, which means that the learning was effective but might be slightly overfitting so a bit of further analysis or some level of regularization would be needed to help improve generalization.

5.3 PRECISION, RECALL & F1 SCORE TABLE:

Model	Precision (%)	Recall (%)	F1 Score (%)
InceptionDense	99.47	100.00	99.73
EfficientDense	100.00	100.00	100.00
EfficientV3	100.00	100.00	100.00
EfficientVGG	99.47	100.00	99.73
ResNetV2	57.72	100.00	73.19
VGG16V2	100.00	42.78	59.93

Figure 6 Precision, Recall & F1 Score Table

The analysis charts show that EfficientDense and EfficientV3 have held precision, recall, and F1 scores perfectly where the model might tend to overfit. **InceptionDense** and EfficientVGG followed this train as well with a high precision of 99.47% and perfect recall of 100% again showing the same trend. There is a great contrast in the case of ResNetV2 regarding its precision (57.72%) and recall (100%), thus also quite low, which would mean that ResNetV2 has a more balanced but lower performance in general. VGG16V2, at 100% precision, lagged behind in recall at 42.78%, therefore scoring a poorer F1 score of 59.93%.

5.4 MODEL PARAMETERS TABLE:

Model	Total Images	Train Parameters	Test Parameters
InceptionDense	1,411	1,128 (80%)	283 (20%)
EfficientDense	1,411	1,128 (80%)	283 (20%)
EfficientV3	1,411	1,128 (80%)	283 (20%)
EfficientVGG	1,411	1,128 (80%)	283 (20%)
ResNetV2	1,411	1,128 (80%)	283 (20%)
VGG16V2	1,411	1,058 (75%)	353 (25%)

Figure 7 Training & Testing Parameter Table

Figure 7: Total images and training/testing split for each model.**InceptionDense**, EfficientDense, EfficientV3, EfficientVGG, and ResNetV2 used 1,411 images with a split of 80% train at 1,128 images and test at 283 images. VGG16V2, however, split the images differently using 75% to train at 1,058 images and 25% to test at 353 images.

5.5 CONFUSION MATRIX OF ALL MODELS:

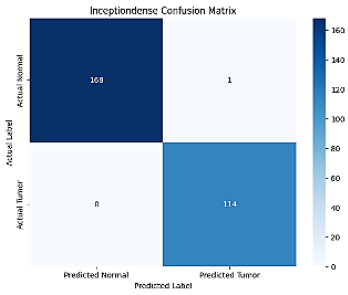


Figure 8 InceptionDense

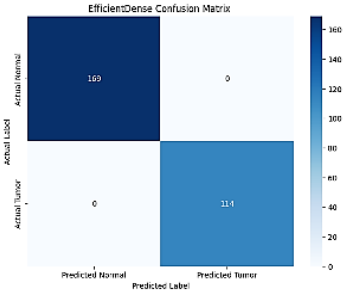


Figure 9 EfficientDense

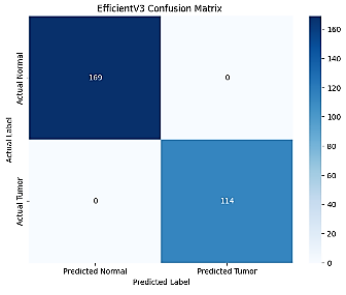


Figure 10 EfficientV3

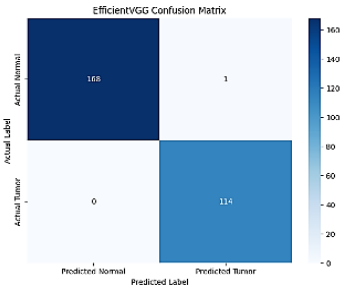


Figure 11 EfficientVGG

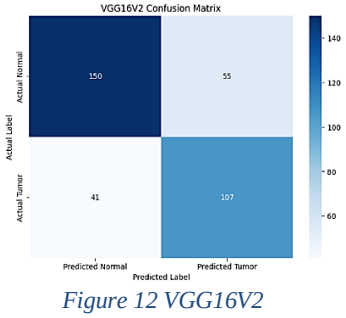


Figure 12 VGG16V2

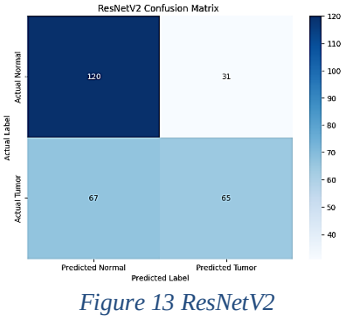


Figure 13 ResNetV2

VI.RESULTS

The best accuracy was obtained by the **InceptionDense** model at 99.75%, suggesting that this model is indeed great at describing some complex features in the cancerous tissue. The accuracy of SSA with Stacked Deep Learning was also impressive, standing at 99.26%. Both models were very competitive in medical imaging, including the detection of pancreatic cancer. Its advanced architecture, which brings out the strengths of both InceptionV3 and DenseNet121, also makes the **InceptionDense** capable of better feature extraction and higher accuracy classification.

These models look very promising for pancreatic cancer detection. The techniques can be further extended to other imaging modalities, such as MRI and ultrasound, and can also be made to work in real-time diagnostics. The techniques must then be validated by larger, more diversified datasets to improve robustness and applicability to the clinic. By combining AI with liquid biopsy methods and the optimization of treatment protocols, it may be possible to improve the patient outcome considerably. Indeed, the advances in medical imaging combined with such AI techniques may one day be able to make early detection and thus better survival from pancreatic cancer possible.

VII.REFERENCES

[1] J. V. N. Ramesh *et al.*, "Sparrow Search Algorithm With Stacked Deep Learning Based Medical Image Analysis for Pancreatic Cancer Detection and Classification," in *IEEE Access*, vol. 11, pp. 111927-111935, 2023, doi: 10.1109/ACCESS.2023.3322376.

[2] Y. Wang, C. Li and Z. Wang, "Advancing Precision Medicine: VAE Enhanced Predictions of Pancreatic Cancer Patient Survival in Local Hospital," in *IEEE Access*, vol. 12, pp. 3428-3436, 2024, doi: 10.1109/ACCESS.2023.3348810.

[3] D. Agarwal, O. Covarrubias-Zambrano, S. H. Bossmann and B. Natarajan, "Early Detection of Pancreatic Cancers Using Liquid Biopsies and Hierarchical Decision Structure," in *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 10, pp. 1-8, 2022, Art no. 4300208, doi: 10.1109/JTEHM.2022.3186836.

[4] M. Li et al., "Computer-Aided Diagnosis and Staging of Pancreatic Cancer Based on CT Images," in IEEE Access, vol. 8, pp. 141705-141718, 2020, doi: 10.1109/ACCESS.2020.3012967.

[5] H. Ghorpade *et al.*, "Automatic Segmentation of Pancreas and Pancreatic Tumor: A Review of a Decade of Research," in *IEEE Access*, vol. 11, pp. 108727-108745,

2023, doi: 10.1109/ACCESS.2023.3320570.

[6] Q. Ding *et al.*, "NIPMI: A Network Method Based on Interaction Part Mutual Information to Detect Characteristic Genes From Integrated Data on Multi-Cancers," in *IEEE Access*, vol. 7, pp. 135845-135854, 2019, doi: 10.1109/ACCESS.2019.2941520.

[7] C. Zhang, A. Achuthan and G. M. S. Himel, "State-of-the-Art and Challenges in Pancreatic CT Segmentation: A Systematic Review of U-Net and Its Variants," in *IEEE Access*, vol. 12, pp. 78726-78742, 2024, doi: 10.1109/ACCESS.2024.3392595.

[8] C. H. Crane, "Hypofractionated ablative radiotherapy for locally advanced pancreatic cancer," in *Journal of Radiation Research*, vol. 57, no. S1, pp. i53-i57, Aug. 2016, doi: 10.1093/jrr/rrw016.

[9] E. Kurnaz and R. Ceylan, "Pancreas Segmentation in Abdominal CT Images with U-Net Model," *2020 28th Signal Processing and Communications Applications Conference (SIU)*, Gaziantep, Turkey, 2020, pp. 1-4, doi: 10.1109/SIU49456.2020.9302180.

[10] C. H. Crane, "Hypofractionated ablative radiotherapy for locally advanced pancreatic cancer," in *Journal of Radiation Research*, vol. 57, no. S1, pp. i53-i57, Aug. 2016, doi: 10.1093/jrr/rrw016.

ORIGINALITY REPORT

3%	1%	2%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

3rd Congress on Smart Computing Technologies

(CSCT 2024)

Organized by

National Institute of Technology, Sikkim, India

Certificate of Presentation

This certificate is proudly awarded to

Gandikota Narendra

for presenting the paper titled

Advanced Techniques in Deep Learning for Pancreatic Cancer Detection and Classification

authored by

Kandi Narasimhareddy, Gandikota Narendra, Katta Subbarao, Nallabothu Narendra, Dodda Venkata Reddy, Sireesha Moturi

in the 3rd Congress on Smart Computing Technologies (CSCT 2024)

held during

December 14-15, 2024.

Prof. Mukesh Saraswat
General Chair

Dr. Abhishek Rajan
General Chair

Springer

https://www.scrs.in/conference/csct2024

SCRS/CSCT2024/PC/345