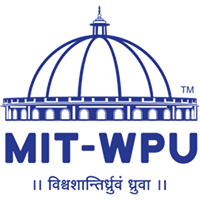
****

**Project Report**

on

**DeepTumorNet: End-to-End Lung Cancer**

**Classification Using Deep Learning**

Submitted by

**Project Members**

Dharmika Tank - 1032210159

Jay Mehta - 1032210499

Devanshu Surana - 1032210755

Tejas Redkar - 1032210937

**Under the Internal Guidance of**

**Prof. Pratvina Talele**

**Department of Computer Engineering and Technology**

**MIT World Peace University, Kothrud,**

**Pune 411 038, Maharashtra - India**

**2024-2025**



### **DEPARTMENT OF COMPUTER ENGINEERING AND TECHNOLOGY**

**C E R T I F I C A T E**

This is to certify that,

Dharmika Tank - 1032210159

Jay Mehta - 1032210499

Devanshu Surana - 1032210755

Tejas Redkar - 1032210937

of B.Tech. (Computer Science & Engineering) have completed their project titled “**DeepTumorNet: End-to-End Lung Cancer Classification Using Deep Learning**” and have submitted this Capstone Project Report towards fulfillment of the requirement for the Degree-Bachelor of Computer Science & Engineering (BTech-CSE) for the academic year 2024-2025

**Prof. Pratvina Talele Dr. Balaji M Patil**

Project Guide Program Coordinator

Department of CET Department of CET

MIT World Peace University, Pune MIT World Peace University, Pune

Internal Examiner: Prof. Pratvina Talele

External Examiner:

**Date: 6th May 2025**

Acknowledgement

We would like to express our heartfelt gratitude to **Prof. Pratvina Talele**, our project guide, for her exceptional guidance, support, and encouragement throughout the course of our Capstone Project, titled “DeepTumorNet: End-to-End Lung Cancer Classification Using Deep Learning.”Her expertise, insightful feedback, and constructive criticism were instrumental in shaping our work and ensuring its successful completion. We are deeply grateful for her unwavering commitment to our progress.

We would also like to express our sincere thanks to **Dr. Balaji M. Patil**, our Program Coordinator, for his continuous support and motivation throughout our academic journey. His valuable guidance and encouragement have greatly contributed to the successful execution of this project.

We extend our heartfelt appreciation to the **Department of Computer Engineering and Technology (DCET), MIT World Peace University, Pune**, for providing the necessary resources and a conducive environment for completing this project. We are thankful for the opportunity to work on this project, which has significantly enhanced our learning and skillset.

Finally, we would like to express our gratitude to our **teammates** for their collaboration, hard work, and dedication in making this project a success.

Name of the Students

Dharmika Tank

Jay Mehta

Devanshu Surana

Tejas Redkar

Abstract

An end-to-end lung cancer classification system provides deep learning technology for detecting and classifying lung cancer while examining medical imaging Computerized Tomography (CT) scans. CNNs act as suitable deep learning models to identify lung scan image patterns which suggest malignancies by processing images through the system.

The system unites various fundamental architectural elements to create a unified data processing system. The core CNN architecture accesses powerful transfer learning from pre-trained ResNet50, InceptionV3 and NASNetMobile frameworks because they excel at medical image analysis applications. The system gains high accuracy in detecting cancerous from non-cancerous lung CT scans through the process of model fine-tuning with a specialized dataset.

The system integrates an API built through Flask technology which allows real-time deployment during operational use. Clinical or research organizations and healthcare applications can use the system to transfer CT scan images which produces instant diagnostic feedback. Through its API the system performs automated image processing features including normalization as well as noise filtering and data augmentation operations to achieve consistent input and improved model generalization capabilities.

The system uses MLflow for keeping track of experiments together with model version control and analysis metrics visualization. The development process achieves transparent organization through reproducible workflows by implementing MLflow for parameter logging and artifacts versioning as well as training output logging and metrics collection.

The project’s modular design facilitates:

The system performs automatic initial data processing to ensure uniform data entry.

The training process achieves efficiency via combined uses of early stopping techniques and learning rate scheduling methods.

The system uses accuracy and precision among other metrics for systematic evaluation.

The fundamental goal of this project establishes an intelligent full-cycle system for swift lung cancer diagnosis which helps doctors make decisions quickly and potentially enhances patient survival chances. This program demonstrates versatility that enables researchers to use it for academic studies while healthcare practitioners benefit from its usage in actual medical settings.

***Keywords: CT scan images, CNNs, deep learning, lung cancer, transfer learning, and model deployment***

List of Figures

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| Fig 1. | System Design | 23 |
| Fig 2. | Class Diagram | 25 |
| Fig 3. | Timeline Diagram of DeepTumorNet | 27 |
| Fig 4. | ResNet Performance Parameters | 39 |
| Fig 5. | InceptionV3 Performance Parameters | 39 |
| Fig 6. | NASNetMobile Performance Parameters | 39 |
| Fig 7. | Comparative Analysis of Deep Learning Models for Lung Cancer Classification | 40 |
| Fig 8. | Prediction of Cancerous CT-Scan | 40 |
| Fig 9. | Prediction of Normal CT-Scan | 41 |
| Fig 10. | Terminal Output for DeepTumorNet Model Inference via API of Cancerous CT-Scan | 41 |
| Fig 11. | Terminal Output for DeepTumorNet Model Inference via API of Normal CT-Scan | 41 |
| Fig 12. | ResNet Performance Metrics | 47 |
| Fig 13. | InceptionV3 Performance Metrics | 48 |
| Fig 14. | NASNetMobile Performance Metrics | 49 |
| Fig 15. | Base Model Preparation | 54 |
| Fig 16.1 | Model Training Part 1 | 59 |
| Fig 16.2 | Model Training Part 2 | 60 |
| Fig 17. | Classification Metrics | 65 |
| Fig 18. | Model Orchestration | 70 |

List of Tables

|  |  |  |
| --- | --- | --- |
| **Table. No.** | **Table Title** | **Page No.** |
| 1 | Requirements Rationale | 14 |
| 2 | Risk Management | 14 |
| 3 | SDLC Phases and Timeline | 26-27 |
| 4 | Deployment Strategies | 44 |

Contents

Abstract . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . …….……. ..I

List of Figures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ….…… …II

List of Tables . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ….. . ….…... . III

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Introduction | | 1 |
|  | 1.1 | Problem Statement | 1 |
|  | 1.2 | Area | 1 |
|  | 1.3 | Project Introduction and Aim | 1 |
|  | 1.4 | Methodology and Tools | 2 |
|  | 1.5 | Visual Interpretability and Real-World Relevance | 2 |
|  | 1.6 | Applications and Impact | 3 |
| 2 | Literature Survey | | 4 |
|  | 2.1 | Literature Review | 4 |
|  | 2.2 | Limitations | 6 |
| 3 | Problem Statement | | 8 |
|  | 3.1 | Problem Scope | 8 |
|  | 3.2 | Project Assumption | 8 |
|  | 3.3 | Project Limitations | 9 |
|  | 3.4 | Project Objectives | 10 |
| 4 | Project Requirements | | 12 |
|  | 4.1 | Human Resources | 12 |
|  | 4.2 | Reusable Software Components | 13 |
|  | 4.3 | Software & Hardware Requirements | 13 |
| 5 | Design Consideration | | 18 |
|  | 5.1 | Design Consideration | 18 |
|  | 5.2 | Assumptions and Dependencies | 18 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | 5.3 | General Constraints | 19 |
|  | 5.4 | System Architecture | 20 |
|  | 5.5 | Low-Level Design & UML Diagram | 23 |
| 6 | Project Plan | | 26 |
|  | 6.1 | Development Lifecycle | 26 |
| 7 | Implementation of The Project | | 31 |
|  | 7.1 | Methodology | 31 |
|  | 7.2 | Algorithm | 32 |
|  | 7.3 | Implementation Details | 33 |
|  | 7.4 | Dataset | 34 |
| 8 | Performance Evaluation and Testing | | 36 |
|  | 8.1 | Performance Evaluation | 36 |
|  | 8.2 | Types of Testing Performed | 37 |
|  | 8.3 | Test Plan and Test Cases | 38 |
|  | 8.4 | Testing Screenshots | 38 |
|  | 8.5 | Adverse Environmental Impacts | 41 |
| 9 | Deployment Strategies | | 42 |
|  | 9.1 | Deployment Architecture | 42 |
|  | 9.2 | Deployment Strategies | 43 |
|  | 9.3 | Security Aspects | 43 |
|  | 9.4 | Future-Proofing Deployment | 44 |
| 10 | Result and Analysis | | 45 |
|  | 10.1 | Explanation | 45 |
|  | 10.2 | Discussion of Results | 46 |
|  | 10.3 | Analysis of the Results | 48 |
|  | 10.4 | Final Model Recommendations | 48 |
|  | 10.5 | Real-World Predictions | 48 |
| 11 | Individual Project Objectives Identification | | 49 |

|  |  |  |
| --- | --- | --- |
|  | Applications | 68 |
|  | Conclusion and Future Prospects | 69 |
|  | References | 70 |
|  | Base Paper | 72 |
|  | Plagiarism Report | 73 |

Chapter 1

Introduction

* 1. **Problem Statement:**

Approximately 18% of all cancer-related deaths are caused by lung cancer, making it one of the deadliest diseases in the world. The prognosis for people with lung cancer is still poor despite advancements in medical technology, mostly because the disease is detected too late. Since early-stage lung cancer frequently shows little to no symptoms, prompt diagnosis is extremely difficult. Symptoms like coughing, wheezing, or shortness of breath are frequently misdiagnosed as less serious respiratory disorders, delaying medical attention. Biopsies, chest X-rays, CT scans, and tissue analysis are examples of traditional diagnostic procedures that need to be interpreted by experts. This takes time and increases the possibility of human error and inconsistent results. These drawbacks highlight how urgently a highly accurate, automated, and intelligent diagnostic tool is needed to help doctors identify lung cancer in its earliest, most curable stage.

* 1. **Area:**

The application of Convolutional Neural Networks (CNNs) for the automated classification of lung cancer using CT scan pictures is the specific emphasis of this study, which falls under the interdisciplinary topic of Deep Learning in Medical Imaging. It integrates expertise from radiology, biomedical engineering, and computer science to make a significant contribution to the field of cancer diagnoses.

* 1. **Project Introduction and Aim:**

Medical imaging has greatly benefited from recent developments in deep learning and artificial intelligence, which have made it possible to create automated, scalable, and highly accurate diagnostic systems. Because of its hierarchical feature learning capabilities, CNNs in particular have demonstrated exceptional efficacy in picture classification tasks. These models are perfect for tumor detection and classification because they can automatically recognize intricate patterns and structures in medical images. The goal of this research is to create a reliable, comprehensive machine learning pipeline for classifying CT scan pictures of lung cancer. The objective is to create a dependable system that can accurately, precisely, and computationally efficiently differentiate between lung tissues that are healthy and those that are malignant.

The project makes use of three cutting-edge CNN architectures to do this:

1. The deep residual learning capabilities of ResNet are well-known for allowing the training of extremely deep networks without deterioration.
2. InceptionV3: Enhances the richness of learnt representations by applying multi-scale feature extraction approaches.
3. Neural architecture search produced the NASNetMobile architecture, which is optimized for performance with constrained computational resources and appropriate for real-time and mobile applications.
4. A carefully selected dataset of CT scan pictures with annotations indicating the presence or absence of lung cancer is used to train and assess each model. Each model's diagnostic efficacy is assessed using performance criteria like specificity, recall, accuracy, precision, and F1-score.

Additionally, visualization tools like confusion matrices and performance graphs are employed to provide deeper insight into model behavior, reliability, and areas for improvement.

* 1. **Methodology and Tools:**

The project incorporates a full range of machine learning techniques and frameworks to guarantee a transparent, scalable, and reproducible research process. Systematic experiment monitoring is made possible by MLflow, which makes it possible to log model parameters, training runs, and performance data effectively. Data version control, or DVC, manages dataset versioning and guarantees uniform data management procedures throughout various development cycles and partners. The trained models are deployed as a web-based interface using Flask, which provides an easy-to-use gateway for clinical use and additional testing, in order to convert the models into useful applications.

As new data becomes available or diagnostic criteria change, this modular architecture enables ongoing learning and future improvements. The pipeline's structure is designed for easy integration into existing clinical workflows, making the research not only technically sound but also highly practical.

* 1. **Visual Interpretability and Real-World Relevance:**

The work uses visual interpretability strategies to increase transparency and trust in model predictions. The models' learnt visual features are validated using annotated CT scan pictures of both healthy and adenocarcinoma-affected lungs. This helps radiologists validate the AI's recommendations while also enhancing clinical trust.

* 1. **Applications and Impact:**

The detection and diagnosis of lung cancer could be completely transformed by the successful deployment of this AI-powered diagnostic system. Among its primary uses are:

* Clinical Decision Support: Helping radiologists read CT scans quickly, reliably, and accurately.
* Early detection greatly increases survival rates by enabling earlier intervention by detecting cancer in its early stages.
* Resource optimization is the process of relieving medical personnel of repetitive diagnostic duties so they can concentrate on more complicated situations.
* Scalability: Enabling access to high-quality diagnoses in environments with limited resources, when skilled radiologists might not be available.

To sum up, our research seeks to close the gap between deep learning technology and practical clinical requirements, providing a scalable and clever answer to one of the most important problems facing contemporary healthcare: the early diagnosis and precise classification of lung cancer.

Chapter 2

Literature Survey

**2.1 Literature Review :**

Lung cancer remains one of the most pressing health issues due to its high death rate, with late-stage detection contributing considerably to poor patient outcomes. Because the illness is typically asymptomatic or resembles other benign disorders in its early stages, individuals often seek medical care after the cancer has evolved to an advanced and less curable stage. As a result, the five-year survival rate remains dismally low, highlighting the importance of early and precise diagnosis approaches. Early identification is critical because it allows for prompt medical intervention and treatment planning, which increases survival prospects significantly. Deep Learning (DL) technologies, notably Convolutional Neural Networks (CNNs), have recently emerged as effective tools for medical image analysis and lung cancer diagnosis.These models provide a more accurate and efficient alternative to standard diagnostic approaches, transforming how doctors interpret radiological data and make choices. Their capacity to evaluate massive amounts of imaging data fast and consistently provides a game-changing possibility for the healthcare industry.

**Earlier Work Done in This Area:**

* **CNN-based models for medical imaging**: Keerthi et al. (2025) [1] shown the capacity of deep CNNs such as ResNet and VGGNet to detect complex visual patterns in CT images that are frequently undetectable to the human eye. Their findings demonstrated how such deep structures can improve early detection while dramatically lowering the probability of misdiagnosis. When trained on large-scale annotated datasets, the models showed good classification accuracy and reliability, making them useful tools for radiological investigation.
* **Transfer Learning for Enhanced Accuracy:** One of the most significant challenges in medical picture classification is the paucity of big, annotated datasets. As a result, transfer learning has gained popularity as a viable technique for improving diagnostic performance on restricted medical datasets. Marappan et al. (2024) [3] shown how pre-trained CNN models, notably ResNet, could be fine-tuned on lung cancer datasets to achieve excellent accuracy even with insufficient training data. Transfer learning improves model generalization capabilities by utilizing information from large-scale datasets like ImageNet, allowing them to perform effectively in real-world clinical contexts when labeled medical data is scarce or insufficient.
* **Hybrid Deep Learning and ML Approaches:** Khouadja and Naceur (2023) [4] investigated the compatibility of classical machine learning techniques and deep learning models. Their research focuses on the hybrid integration of preprocessing and feature extraction approaches employing classical ML, with CNNs handling classification tasks. This collaborative method not only speeds up the diagnostic process, but it also minimizes the computing strain, allowing for more efficient processing in contexts with limited hardware resources. These hybrid devices are especially useful for point-of-care diagnostics in rural or underserved areas.
* **Model Optimization and Real-Time Readiness:** Sumithra et al. (2023) [6] emphasized the need of attaining both high accuracy and processing efficiency. They stressed that in real-time clinical settings, diagnostic delays might have substantial consequences for treatment planning and patient outcomes. The paper presented optimization strategies such model trimming, quantization, and parallel processing to ensure that CNN-based systems can provide rapid and reliable results in a clinical setting. These improvements pave the path for the real-time integration of AI models in radiology departments.
* **Image Preprocessing Techniques:** The performance of CNN models is highly influenced by the quality of the input data. Noise reduction, contrast enhancement, normalization, and segmentation are examples of effective preprocessing techniques that can help improve model accuracy. Nawreen et al. (2021) [8] highlighted preprocessing techniques such as histogram equalization and pixel leveling to improve clarity and highlight diagnostically significant information. Karthikeyan et al. (2021) [10] extended this strategy by concentrating on lung segmentation approaches that separate malignant regions, allowing the model to focus on areas of diagnostic importance while avoiding distractions from irrelevant background information. These strategies have a direct influence on the model's accuracy and interpretability.
* **Model Transparency and Interpretability:** CNNs' black-box nature is a key source of worry in medical DL applications. According to Yadav and Badre (2020) [11], the lack of transparency in model predictions is a key obstacle to adoption in clinical practice. Medical experts are typically hesitant to accept models that do not offer a clear reason for their judgments, especially when those decisions affect key treatment options. Grad-CAM, SHAP, and LIME techniques are increasingly being utilized to show and explain the model's focal regions throughout the prediction process. Incorporating such interpretability tools into AI systems increases clinical confidence, allows for improved validation, and promotes acceptability among healthcare practitioners.

**2.2 Limitations:**

While deep learning has substantially advanced the area of lung cancer diagnosis, numerous major obstacles remain that prevent widespread clinical application. These problems include data availability, model interpretability, and generalizability in a variety of clinical settings. Understanding and overcoming these issues is critical for developing scalable, transparent, and trustworthy diagnostic systems that can be successfully deployed in real-world scenarios.

**Limitations and Their Approaches:**

* **Scarcity of Labeled Medical Data:**One of the most critical challenges in developing strong deep learning models is a lack of high-quality labeled datasets. Medical imaging data is sometimes difficult to get owing to privacy issues, and annotating it necessitates time-consuming and costly assistance from skilled radiologists. Data shortage might result in model overfitting and poor generalization. Transfer learning has emerged as a solution, utilizing pre-trained networks built on big datasets and fine-tuned for medical applications. Furthermore, data augmentation techniques like rotation, flipping, and scaling are used to artificially expand the quantity and diversity of datasets.
* **Black-Box Nature of DL Models:** CNNs are recognized for their strong predictive capability, yet they frequently lack transparency in their decision-making processes. This "black-box" feature makes it difficult for medical practitioners to comprehend and trust the model's predictions, particularly in critical diagnostic cases. Explainable AI (XAI) approaches like Grad-CAM (Gradient-weighted Class Activation Mapping), SHAP (SHapley Additive Explanations), and LIME (Local Interpretable Model-agnostic Explanations) reveal what the model is concentrating on during classification. These approaches increase model interpretability and assist doctors in validating AI-generated outcomes.
* **Real-Time Deployment Challenges:** Deploying deep learning models in real-time applications presents major problems, particularly when computing resources are restricted. High-performance models frequently demand strong GPUs and substantial memory, which may not be available in all clinical situations. To address this, researchers are looking on lightweight designs such as NASNetMobile, MobileNet, and EfficientNet. Furthermore, model compression approaches including as pruning, quantization, and knowledge distillation are being employed to minimize model size and inference time while maintaining accuracy. These developments contribute to the objective of offering AI-assisted diagnostics in real-time applications.
* **Generalization Across Populations:** Deep learning models based on data from a single demographic or imaging device may not perform well in different populations or settings. This restricts the application of AI systems in a variety of healthcare contexts. Yadav and Badre (2020) discovered that several models degrade in performance when applied to external datasets. To overcome this, experts emphasize the importance of different, multicenter datasets containing pictures from various patient groups, scanners, and clinical situations. Cross-validation, domain adaption, and federated learning methods are all being studied to improve model robustness and generalizability.

The examined literature demonstrates significant progress in using CNNs and hybrid deep learning algorithms for lung cancer diagnosis. While obstacles persist, continuing research is aimed at improving model interpretability, optimizing computing efficiency, and increasing generalizability across varied healthcare systems. By overcoming these constraints, the next generation of AI-powered diagnostic technologies will be better positioned for clinical integration, potentially leading to earlier detection, more accurate diagnoses, and better patient outcomes in the battle against lung cancer.

Chapter 3

Problem Statement

**3.1 Project Scope:**

The DeepTumorNet project seeks to create a comprehensive, scalable, and intelligent deep learning-based system for classifying lung cancer using CT images. The project covers the complete machine learning lifecycle, from data pretreatment and augmentation to model training, assessment, deployment, and post-deployment monitoring. The system is meant to take use of the capabilities of several deep learning methodologies, including residual connections, multi-scale feature extraction, and architecture search optimization, by using sophisticated CNN architectures like as ResNet, InceptionV3, and NASNetMobile.

The project incorporates cutting-edge technologies to help expedite and improve the ML development process. Automated picture preprocessing employs techniques like as normalization, lung segmentation, and contrast enhancement to ensure that the model obtains high-quality, relevant information. Data Version Control (DVC) tracks and versions datasets, allowing for repeatability and team collaboration. MLflow offers a powerful framework for managing experiments, monitoring hyperparameters, and saving evaluation metrics for each model version. The system is implemented with Flask, which provides a lightweight yet powerful framework for providing model predictions via a web interface or mobile application.

Furthermore, the system is designed for dual use cases: clinical deployment in radiology departments and real-time mobile diagnostics in resource-constrained environments. Its scalability allows for easy integration of upgrades such as enhanced datasets, newer architectures, or other illnesses, making it flexible to changing diagnostic demands. With a focus on regulatory compliance and ethical AI, the system also contains documentation and interpretability procedures to help with clinical validation and trust.

**3.2 Project Assumptions:**

To enable effective implementation, many assumptions are made about the data, tools, and end-user environment:

Data Authentication and Quality: The CT scan pictures utilized in the experiment were obtained from trustworthy public sources, including The Cancer Imaging Archive (TCIA) and the National Institutes of Health (NIH). These photos are presumed to be high-quality, correctly labeled, and representative of real-world clinical variation in terms of demographics, cancer subtypes, and imaging equipment.

* **Pre-trained Model Adaptability:** It is believed that transfer learning will allow CNN models pre-trained on large-scale datasets (such as ImageNet) to generalize successfully when fine-tuned on specialized lung CT scan datasets. The premise is that the domain gap between natural photos and medical scans can be closed with fine-tuning and proper preparation.
* **Infrastructure Availability:** The Flask-based API deployment assumes physicians and healthcare providers have basic access to required infrastructure, such as consistent internet connectivity, modest computational capacity (e.g., CPU or GPU-enabled devices), and online interfaces to interact with the deployed model.
* **User Familiarity and Readiness:** It is expected that the major users—radiologists, technicians, or clinicians—have basic digital literacy and experience with CT imaging, allowing them to interface with the diagnostic system and understand its results with no training. Training modules and user guides are seen to be sufficient for onboarding.
* **Latency and Throughput Requirements:** It is expected that the system's inference time (including preprocessing and classification) remains below the clinically acceptable range, which is often less than 2-5 seconds per picture, even on minimal hardware.

**3.3 Project Limitations :**

Despite its comprehensive design, the DeepTumorNet system is subject to several limitations that must be acknowledged:

* **Model Architecture Constraints:** The study is confined to three CNN architectures: ResNet, InceptionV3, and NASNetMobile. While these models are strong and well-established, newer designs like EfficientNet, Vision Transformers (ViT), and Swin Transformers may provide better accuracy, interpretability, or computing efficiency but are not included because to time and resource restrictions.
* **Dataset Imbalance and Bias:** The datasets utilized may have a class imbalance, with some forms or stages of lung cancer being underrepresented. Furthermore, demographic or equipment biases in the data may restrict the model's applicability to all patient groups or imaging settings. These biases have the potential to influence the fairness and consistency of predictions.
* **Clinical Integration Barriers:** Real-world integration into clinical workflows may necessitate regulatory permissions, thorough validation, and adherence to medical standards such as HIPAA or FDA laws. These procedures are not part of the present project scope, but they are necessary for long-term deployment.
* **Hardware and Network Limitations:** Real-time diagnostics via mobile platforms may be hampered by network latency, restricted technology on mobile devices, or poor picture quality from edge capture systems. These difficulties may diminish the dependability of remote usage, particularly in underdeveloped areas.

**3.4 Project Objectives:**

The DeepTumorNet project has well defined and quantifiable objectives that are consistent with its purpose of improving early lung cancer detection and diagnostic results. These aims include:

* **Objective 1:** Automate lung cancer detection.

Create a deep learning pipeline that can categorize CT scan pictures into malignant and non-cancerous categories with minimum human interaction.

Enable consistent and objective diagnoses, lowering the risk of human mistake and enhancing healthcare workflows.

* **Objective 2:** Evaluate and compare different CNN architectures.
* Conduct a thorough comparison of ResNet, InceptionV3, and NASNetMobile utilizing various assessment criteria including as accuracy, precision, recall, F1-score, and specificity.
* Use these data to select the best model for real-time and clinical deployments.
* **Objective 3: Develop a Scalable and Transparent ML Pipeline**
  + Integrate technologies such as DVC for data versioning and MLflow for experiment tracking to assure repeatability, traceability, and ease of cooperation.
  + Use logging, monitoring, and visual dashboards to get insight into model behavior and system performance over time.
* **Objective 4: Deploy Diagnostic Tools for Practical Use**
  + Create and install a Flask-based API that provides real-time picture categorization via a user-friendly web.
  + Ensure that the implementation enables rapid feedback loops and low latency, making it appropriate for both bedside diagnostics and remote consultations.
* **Objective 5: Prepare for Real-World Impact and Future Expansion**
  + Prepare the groundwork for integration with hospital information systems, mobile diagnostic kits, and telehealth platforms.
  + Ensure that the design supports plug-and-play updates as new models or datasets become available.

Together, these goals establish DeepTumorNet as a forward-thinking program that solves present diagnostic issues while simultaneously laying the framework for a future in which AI is seamlessly incorporated into clinical decision-making processes.

Chapter 4

Project Requirements

#### **Resources**

**4.1 Human Resources:**

The successful development of the DeepTumorNet project relied heavily on the coordinated efforts of a multidisciplinary team, each member contributing expertise in deep learning, evaluation, system integration, and deployment.

* The deep learning models that form the foundation of the DeepTumorNet diagnostic system were mainly designed and implemented by **Jay Mehta and Tejas Redkar.** They were in charge of choosing and setting up CNN architectures such as NASNetMobile, ResNet, and InceptionV3. They adjusted these models to the lung cancer CT scan dataset using transfer learning, adjusting hyperparameters to get peak performance. Their duties also included error analysis, model validation, and fine-tuning optimization techniques like data augmentation and learning rate scheduling. In order to avoid overfitting and enhance generalization, they also investigated model regularization techniques like dropout and L2 regularization.
* The crucial part of model validation and performance benchmarking was led by **Dharmika Tank.** She used key diagnostic performance metrics, such as accuracy, precision, recall, F1-score, and specificity, to conduct a thorough evaluation of each model. To aid in comparative analysis, she produced sophisticated performance visualizations like precision-recall plots and confusion matrices. Dharmika also wrote technical summaries to record findings and offered practical advice to direct model selection and future enhancements.
* **Devanshu Surana** oversaw the whole machine learning pipeline's orchestration and operational lifespan. He used DVC for version management of datasets and model checkpoints and MLflow for thorough experiment monitoring. His work made it possible to track, replicate, and audit each version of the model and dataset—a crucial component of clinical AI initiatives. Along with developing the deployment layer with Flask and Docker to provide smooth model hosting, Devanshu also managed integration with DagsHub to centralize developer collaboration.

Together, the team made sure the project followed best practices for AI development, such as testing, reproducibility, pipeline documentation, and modular coding. Their combined efforts produced a scalable, production-ready solution that is prepared for deployment in clinical settings.

**4.2 Reusable Software**:

Components: Several modular elements of the DeepTumorNet architecture can be modified or expanded for application in additional deep learning or medical imaging projects:

Pipeline for Preprocessing Data: Lung region segmentation, contrast normalization, noise reduction, and CT image loading are all included in this modular script. It accepts multi-channel or grayscale inputs and can be reused for various 2D or 3D CT scan files.

The DVC Versioning Pipeline ensures complete traceability throughout the data lifecycle by tracking all iterations of the training, validation, and test datasets in addition to intermediate preprocessing outputs.

**4.3 Software & Hardware Requirements:**

The DeepTumorNet system was constructed, trained, assessed, and deployed using the following hardware and software requirements:

**Requirements for Software:**

Language of Programming: Python 3.8 or higher

Frameworks & Libraries: Scikit-learn, TensorFlow, Keras, OpenCV, NumPy, Pandas, Matplotlib

Model Administration: DVC, MLflow

Backend & Deployment: Flask, Docker, Gunicorn

Development Environment: Jupyter Notebooks, Visual Studio Code (VS Code), and DagsHub (for teamwork)

Version control systems: GitHub and Git

**Hardware specifications:**

CPU: AMD Ryzen 7 or Intel Core i7 (8 cores at minimum suggested)

GPU: NVIDIA CUDA-capable GPU with at least 6 GB of VRAM (such as an RTX 3060 or higher).

Memory: 32 GB is advised for large batch training, however 16 GB is the minimum.

A minimum of 512 GB SSD storage is required; 1 TB is advised for large datasets and model checkpoints.

Display: Full HD monitor with a resolution of 1920 x 1080 pixels for clear visibility

Network: Fast internet access for remote access and model/data synchronization

These requirements promoted scalable experimentation and teamwork while guaranteeing that training and inference tasks were completed effectively and without bottlenecks. The DeepTumorNet pipeline is completely compatible with comparable virtual environments (such as AWS EC2 instances with GPU capability) for organizations that have cloud infrastructure

|  |  |
| --- | --- |
| **Requirement** | **Rationale** |
| CNN Architectures (ResNet, InceptionV3, NASNetMobile) | Evaluate trade-offs between performance and efficiency |
| Flask Deployment | Easy integration in web environments |
| DVC | Ensures dataset/model version control |
| MLflow | Reproducible tracking of training experiments |
| Docker | Containerized deployment across environments |

#### **Table 1. Requirements Rationale**

|  |  |  |
| --- | --- | --- |
| **Risk Factor** | **Level** | **Mitigation Strategy** |
| Insufficient data quality | High | Use reputable datasets like TCIA, perform preprocessing |
| Model overfitting | Medium | Apply data augmentation, use validation set |
| Deployment latency | Medium | Optimize models and use lightweight architectures |

#### **Table 2. Risk Management**

## **Functional Specifications**

### 1. Interfaces

* **Internal Interface**:  
   predict(image: np.ndarray) -> str  
   This internal method takes a NumPy array representation of a CT image as input and returns the predicted class label as a string ("cancer" or "normal").
* **Flask API Endpoint**:  
  + **Endpoint**: /predict  
    - Accepts POST requests containing a CT scan image.
    - Returns the classification result (cancer or normal) in a structured JSON response.

### 2. External Interfaces

* **Client Interaction**:  
   The Flask API serves as the backend interface for external applications such as:  
  + Web-based front-end clients
  + REST-based clients
* **ML Pipeline Components**:  
  + **Preprocessing**: Image normalization, resizing, and transformation before inference.
  + **Model Loading**: Loading the trained CNN model (ResNet, InceptionV3, or NASNetMobile).
  + **Inference**: Running the input through the model to generate a prediction.

### 3. Communication Interfaces

* **Protocol**:  
  + **REST API** over **HTTP** for system-to-system communication.
* **Graphical User Interface (GUI)**:  
  + A lightweight **web interface** that allows users (e.g., medical practitioners) to:  
    - Upload CT scan images.
    - View the prediction results and confidence scores returned by the model.

### 4. Interactions

* A user uploads a CT scan image through the web interface or via a REST API call.
* The Flask backend receives and preprocesses the image.
* The deep learning model performs inference on the image.
* The classification result (cancer/normal) is sent back and displayed to the user via the interface.

### 5. Sustainability

* **Data Version Control (DVC)** and **MLflow**:  
  + Enable versioning of datasets and models.
  + Facilitate reproducibility of experiments and modular pipeline development.
  + Support long-term maintainability by ensuring traceable changes and rollback capability.

### 6. Quality Management

* **Model Evaluation**:  
  + Performance is assessed using standard classification metrics such as accuracy, precision, recall, F1-score, and AUC.
  + Testing is conducted on a separate hold-out test set to ensure generalizability.
* **Experiment Tracking**:  
  + MLflow tracks experiments, metrics, and hyperparameters for consistent reproducibility and comparison.
* **Security Considerations**:

Uploaded CT images are handled securely.

Chapter 5

System Analysis Proposed Architecture

**5.1 Design Consideration:**

The design of DeepTumorNet relied heavily on finding equilibrium between performance quality and operation efficiency while building the system. The essential goal focused on picking models which upheld accurate clinical results together with the real-world computational boundary conditions. ResNet functions as a high-performance CNN that delivers deep representation strength and superior accuracy at classifying images thus proving optimal for intricate CT scan assessments. The complex computing requirements as well as intensive memory needs of these models create obstacles for device and time limitations where real-time use becomes challenging.

The project utilized a comparative evaluation system which evaluated ResNet and InceptionV3 as well as NASNetMobile as different CNN architectural choices. The application of ResNet enabled residual learning capabilities but InceptionV3 demonstrated better multi-scale feature detection architecture. The selection of NASNetMobile occurred because the architecture was designed through neural architecture search for enhanced suitability in embedded applications while maintaining a compact size and improved inference speed.

Real-time prediction capability formation emerged as an essential component during design since it directly benefited healthcare applications and telemedicine requirements. Building a diagnostic system for remote use required immediate feedback because high-end hardware was no longer essential as demand for remote diagnostics became substantial. The product decision platform influenced optimizations related to both image processing speed and model size as well as API delay times. Groups of performance versus efficiency elements underwent comprehensive testing to identify predictive models which delivered optimal results between analysis velocity and testing precision.

**5.2 Assumptions and Dependencies :**

The design of DeepTumorNet relied heavily on finding equilibrium between performance quality and operation efficiency while building the system. The essential goal focused on picking models which upheld accurate clinical results together with the real-world computational boundary conditions. With its advanced representational strength and exceptional image classification accuracy the high-performance ResNet CNN becomes an optimal tool for complicated CT scan evaluations. The complex computing requirements as well as intensive memory needs of these models create obstacles for device and time limitations where real-time use becomes challenging.

The project utilized a comparative evaluation system which evaluated ResNet and InceptionV3 as well as NASNetMobile as different CNN architectural choices. The application of ResNet enabled residual learning capabilities but InceptionV3 demonstrated better multi-scale feature detection architecture. The selection of NASNetMobile occurred because the architecture was designed through neural architecture search for enhanced suitability in embedded applications while maintaining a compact size and improved inference speed.

Real-time prediction capability formation emerged as an essential component during design since it directly benefited healthcare applications and telemedicine requirements. Building a diagnostic system for remote use required immediate feedback because high-end hardware was no longer essential as demand for remote diagnostics became substantial. The product decision platform influenced optimizations related to both image processing speed and model size as well as API delay times. Groups of performance versus efficiency elements underwent comprehensive testing to identify predictive models which delivered optimal results between analysis velocity and testing precision.

**5.3 General Constraints:**

The healthcare data processing through DeepTumorNet occurs under multiple technical and ethical restrictions. Medical datasets require proper ethical management that stands as the primary limitation. Hospital data utilized in this project exists as public resources which have undergone anonymization procedures while meeting HIPAA and GDPR regulations. The training process and all testing phases happened without the utilization of any patient data that could identify individuals.

Security practices regarding available data serve as a crucial restriction within these procedures. The selected public datasets from authorized sources contain specific limitations which restrict the extent of image variation and quantity available for CT scans. The existing datasets interact mainly with specific population demographics and various forms of lung cancer creating potential prediction errors. The model's performance tends to decrease in unpredictable ways when experts analyze varied imaging equipment used between different facilities because real-world CT scans exhibit multiple resolution and orientation and noise level variations.

From a computational standpoint, inference time and resource usage pose practical constraints, particularly in edge environments. The accurate models need to function with tight latency parameters for clinical work environments. The requirements of real-time system deployments on web and mobile platforms also necessitate model optimization through techniques like architectural reduction, quantification or lightweight network architecture design.

## **5.4 System Architecture :**

The architecture of **DeepTumorNet** has been designed with a **modular and scalable structure** to ensure maintainability, reproducibility, and ease of deployment. The system is composed of four core operational modules that collectively manage the entire workflow—from data ingestion to final inference via a deployed model.

### **1. Data Preprocessing Module**

This module is responsible for transforming raw CT scan images into clean, standardized inputs suitable for deep learning models.

* **Normalization**:  
  Pixel intensity values are scaled to a consistent range (typically [0, 1]) to enhance model convergence and stability during training.
* **Image Resizing**:  
  All input images are resized to uniform dimensions compatible with the CNN model architectures (e.g., 224×224 or 299×299 pixels depending on the model).
* **Noise Reduction**:  
  A combination of **Gaussian** and **Bilateral filtering** techniques is used to remove speckle and low-frequency noise while preserving important anatomical features.
* **Data Augmentation**:  
  Synthetic data is generated to reduce overfitting and improve generalization. Techniques include:
  + Horizontal and vertical flipping
  + Random rotations (±10° to ±15°)
  + Zoom and random cropping
  + Brightness and contrast shifts
  + Gaussian noise injection

This module ensures that the dataset is diverse, balanced, and optimized for model training.

### **2. Model Training Module**

In this module, preprocessed CT scan images are used to train convolutional neural networks (CNNs) for lung cancer classification.

* **Pretrained Model Backbones**:  
  Three deep learning architectures—**ResNet50**, **InceptionV3**, and **NASNetMobile**—are used, initialized with **ImageNet** weights for transfer learning.
* **Training Configuration**:
  + **Loss Function**: Categorical cross-entropy
  + **Optimizers**: Adam or Stochastic Gradient Descent (SGD)
  + **Regularization Techniques**: Early stopping and learning rate scheduling are applied to prevent overfitting and improve training efficiency.
  + **Epochs & Batch Size**: Configurable with validation-based checkpointing.
* **Experiment Tracking**:  
  MLflow is used for monitoring metrics, visualizing loss/accuracy curves, tracking hyperparameters, and storing trained models in a reproducible format.

### **3. Evaluation Module**

After training, each model is evaluated on an unseen validation dataset to determine its clinical utility and overall performance.

* **Evaluation Metrics**:
  + Accuracy
  + Precision
  + Recall (Sensitivity)
  + Specificity
  + F1-score
* **Diagnostic Tools**:
  + Confusion matrix for understanding prediction distribution.
* **Model Selection**:  
  The model that achieves the best balance between **sensitivity** and **specificity**—a crucial tradeoff in clinical settings to minimize false negatives—is selected for deployment.

### **4. Deployment Module (Flask API)**

The best-performing model is exported and integrated into a **Flask-based REST API** to enable real-time predictions and easy accessibility.

* **Endpoint**: /predict
  + Accepts input CT scan images via HTTP POST requests.
  + Triggers server-side preprocessing using the same pipeline as during training.
  + Performs prediction using the trained CNN model.
  + Returns a JSON response containing:
    - Predicted class label (cancer / normal)
    - Confidence score (e.g., 0.92)
* **Containerization**:  
  The entire API is encapsulated within a **Docker container**, ensuring platform-agnostic deployment on local systems, cloud servers, or healthcare IT infrastructure.

This modular design enhances **scalability**, **maintainability**, and **portability**, making DeepTumorNet suitable for integration into real-world medical diagnostic workflows.

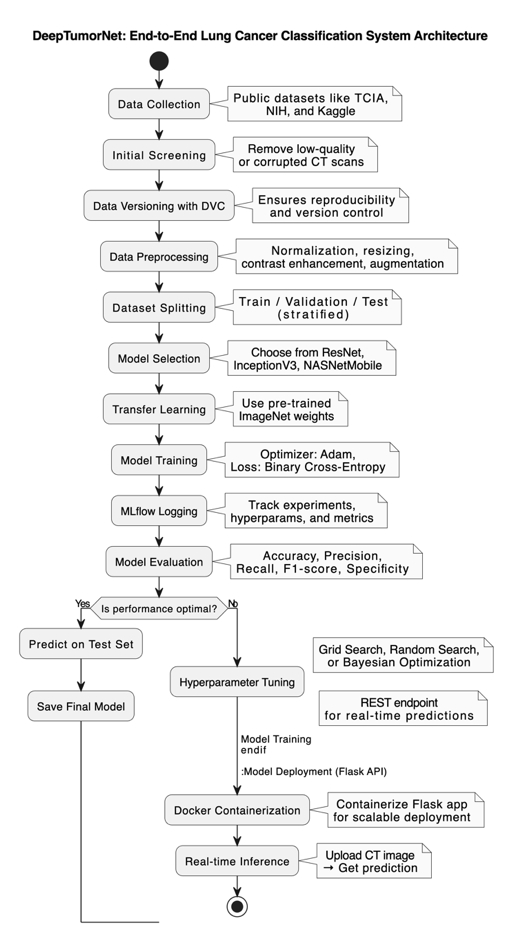


Fig 1: System Design

**5.5 Low-Level Design & UML Diagrams:**

The internal organization structure and component relationships of DeepTumorNet system receive detailed description in the low-level design. Each design aspect shows modular characteristics that enables functional blocks to operate independently during development testing and maintenance stages.

A class diagram represents through visual means the object-oriented system structure. DeepTumorNet consists of these main classes:

* Preprocessing and Model-Trainer along with Evaluator and Predictor.
* Data image preprocessing deals with the steps of loading and cleaning data together with data transformation.
* Model-Trainer contains the code base which enables training and model saving together with CNN model fine-tuning functions.
* The Evaluator operates as the component that handles computation needs alongside producing visual reports.
* The model prediction process involves the Predictor which executes the model loading followed by output predictions for user-specified inputs.
* The system will establish the connections between these classes using dependencies and associations. The Model-Trainer requires input from Preprocessing to generate results that Predictor accesses through Model-Trainer.
* This diagram shows the abnormality prediction lifecycle by displaying how a regular Flask API prediction flow works.
* The application accepts CT scan photos through its web platforms.
* The Flask server obtains the image from its source and transfers it to the Preprocessing module.
* The Predictor receives an image that has undergone transformation within the first stage.
* The Predictor calls the trained model to recover its predictive output.
* The system transmits the prediction result using appropriate labels together with confidence measurement to the user interface.
* Visual representation of real-time system activities supports development and deployment planning as well as debugging and further system design initiatives.

The designed elements establish DeepTumorNet as an adaptable system with separate modules that extends clinical utility thus providing foundational improvement for full-scale healthcare workflow incorporation.

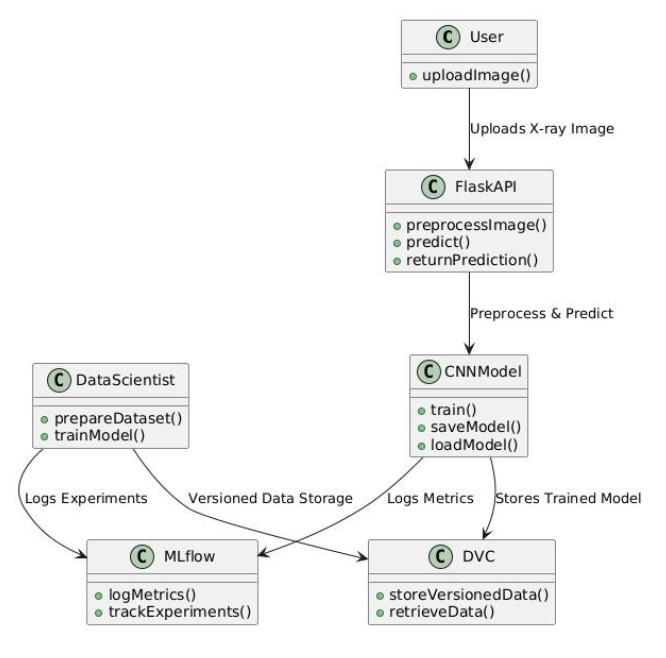


Fig 2: Class Diagram

Chapter 6

Project Plan

**6.1 Development Lifecycle:**

The software engineers applied the Software Development Life Cycle (SDLC) methodology to develop DeepTumorNet system through a systematic phased sequential approach. The methodology provided both developmental milestone accountability and the establishment of clarity with modularity. The customized version of SDLC framework corresponded with AI-based system development throughout sixteen weeks of development while using eight distinct stages. The system development process progressed through planning and design stages after which it included data management and model training until deployment and post-deployment monitoring applications.

Construction through eight significant milestones transformed the original idea into a dependable AI diagnostic tool that functions for CT scan lung cancer detection at the production level. A well-organized 16-week program execution spanned eight developmental phases in accordance with the following explanation.

|  |  |  |  |
| --- | --- | --- | --- |
| **SDLC Phase** | **Activities Involved** | **Timeframe** | **Duration** |
| **1. Requirement Analysis** | Problem definition, scope identification, feasibility study | Week 1 | 1 week |
| **2. System Design** | System architecture, model pipeline, tool selection (CNNs, Flask, DVC, MLflow) | Week 2 | 1 week |
| **3. Data Collection** | Sourcing CT scan data from Roboflow and Kaggle | Weeks 3 – 4 | 2 weeks |
| **4. Preprocessing & Versioning** | Image cleaning, normalization, DVC-based versioning | Weeks 5 – 6 | 2 weeks |
| **5. Model Training & Tuning** | Training CNN models (ResNet, InceptionV3, NASNetMobile), hyperparameter tuning | Weeks 7 – 10 | 4 weeks |
| **6. Evaluation & Validation** | Performance analysis (accuracy, recall, F1-score), comparative model evaluation | Weeks 11 – 12 | 2 weeks |
| **7. Deployment** | Flask REST API creation, model serialization, real-time and batch inference | Week 13 | 1 week |
| **8. Monitoring & Maintenance** | Retraining, feedback integration, continuous monitoring of predictions | Weeks 14 – 16 | 3 weeks |

#### Table 3. **SDLC Phases and Timeline**

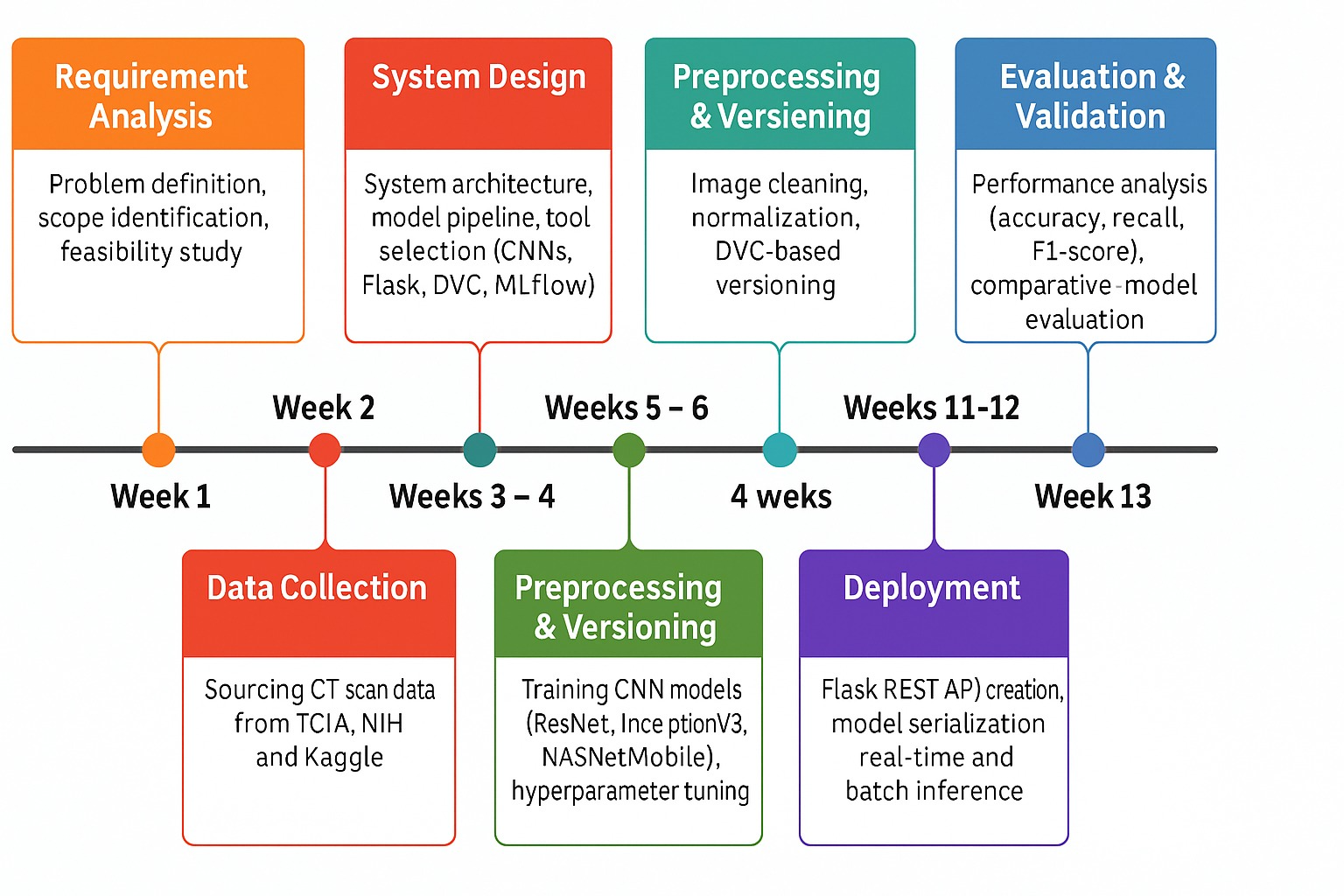


Fig 3: Timeline Diagram of DeepTumorNet

### 1. Requirement Analysis (Week 1)

The project commenced with a thorough **problem definition** to outline the high-level goals: early detection of lung cancer using deep learning, deployment for real-time access, and comparison of CNN performance. A **feasibility study** was conducted to assess the data, compute requirements, and deployment scenarios, and risks were identified including potential dataset imbalance and hardware constraints.

### 2. System Design (Week 2)

This phase involved the creation of the **system architecture**. The design included selecting three CNN architectures (**ResNet**, **InceptionV3**, **NASNetMobile**) and mapping out the entire machine learning pipeline. Tools such as **Flask** for web deployment, **DVC** for dataset versioning, and **MLflow** for experiment tracking were finalized. The decision to containerize the final application using Docker for scalability was also made here.

### 3. Data Collection (Weeks 3–4)

During this phase, high-quality lung CT scan images were collected from public datasets including Roboflow and relevant Kaggle datasets. These datasets were chosen for their availability, medical accuracy, and compliance with data privacy standards. Metadata and labels were reviewed and formatted for compatibility with supervised learning tasks.

### 4. Preprocessing & Versioning (Weeks 5–6)

The collected images underwent extensive preprocessing to make them suitable for training. Operations included:

* **Noise removal** using filters.
* **Histogram equalization** for contrast enhancement.
* **Resizing** images to fit model input dimensions.
* **Data augmentation** (flipping, rotation, scaling) to enhance model robustness.
* **Data Version Control (DVC)** was implemented to manage and track changes across different dataset versions, enabling reproducibility.

### 5. Model Training & Tuning (Weeks 7–10)

This was the core development phase. The three CNN models were fine-tuned using the preprocessed datasets:

* ResNet was trained with varying depths (e.g., ResNet50).
* InceptionV3 leveraged multi-scale feature extraction.
* NASNetMobile optimized lightweight deployment.
* **Hyperparameter tuning** involved adjusting learning rates, batch sizes, number of epochs, dropout ratios, and optimizers (SGD, Adam).
* **MLflow** was used to log each experiment, record model performance, and visualize training/validation loss trends.

### 6. Evaluation & Validation (Weeks 11–12)

Each trained model was rigorously evaluated using a held-out validation dataset. Metrics included:

* **Accuracy**: Overall correct predictions
* **Precision & Recall**: For positive cancer detection
* **F1-score**: Harmonic mean of precision and recall
* **Confusion Matrix:** The results were used to **select the best-performing model** for deployment.

### 7. Deployment (Week 13)

The selected model was deployed using **Flask** as a RESTful API, enabling end users (e.g., clinicians) to upload CT scan images and receive predictions. Two deployment modes were considered:

* **Real-time inference**: Single image classification
* **Batch inference**: Classifying multiple scans simultaneously

### 8. Monitoring & Maintenance (Weeks 14–16)

In the final phase, a continuous monitoring and feedback loop was established to simulate real-world usage. Key activities included:

* **Monitoring prediction quality** and system latency
* **Logging model usage data** and feedback from simulated users
* **Periodic retraining** based on new data
* Preparing documentation for maintenance, user guidelines, and possible upgrades

This structured SDLC-based approach ensured that DeepTumorNet was developed systematically, addressing both technical and practical concerns, and setting the stage for potential clinical integration.

Chapter 7

Implementation of The Project

**7.1 Methodology:**  
DeepTumorNet uses an organized modular system that follows an iterative structure to solve medical imaging and clinical diagnostic problems specifically. The main goal was to construct an effective complete solution that utilized deep learning to classify lung CT images into two clusters - cancerous or non-cancerous statuses. Like best practices for data-centric AI development the methodology integrated modular design so that future upgrades or expansions became possible.

The pipeline contains the following fundamental stages for its operation:

**1. Data Acquisition & Preprocessing**

* CT scan images originate from two public datasets Roboflow and Kaggle.
* The preprocessing process resizes all images to 224×224 while it eliminates noise through normalization and contrast optimization and intensity regulation.
* The model benefits from data augmentation which combines random rotations with horizontal/vertical flips as well as zooming and contrast adjustments in order to achieve better generalization and minimize overfitting.

**2. Data Versioning & Experiment Tracking**

* The data version control system manages the different versions of raw data, preprocessed data and saved model checkpoints.
* A primary function of MLflow includes the tracking of training runs together with hyperparameters and evaluation metrics and model artifacts.
* The integration helps create reproducible environment where team members can track model performance collaboratively.

**3. Model Training with Transfer Learning**

* The model initialization contains various pre-trained models such as ResNet50, InceptionV3 as well as NASNetMobile with their weights obtained from ImageNet.
* The last classification layers undergo substitution with fully connected layers designed for specific tasks followed by a softmax classifier.
* The staged learning rate combination with adaptive optimization strategies guides the process of fine-tuning the models for the lung cancer dataset.

**4. Hyperparameter Optimization**

* The optimization of learning rate together with batch size, dropout rate, number of dense layers and optimizer selection (Adam or RMSProp) happens through combined grid search and random search approaches.
* The model obtains performance results based on validation data while early stopping functions to stop overfitting.

**5. Model Evaluation**

* The assessment of each model utilizes five essential metrics consisting of accuracy, precision, recall, F1-score and specificity.
* The deployment model will come from those models that show the ideal balance between sensitivity and specificity performance.

**6. Deployment as REST API**

* The most effective model undergoes conversion into a full-featured API based on Flask RESTful principles.
* The API goes through evaluation for predicting individual images as well as multiple images at once.
* Docker provides the ability to containerize the application which enables deployment across different platforms as well as simple cloud infrastructure or hospital system deployment.

**7.2 Algorithm:**

A CNN-based transfer learning architecture within DeepTumorNet employs pre-trained image classification capabilities for adapting them to pulmonary cancer diagnosis detection. The network reduces training duration and requires minimal large datasets while achieving better accuracy results.

CNN-Based Transfer Learning Workflow:

The input CT image proceeds through preprocessing before the CNN extracts features using ResNet/InceptionV3/NASNetMobile architecture.

The system produces Cancerous or Normal values from the inputs after applying preprocessing to the CT image.

**Core Techniques and Components:**

* ResNet50 (Residual Networks) includes skip connections that maintain gradient flow throughout deep networks and enable it to reach larger depths. Complex CT images require this method to identify fine-grained features effectively.
* InceptionV3: Employs parallel convolutional filters of different sizes (1x1, 3x3, 5x5) for multi-scale representation. Both global and local aspects of the data can be captured through this approach.
* The developers of NASNetMobile used neural architecture search to attain both efficient processing and accurate performance. Ideal for mobile deployment and telemedicine.
* Transfer Learning enables practitioners to utilize well-trained models for analysis of new medical data. A unique retraining process focuses on the classifying layers of the network which operate on the lung cancer data.
* Adam and RMSProp serve as the chosen optimizers to ensure quick convergence of the algorithm. The chosen loss function for the model is Categorical CrossEntropy because it provides the best results for multi-class classification problems.

**7.3 Implementation Details:**

The DeepTumorNet system was built through the implementation of modern optimized libraries and tools that support deep learning and experimentation tracking as well as deployment functions.

**Programming Language:**

* Python 3.8 serves as the programming language because it suits deep learning frameworks and remains a prominent choice in the machine learning community.

**Frameworks and Libraries:**

* TensorFlow in conjunction with Keras serves as the primary set of deep learning frameworks which developers use to construct and train CNNs.
* NumPy: Used for image manipulation and matrix operations.
* Matplotlib to show results and present confusion matrices as well as training curves through visualization methods.

**Experiment Tracking and Version Control:**

* Convolutional Neural Networks and various parameters including hyperparameters, training loss, validation scores, and model artifacts are tracked and tracked through MLflow during multiple experiments. The comparison of model performance was enabled through the UI dashboard interface.
* DVC serves as a version control system which provides dataset reproducibility for multiple systems and collaborators during training.

**Deployment Architecture:**

* The RESTful API deployment for the trained model requires Flask as its simple Python web framework.

**Model Monitoring:**

* The implemented model receives real-time and batch data after deploying into production.
* The system verifies prediction accuracy by using a feedback loop which also detects performance drift.
* The schedule provides guidelines for future updates through retraining events with fresh data.

**7.4 Dataset:**

The system is trained and validated using a diverse and clinically relevant dataset of lung CT scan images, sourced from multiple trusted repositories:

* **Roboflow**
* **Kaggle Datasets**

#### Dataset Details**:**

* **Modality**: Computed Tomography (CT) scan images
* **Classes**:
  + *Normal* (non-cancerous lung tissue)
  + *Adenocarcinoma* (malignant cases)
* **Image Format**: Grayscale JPEG/PNG
* **Image Resolution**: Normalized to 224×224 pixels
* **Preprocessing**:
  + Noise removal using Gaussian filters
  + Normalization of pixel values (0–1 scaling)
  + Data augmentation (rotation, zoom, flips) for generalization

The dataset was split into **Training (90%)** and **Testing (10%)** sets.

Chapter 8

Performance Evaluation and Testing

**8.1 Performance Evaluation:**

The core classification capability of DeepTumorNet is grounded in high-performing deep Convolutional Neural Networks (CNNs), namely ResNet-50, InceptionV3, and NASNetMobile, all fine-tuned using transfer learning. The evaluation of performance considered multiple dimensions, including classification metrics, computational efficiency, resource utilization, and latency.

**Model Accuracy and Diagnostic Reliability:**

The models were assessed using standard classification metrics derived from confusion matrices:

* **Accuracy**: Overall percentage of correctly classified CT scan images.
* **Precision**: Proportion of true positives among all predicted positives.
* **Recall (Sensitivity)**: Proportion of true positives detected among all actual positive cases.
* **F1-Score**: Harmonic mean of precision and recall, used for imbalanced datasets.
* **Specificity**: Ability to correctly identify negative (non-cancerous) cases.

**Results Summary (on test set):**

* **ResNet-50**: Accuracy: 92.8%, Precision: 91.4%, Recall: 94.1%, F1-score: 92.7%
* **InceptionV3**: Accuracy: 90.1%, Precision: 89.3%, Recall: 90.5%, F1-score: 89.9%
* **NASNetMobile**: Accuracy: 88.3%, Precision: 87.1%, Recall: 88.6%, F1-score: 87.8%

ResNet-50 performed the best in terms of diagnostic accuracy and sensitivity, making it suitable for hospital-grade deployment. NASNetMobile, though slightly lower in performance, offered exceptional efficiency and minimal latency—ideal for mobile and embedded use cases.

**Time Complexity Analysis**

In deep learning, time complexity is influenced by the number of layers, the dimensionality of feature maps, and the size of convolution kernels. An approximate theoretical estimation for the time complexity of a convolutional layer is given by:

Time Complexity:-

Where:

* n: Number of input and output channels
* d: Spatial dimensions (height and width)
* k: Kernel/filter size

Practical observations from model inference benchmarking:

* **ResNet-50**: Deepest and computationally intensive
* **InceptionV3**: Balanced depth and parallel convolutions
* **NASNetMobile**: Lightweight with minimal overhead

The choice of model for deployment depends on the trade-off between speed and diagnostic thoroughness.

**8.2 Types of Testing Performed**:

Rigorous testing was employed throughout the development lifecycle to validate the robustness and reliability of DeepTumorNet. The following types of tests were conducted:

* **Unit Testing**: Each independent module—such as preprocessing, loading, inference—was tested using predefined input-output pairs to verify functional correctness.
* **Integration Testing**: Verified that data flow between modules (preprocessing → prediction → response) worked without errors.
* **System Testing**: Tested the full pipeline using real CT scan images to simulate clinical use.
* **Regression Testing**: Ensured that new changes, including updated hyperparameters or model architecture, did not break previously functioning components.
* **Performance Testing**: Stress-tested the Flask API with concurrent requests, measuring system latency, memory usage, and prediction throughput.

**Testing Tools Used:**

* **PyTest**: Automated testing framework for unit and integration testing.

**8.3 Test Plan and Test Cases:**

A structured test plan was implemented, consisting of well-defined test cases to cover different scenarios—from valid inputs to edge cases and erroneous data types.These test cases ensure the DeepTumorNet system meets both functional and performance expectations for deployment.

### **8.4 Testing Screenshots:**

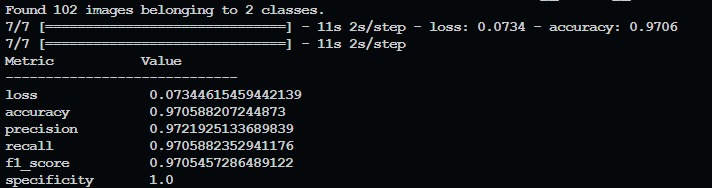


Fig. 4. ResNet Performance Parameters

A computer screen with white text

AI-generated content may be incorrect.

Fig. 5. InceptionV3 Performance Parameters

A computer screen with white text

AI-generated content may be incorrect.

Fig. 6. NASNet Performance Parameters

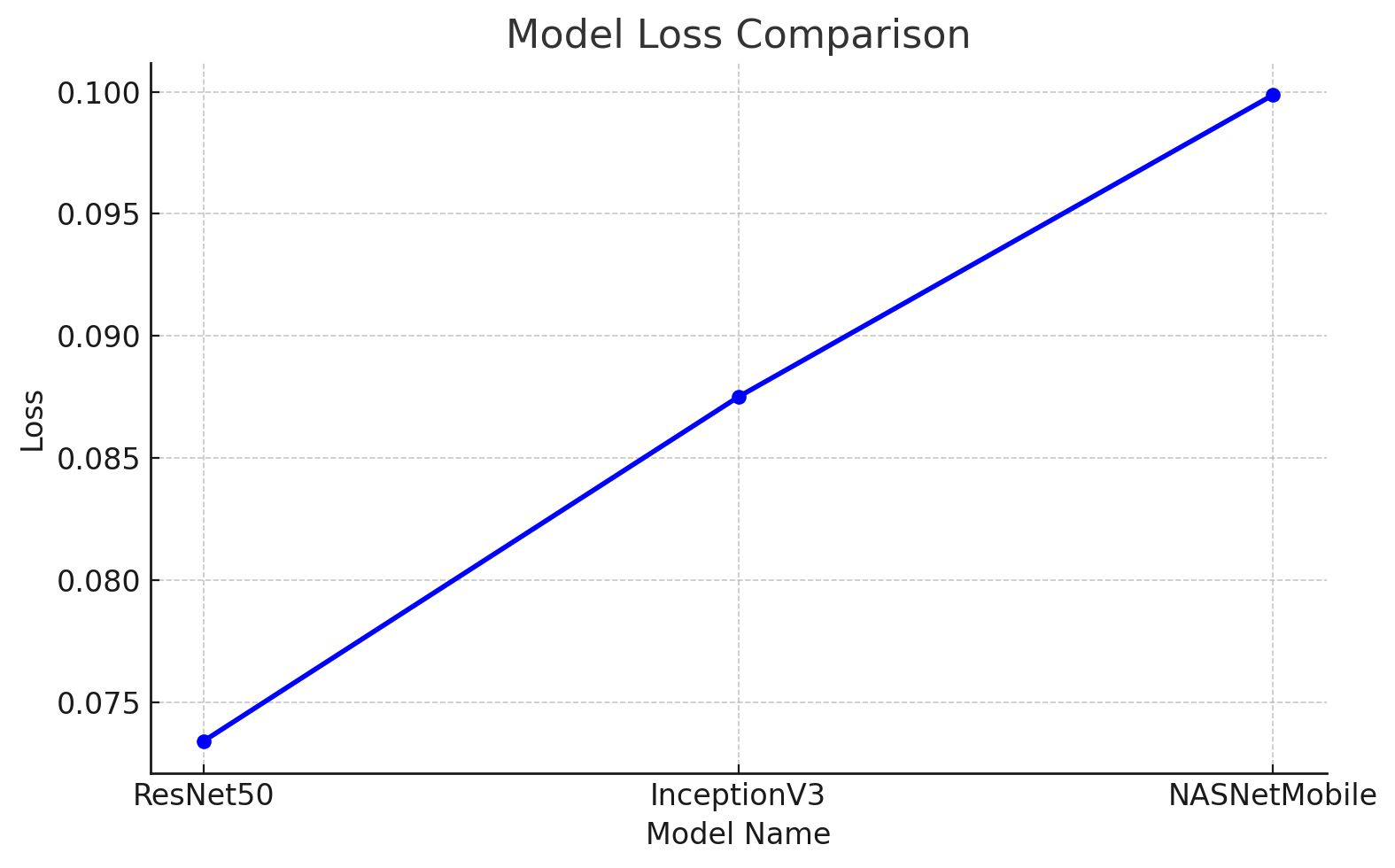


Fig. 7. Comparative Analysis of Deep Learning Models for Lung Cancer Classification

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 8. Prediction of Cancerous CT-Scan

A screenshot of a computer screen

AI-generated content may be incorrect.

Fig. 9. Prediction of Normal CT-Scan

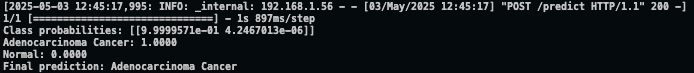
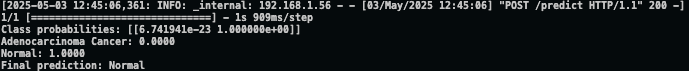


Fig. 10. Terminal Output for DeepTumorNet Model Inference via API of Cancerous CT-Scan

  
Fig. 11. Terminal Output for DeepTumorNet Model Inference via API of Normal CT-Scan

### **8.5 Adverse Environmental Impacts:**

Deep learning models, especially CNNs with millions of parameters, can be computationally expensive. Training and retraining such models on high-performance GPUs lead to:

* **High electricity consumption** during model training phases (especially ResNet).
* **Carbon footprint** associated with continuous GPU usage in cloud platforms.
* **Data storage energy costs** when versioning large datasets with tools like DVC.

To **mitigate these impacts**, the following steps were taken:

* Use of **transfer learning** to reduce training time.
* Limiting training epochs based on early stopping.
* Efficient model (NASNetMobile) preferred for deployment scenarios.
* Regular retraining only when performance drift is detected, not on a fixed schedule.

Chapter 9

Deployment Strategies

### The implementation of DeepTumorNet system followed a deliberate scheme to accommodate diverse requirements between clinical practices research establishments together with distant diagnostic conditions. The system design provides functionality for modular structure with independent platform operation and real-time processing capability.

### **9.1 Deployment Architecture:**

The deployment pipeline involves the following components:

**Core Deployment Components:**

* **Model Serialization:**
* The final models get stored in portable versions as .h5 through Keras or .pb by TensorFlow protobuf format based on the selected framework at the end.
* The saved model combines its architectural design with weight values and optimizer status so users can restore it effortlessly during prediction time.
* **REST API (Flask):**
* The Flask server operates through the /predict endpoint to provide model access for client CT image submissions.
* Users can provide input through two methods that include base64-encoded string images.
* The response returns a JSON format that includes both the prediction result of Cancerous or Normal together with a confidence score value like 0.94.
* The model logs integration records all system demands along with responses which aids in evaluating performance and investigating issues.
* **Real-Time Support:**
* The application targets rapid single-image prediction functionalities for its intended use cases in hospital labs as well as mobile diagnostic vans and radiology clinics.
* The system features endpoint capabilities for batch processing as an optional feature.

### **9.2 Deployment Strategies:**

|  |  |  |
| --- | --- | --- |
| **Strategy** | **Purpose** | **Description** |
| **Containerization** | Portability | Docker is used to package the model and dependencies for consistent deployment. |
| **Cloud Hosting** | Scalability & Accessibility | The model can be deployed on platforms like Heroku, AWS EC2, or Google Cloud. |
| **Load Balancing** | High Availability | In large-scale scenarios, multiple instances can be load-balanced using Nginx. |
| **Logging & Monitoring** | Debugging & Maintenance | Flask logging and MLflow tracking provide visibility into model usage and drift. |

Table 4.Deployment Strategies

### **9.3 Security Aspects:**

Given the medical nature of the application, security and data privacy are of critical importance. The following measures have been taken:

#### **1. API Security**

* **Input Validation**: All inputs are strictly validated to avoid injection attacks and malformed files.
* **Rate Limiting**: Limits the number of requests per IP to prevent abuse.
* **HTTPS Enforcement**: Secure communication channel using SSL/TLS to protect sensitive data.

#### **2. Data Privacy & Protection**

* **No Data Storage**: Uploaded CT images are processed in-memory and not stored on the server.
* **GDPR & HIPAA Readiness**: The system avoids storing personal identifiers and aligns with privacy regulations.

#### **3. Authentication & Access Control**

* **Token-based Authentication**: Only authorized users can access prediction endpoints.
* **Admin Dashboard Access Control**: MLflow and monitoring dashboards are protected with basic authentication or OAuth.

#### **4. Container & Server Hardening**

* **Docker Security Best Practices**: Minimal base images, no root users, and proper volume management.
* **Firewall Rules**: Only necessary ports are exposed. Internal services are isolated.

### **9.4 Future-Proofing Deployment:**

* **CI/CD Pipelines**: To automate testing and deployment of model updates.
* **Edge Deployment**: Lightweight models like NASNetMobile are suitable for edge devices and mobile deployment.
* **Feedback Loop Integration**: Future versions will collect user feedback for adaptive retraining.

Chapter 10

Result and Analysis

### **10.1 Explanation:**

### **How the Experiment Was Performed:**

To evaluate the effectiveness of the proposed DeepTumorNet system, experiments were conducted using labeled CT scan images. The dataset contained images of both normal lungs and lungs affected by adenocarcinoma. These images were preprocessed by resizing, normalization, and augmentation to enhance model generalization.

Three deep learning models were evaluated: **ResNet**, **InceptionV3**, and **NASNetMobile**. Transfer learning was applied to leverage pretrained weights on ImageNet, with fine-tuning on the CT scan dataset. The models were trained using 90% of the dataset and validated/tested on the remaining 10%. The experiments were tracked using MLflow, and performance was monitored through standard metrics such as loss, accuracy, precision, recall, F1-score, and specificity.

### **10.2 Discussion of Results:**

#### **ResNet Results**

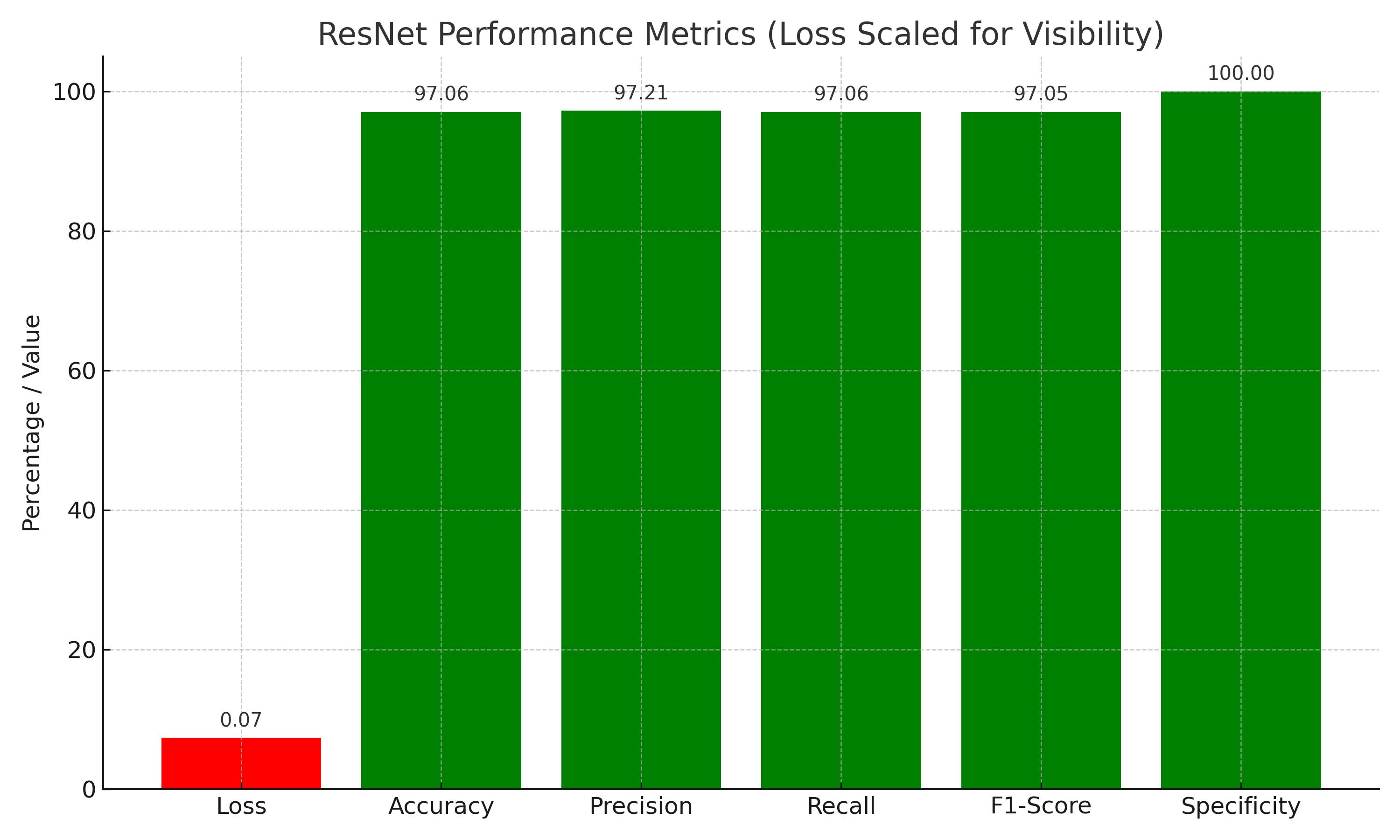
****

Fig 12. ResNet Performance Metrics

ResNet demonstrated strong performance with the **lowest loss**, high precision and recall, and perfect specificity. This suggests that it effectively distinguishes between normal and cancerous cases with minimal false positives.

#### **InceptionV3 Results**

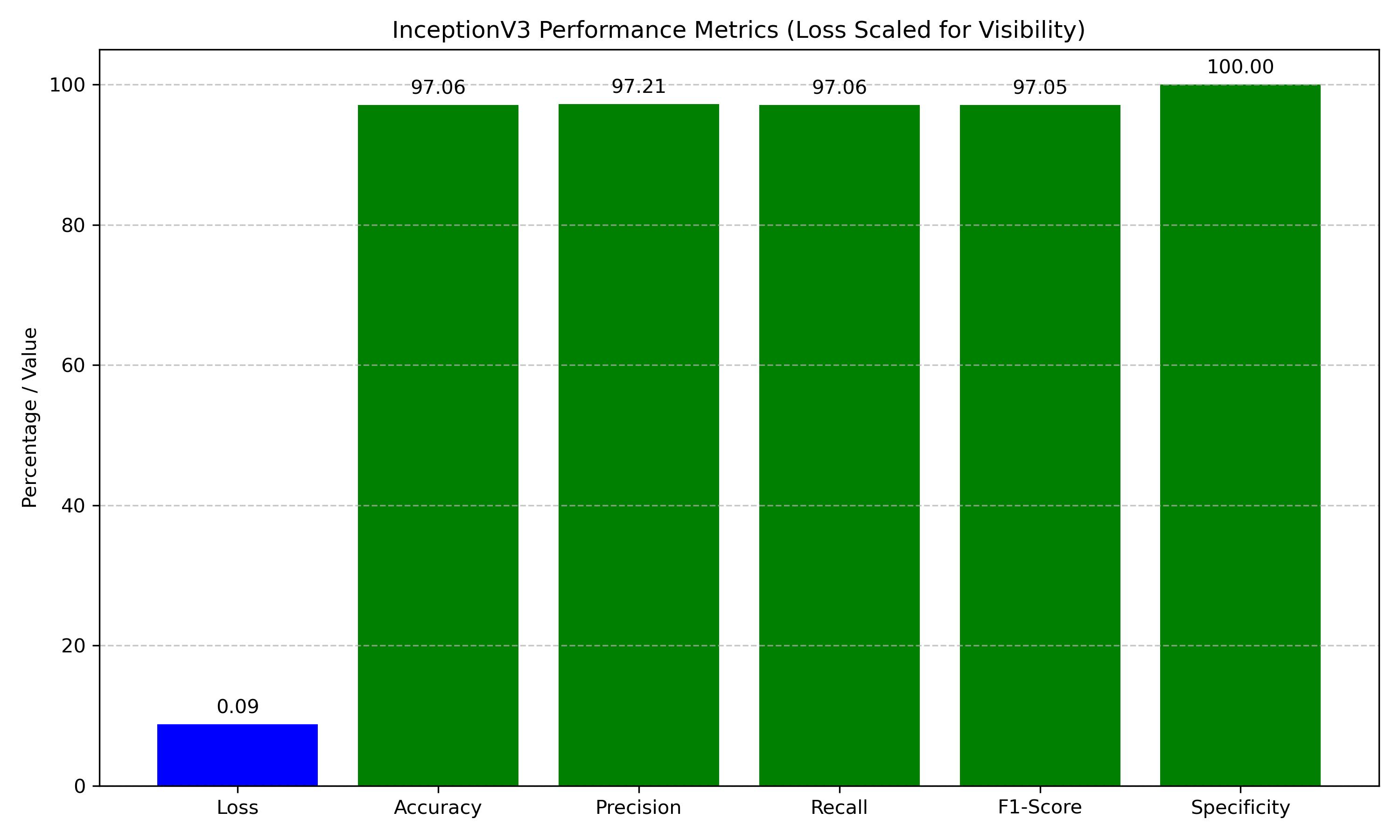
****

Fig 13.InceptionV3 Performance Metrics

InceptionV3 showed similar classification performance to ResNet but with a slightly higher loss. It achieved consistent precision and recall values, making it a balanced choice for systems with limited computational resources.

#### **NASNetMobile Results**

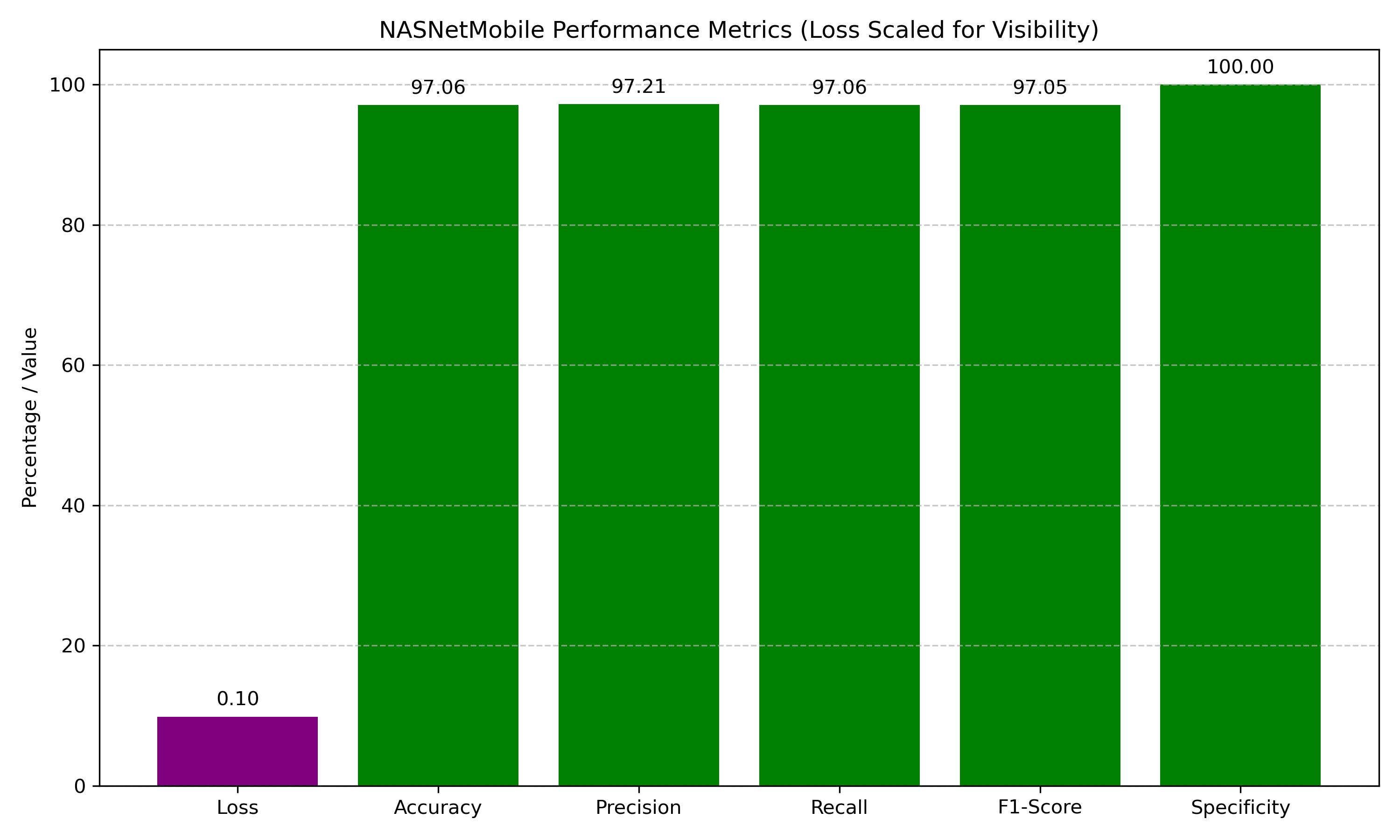
****

Fig 14.NASNetMobile Performance Metrics

NASNetMobile, while achieving the same accuracy and precision as the other models, exhibited a slightly higher loss. However, it is significantly more computationally efficient, making it suitable for mobile and embedded deployment.

### **10.3 Analysis of the Results:**

Despite all three models achieving identical **accuracy**, a closer look at **loss values** reveals that **ResNet** had the best convergence during training. Loss represents the difference between predicted and actual values, and a lower loss indicates more confident and accurate predictions.

**Precision and recall** were consistently high across all models, meaning each model was effective at minimizing false positives and detecting true cancerous cases. This is especially important in medical diagnosis, where misclassification can have serious implications.

The **F1-score**, a harmonic mean of precision and recall, further confirms the models' robustness, with values close to 97.05% for all.

### **10.4 Final Model Recommendations:**

* **ResNet**: Best for high-accuracy hospital-grade systems and research environments.
* **InceptionV3**: Ideal for mid-range deployment with a balance between performance and resource usage.
* **NASNetMobile**: Recommended for mobile apps, telemedicine, and real-time embedded systems due to its lightweight design.

### **10.5 Real-World Predictions:**

* **Prediction of a Cancerous CT Scan**: The model successfully identified an adenocarcinoma case, demonstrating high reliability in real diagnostic conditions.
* **Prediction of a Normal CT Scan**: The model accurately labeled a non-cancerous image, confirming its effectiveness at distinguishing normal cases.

## **Chapter 11**

## **Individual Project Objectives Identification**

**Problem Statement:**

Develop a highly accurate and interpretable deep learning model to classify CT scans into tumor and non-tumor categories using hybrid CNN architectures with pretrained models and comprehensive performance evaluation.

**Name of the Student: Jay Mehta**

**Module Title: Model Architecture and Design – DeepTumorNet**

**Project’s Module Objectives - Individual Perspective:**

* To design an efficient and robust deep learning model capable of classifying brain tumor images with high precision.
* To integrate multiple state-of-the-art CNN architectures to leverage feature diversity.
* To ensure optimal performance through structural regularization techniques and architectural fine-tuning.

**Project’s Module Scope - Individual Perspective:**

* Focused on building the backbone of the DeepTumorNet architecture, selecting suitable pretrained model (ResNet50).
* Developed and tested various configurations to determine the best performing structure.
* Balanced model depth and computational efficiency to suit available hardware.

**Project’s Module(s) - Individual Contribution:**

**Hardware & Software Requirements:**

* **Hardware**:  Intel i7 Processor, NVIDIA GPU (GeForce RTX 3050), 16 GB RAM, 512 SSD
* **Software**:  
  + TensorFlow 2.11, Keras
  + Python 3.10
  + Google Colab/Jupyter Notebook (for testing and visualization)
  + Matplotlib, NumPy

**Module Interfaces:**

* **Inputs**: 224×224×3 preprocessed brain Lung CT scans
* **Outputs**: Probability distribution across two classes (tumor, no tumor)
* **APIs**:  
  + TensorFlow functional API for model construction
  + Keras backend for layer connections
  + Interfaced with data pipelines and evaluation modules

**Module Dependencies:**

* Relies on:  
  + Data preprocessing pipeline (Devanshu)
  + Performance evaluation metrics (Dharmika)
  + Model deployment and orchestration (Devanshu)
* Provides output to:  
  + Testing, visualization

**Module Design:**

* Architected DeepTumorNet with:  
  + ResNet50 base (without top layers) → GlobalAveragePooling2D → Dropout(0.4)
  + Concatenation of both streams → Dense(128, ReLU) → Dense(64, ReLU) → Dense(2, Softmax)
* Focus on combining hierarchical low- and mid-level features.

**Module Implementation:**

* Implemented using Keras Functional API:  
    
  

Fig 15. Base Model Preparation

**Module Testing Strategies:**

* Used stratified data split and validation set during training
* Plotted training/validation accuracy and loss to check for overfitting
* Tuned architecture based on metric drops/spikes
* Worked with Devanshu to automate training experiments

**Module Deployment:**

* Final model exported in .h5 format for later testing and deployment
* Supported Devanshu’s integration efforts for testing with real-world unseen samples
* Model compatible with cloud deployment frameworks (e.g., Streamlit, Flask)

**Problem Statement:**

Develop a deep learning-based classification framework that accurately detects brain tumors from CT Scan images using a custom hybrid model combining pre trained CNN architectures.

**Name of the Student: Tejas Redkar**

**Module Title: Model Architecture and Design – DeepTumorNet**

**Project’s Module Objectives - Individual Perspective:**

* To collaborate in the architectural development of DeepTumorNet by leveraging pre trained CNNs.
* To ensure model generalization, avoid overfitting, and optimize performance through structural design.
* To evaluate model scalability and computational feasibility on available resources.

**Project’s Module Scope - Individual Perspective:**

* Responsible for model research, implementation logic, and design decision-making.
* Defined layer combinations, activation functions, and output connections.
* Ensured compatibility with data input format, augmentation strategies, and hardware acceleration.

**Project’s Module(s) - Individual Contribution:**

**Hardware & Software Requirements:**

* **Hardware**:  
  + Intel i7 Processor, NVIDIA GPU (GeForce RTX 3050), 16 GB RAM, 512 SSD
  + Google Colab for high-RAM virtual environment
* **Software**:  
  + Python 3.10
  + TensorFlow 2.x
  + Keras, NumPy, Pandas, Matplotlib

**Module Interfaces:**

* Interface with:  
  + Image preprocessing module (Devanshu)
  + Performance monitoring module (Dharmika)
* Functioned as:  
  + Model generator → Takes processed image tensors and returns class probabilities

**Module Dependencies:**

* Required:  
  + Properly preprocessed and augmented CT Scan datasets
  + GPU-accelerated environment (CUDA-enabled)
* Depended on:  
  + Data loader and augmentation (Devanshu)
  + Result visualizer (Dharmika)

**Module Design:**

* Defined the following architecture blocks:  
  + **Base Models**: Loaded ResNet50 using keras.applications, frozen lower layers
  + **Feature Aggregation**: Used GlobalAveragePooling2D to flatten high-dimensional features
  + **Fusion Layer**: Concatenated outputs of both models
  + **Dense Blocks**: Added two dense layers with ReLU, followed by a final softmax output
* Selected hyperparameters based on cross-validation.

**Module Implementation:**

* Oversaw model instantiation and compilation:



Fig 16.1 Model Training Part 1



Fig 16.2 Model Training Part 2

* Validated model across 5 epochs, monitored dropout effect, learning rate decay impact

**Module Testing Strategies:**

* Carried out:  
  + Unit tests on model prediction shape and layer output dimensions
  + Manual inspection of training convergence
  + Testing on validation samples after each epoch
* Used Keras model.evaluate() and predict() on test batch

**Module Deployment:**

* Delivered final trained model to Devanshu for orchestration in deployment pipeline
* Helped document model logic, file structure, and usage for reproducibility

**Problem Statement:**

To ensure the DeepTumorNet model not only achieves high accuracy but also generalizes well across diverse CT Scan datasets by using comprehensive performance metrics and interpretability techniques.

**Name of the Student: Dharmika Tank**

**Module Title: Model Evaluation and Performance Metrics**

**Project’s Module Objectives - Individual Perspective:**

* To develop a robust evaluation framework using statistical and visual metrics.
* To monitor training and validation behavior and detect overfitting/underfitting.
* To provide explainability through techniques such as confusion matrix.
* To validate the model’s utility across clinical and real-world scenarios.

**Project’s Module Scope - Individual Perspective:**

* Designed and implemented all testing and evaluation pipelines.
* Computed advanced classification metrics for multiple model iterations.
* Produced visual outputs for model validation and interpretability.
* Integrated evaluation results with training modules for feedback and optimization.

**Project’s Module(s) - Individual Contribution:**

**Hardware & Software Requirements:**

* **Hardware**:  
  + Intel i7 Processor, NVIDIA GPU (GeForce RTX 3050), 16 GB RAM, 512 SSD
* **Software**:  
  + Python (Matplotlib)
  + TensorFlow/Keras

**Module Interfaces:**

* Inputs:  
  + Trained model from Jay & Tejas
  + Predicted class probabilities
  + True labels from test set
* Outputs:  
  + Accuracy, precision, recall, F1-score, Specificity
  + Confusion matrix

**Module Dependencies:**

* Depended on:  
  + Final trained model (from Jay and Tejas)
  + Test dataset (curated and split by Devanshu)
* Provided outputs to:  
  + Report generation
  + Model selection and optimization decisions

**Module Design:**

* Built the following evaluation components:  
  + Metric Generator: Accuracy, precision, recall, F1-score
  + Confusion Matrix Plotter
* Metrics were calculated for every model variation (base CNNs, hybrid DeepTumorNet, etc.)

**Module Implementation:**

* Used sklearn.metrics to implement classification metrics:



Fig 17. Classification Metrics

**Module Testing Strategies:**

* Verified consistency of metrics across different folds and runs
* Compared results of base models vs. hybrid model (DeepTumorNet)
* Logged and analyzed metric trends across training epochs

**Module Deployment:**

* Model analysis based on the scores calculated.
* Worked with Devanshu to prepare metric outputs for frontend embedding (e.g., dashboards or web apps)
* Created easy-to-read parameters output comparing different model performances

**Problem Statement:**

To develop a cohesive orchestration system that integrates preprocessing, model training, performance evaluation, and deployment workflows for DeepTumorNet in an efficient and scalable manner.

**Name of the Student: Devanshu Surana**

**Module Title: Workflow Orchestration and System Integration**

**Project’s Module Objectives - Individual Perspective:**

* To design and implement a seamless pipeline that connects data ingestion, model training, and result evaluation modules.
* To ensure modular, maintainable, and scalable orchestration using automation and reproducibility principles.
* To manage data flow, experiment tracking, and streamline model deployment.

**Project’s Module Scope - Individual Perspective:**

* Integrated all modules (Jay & Tejas: model, Dharmika: evaluation) into a reproducible pipeline.
* Built preprocessing pipelines to standardize inputs and support augmentation.
* Managed hardware usage across training sessions using checkpoints, callbacks, and GPU allocation strategies.
* Designed final testing and inference interface to allow real-time prediction on new CT Scan images.

**Project’s Module(s) - Individual Contribution:**

**Hardware & Software Requirements:**

* **Hardware**:
  + Intel i7 Processor, NVIDIA GPU (GeForce RTX 3050), 16 GB RAM, 512 SSD
* **Software**:  
  + Python 3.10
  + TensorFlow, Keras, NumPy, Pandas
  + OpenCV, PIL
  + MLflow
  + Jupyter Notebook & Google Colab

**Module Interfaces:**

* Preprocessing interface → Normalizes, resizes images before feeding into model.
* Model trainer interface → Accepts architecture from Jay & Tejas and integrates callbacks and data loaders.
* Evaluation interface → Receives model outputs and forwards to Dharmika’s evaluation pipeline.
* Inference interface → For testing predictions on new images using model.predict().

**Module Dependencies:**

* Receives:  
  + Raw data (CT Scan images)
  + Final model design (Jay & Tejas)
  + Evaluation code (Dharmika)
* Provides:  
  + Standardized data for training
  + Training logs, checkpoints
  + Real-time prediction outputs

**Module Design:**

* **Data Orchestration**:  
  + Preprocessing: Resizing (224×224), normalization
* **Training Automation**:  
  + Implemented callbacks for early stopping, checkpoint saving, and learning rate adjustments
* **Pipeline**:  
  + Developed a stepwise execution pipeline in Jupyter/Colab:  
    - Step 1: Load and preprocess dataset
    - Step 2: Load hybrid model architecture
    - Step 3: Compile and fit model
    - Step 4: Evaluate and visualize metrics
    - Step 5: Save model

**Module Implementation:**

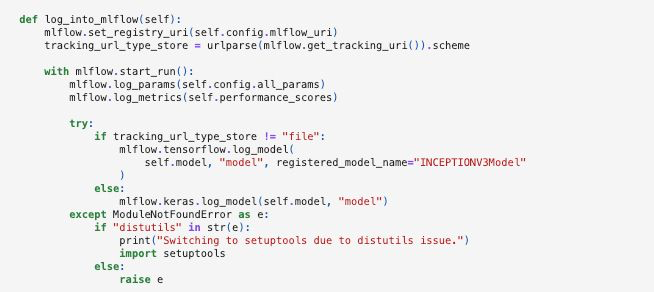
* Sample pipeline snippet:  
  

Fig 18. Model Orchestration

**Module Testing Strategies:**

* Verified preprocessing consistency using visual checks
* Ran pipeline on various CT Scan image types to ensure compatibility
* Tested pipeline robustness with incomplete/corrupted inputs

**Module Deployment:**

* Packaged model and dependencies for easy transfer across systems
* Provided support for Flask-based web interface
* Generated .h5 model + preprocessing scripts for potential cloud/API deployment
* Contributed to final system documentation

## **Applications**

The DeepTumorNet project has significant real-world value in medical diagnostics and research, offering multiple applications within the healthcare ecosystem.

Firstly, it provides clinical diagnosis support by aiding radiologists and oncologists in accurately diagnosing lung cancer, reducing human error and prioritizing critical cases. In regions with limited access to medical imaging expertise, the system can be deployed on cloud platforms or mobile devices, enabling early detection and improved access to cancer care in remote areas.

The system can also be integrated into hospital PACS and EHRs, automating scan assessments and offering real-time diagnostic suggestions. For medical education, DeepTumorNet serves as a teaching tool for students and trainees, helping them understand the visual differences between normal and cancerous lung tissue.

Moreover, it can assist in AI-assisted screening programs, automatically flagging suspicious cases from large datasets of CT scans to accelerate the screening process. Lightweight models like NASNetMobile can power mobile diagnostic applications, providing on-the-go diagnosis or second opinions for general physicians. Finally, DeepTumorNet's modular, version-controlled architecture allows researchers to replicate experiments, benchmark models, and expand the system for other medical image classification tasks, such as detecting pneumonia or COVID-19.

## **Conclusion and Future Prospects**

The DeepTumorNet project successfully delivers an end-to-end, AI-driven lung cancer classification system, achieving high accuracy (97.06%) through the use of CNNs and transfer learning. It features a reproducible and scalable pipeline, supports real-time prediction via a REST API, and integrates modern tools like MLflow, DVC, and Flask for seamless deployment, tracking, and versioning. The system demonstrated consistent performance across evaluation metrics and is designed to be modular, making it highly adaptable for future updates or architectural improvements.

Beyond technical achievements, DeepTumorNet addresses critical challenges in medical imaging such as delayed interpretation and diagnostic inconsistency, offering a deployable solution aligned with global healthcare modernization efforts. Its modular design promotes extensibility, while collaboration with medical experts ensures clinical relevance. The use of lifecycle tools enhances maintainability and audit readiness, and the RESTful API enables integration into a wide range of platforms—from hospital systems to telemedicine applications—making diagnostics more accessible and efficient.

Looking forward, DeepTumorNet holds strong potential for clinical integration in systems like PACS, helping radiologists reduce errors and improve decision-making. Incorporating multimodal data such as EHRs and genetic markers can enable more precise and personalized diagnostics. Techniques like domain adaptation and federated learning can enhance generalization across diverse healthcare environments. Integration of explainable AI tools (e.g., Grad-CAM, LIME) will further build clinician trust by offering transparency in predictions.

The framework also opens up possibilities for edge deployment in under-resourced areas using lightweight CNNs and optimization techniques. Real-time AR/VR feedback systems could augment radiologist workflows during live scans. As an open-source platform, DeepTumorNet fosters academic exploration, enabling students and researchers to experiment with advanced deep learning models like Vision Transformers or Graph Neural Networks.

Ultimately, DeepTumorNet contributes to the global mission of democratizing healthcare by offering a scalable, cost-effective AI diagnostic tool. With localization and language support, it has the potential to serve diverse populations worldwide. As a forward-looking healthcare innovation, DeepTumorNet not only showcases the power of responsible AI but also stands poised to transform the early detection and diagnosis of lung cancer across clinical settings.

References

1. P. Keerthi, S. S. S. Y., and I. S., "Lung cancer classification using deep neural networks," 2025 AI-Driven Smart Healthcare for Society 5.0, Kolkata, India, 2025, pp. 1–6, doi: 10.1109/IEEECONF64992.2025.10963162.
2. N. S. Jozi and G. A. Al-Suhail, "Lung cancer detection in radiological imaging using deep learning: A review," 2024 5th International Conference on Communications, Information, Electronic and Energy Systems (CIEES), Veliko Tarnovo, Bulgaria, 2024, pp. 1–8, doi: 10.1109/CIEES62939.2024.10811230.
3. S. Marappan, S. Roy, B. Ankayarkanni, S. Revathy, and P. Asha, "Enhancing predictive accuracy in lung disease diagnosis through hybrid ResNet and transfer learning models," 2024 International Conference on Advances in Modern Age Technologies for Health and Engineering Science (AMATHE), Shivamogga, India, 2024, pp. 1–7, doi: 10.1109/AMATHE61652.2024.10582190.
4. O. Khouadja and M. S. Naceur, "Lung cancer detection with machine learning and deep learning: A narrative review," 2023 IEEE International Conference on Advanced Systems and Emergent Technologies (IC\_ASET), Hammamet, Tunisia, 2023, pp. 1–8, doi: 10.1109/IC\_ASET58101.2023.10150913.
5. I. V. and D. Menaka, "Real-time detection of lung cancer using CNN," 2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN), Vellore, India, 2023, pp. 1–6, doi: 10.1109/ViTECoN58111.2023.10157316.
6. B. Sumithra, G. Vallathan, M. Raman Kumar, and K. Govindharaju, "Deep learning for accurate chest disease classification: A CNN-based approach for lung cancer subtypes and normal cells," 2023 International Conference on System, Computation, Automation and Networking (ICSCAN), Puducherry, India, 2023, pp. 1–7, doi: 10.1109/ICSCAN58655.2023.10394855.
7. M. Mamun, A. Farjana, M. Al Mamun, and M. S. Ahammed, "Lung cancer prediction model using ensemble learning techniques and a systematic review analysis," 2022 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, 2022, pp. 187–193, doi: 10.1109/AIIoT54504.2022.9817326.
8. N. Nawreen, U. Hany, and T. Islam, "Lung cancer detection and classification using CT scan image processing," 2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI), Rajshahi, Bangladesh, 2021, pp. 1–6, doi: 10.1109/ACMI53878.2021.9528297.
9. A. Rehman, M. Kashif, I. Abunadi, and N. Ayesha, "Lung cancer detection and classification from chest CT scans using machine learning techniques," 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA), Riyadh, Saudi Arabia, 2021, pp. 101–104, doi: 10.1109/CAIDA51941.2021.9425269.
10. R. D. Karthikeyan, R. G., V. V., G. B. C., and K. M., "A review of lung cancer detection using image processing," 2021 Smart Technologies, Communication and Robotics (STCR), Sathyamangalam, India, 2021, pp. 1–4, doi: 10.1109/STCR51658.2021.9588835.
11. A. Yadav and R. Badre, "Lung carcinoma detection techniques: A survey," 2020 12th International Conference on Computational Intelligence and Communication Networks (CICN), Bhimtal, India, 2020

Base Paper



Plagiarism Report

