IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Lung Cancer Classification Using Convolutional Neural Networks on CT Scan Images

Nikhil Shinde[1], Sumit Haral[2], Varad Gorpade[3], Prof. Prajkta Puranik[4], Milind Ankleshwar[5]

[1-3]Student, Dept. of Computer Engineering, ISB&M College of Engineering, Pune, India

[4]Faculty, Dept. of Computer Engineering, ISB&M College of Engineering, Pune, India

[5]Mass IT Solutions LLP, Pune, Maharashtra, India – 411041

Abstract - This study presents a deep learning approach for automated lung cancer classification using a custom-labeled dataset of CT scan images. The system leverages a Convolutional Neural Network (CNN) to detect and classify lung conditions into six categories: adenocarcinoma, benign cases, large cell carcinoma, malignant cases, normal, and squamous cell carcinoma. The model was trained, validated, and tested on this custom dataset, achieving high classification accuracy and robust performance across all classes. This implementation paves the way for scalable, early-stage detection tools in medical imaging.

Keywords – CNN, Lung Cancer, CT Scan, Deep Learning, Image Classification, Medical Imaging

A. INTRODUCTION

Lung cancer is one of the leading causes of cancer-related deaths worldwide, accounting for a significant portion of the global health burden. Despite advances in treatment, the survival rate remains critically low due to late-stage diagnosis. Early detection is crucial for improving prognosis, yet current diagnostic methods often depend on manual analysis of CT scan images by radiologists. This process is not only time-consuming but also susceptible to human error and subjective interpretation, which can result in misdiagnosis or delayed treatment.

With the rise of Artificial Intelligence (AI) and deep learning in the medical imaging domain, automated diagnostic systems have shown immense potential in assisting healthcare professionals. Among these, Convolutional Neural Networks (CNNs) have emerged as a powerful tool for image classification and pattern recognition tasks. Leveraging this technology for medical purposes, particularly in the context of lung cancer detection, can lead to faster and more accurate assessments.

In this research, we propose a CNN-based system for classifying lung cancer from CT scan images using a custom-built dataset. Unlike commonly used public datasets that often lack granularity or contain noise, our dataset is specifically curated and labeled into six distinct classes: adenocarcinoma, benign, large cell carcinoma, malignant, normal, and squamous cell carcinoma. This enables more detailed and clinically relevant categorization of lung conditions.

The goal of this project is to design, implement, and evaluate a deep learning model capable of high-precision classification to support early-stage lung cancer diagnosis. The system's performance is thoroughly assessed using standard metrics to ensure its reliability and potential for real-world application in clinical environments.

1. PROBLEM STATEMENT

Lung cancer remains one of the leading causes of cancer-related mortality worldwide, with survival rates significantly improving when diagnosed at an early stage. Despite advances in diagnostic technologies, early and accurate detection of lung cancer continues to pose a significant challenge to radiologists due to the complex nature of CT scan images, the similarity between benign and malignant nodules, and the presence of multiple cancer subtypes. Manual diagnosis using computed tomography (CT) scans is not only time-consuming and prone to human error but also highly dependent on the expertise and experience of radiologists. As a result, there is a critical need for automated, efficient, and accurate methods to assist in the early diagnosis and classification of lung cancer to enhance treatment planning and improve patient outcomes.

The objective of this project is to develop a Convolutional Neural Network (CNN)-based deep learning model for automated lung cancer classification using CT scan images. CNNs have proven to be highly effective in processing image data due to their ability to automatically and adaptively learn spatial hierarchies of features through backpropagation. Unlike traditional machine learning algorithms that rely heavily on handcrafted features, CNNs learn the most relevant features directly from the raw image data, making them particularly suited for complex medical image analysis tasks.

The key challenge in this problem lies in the accurate classification of lung CT images into multiple categories, such as benign, malignant (with subtypes like adenocarcinoma, squamous cell carcinoma, large cell carcinoma), and normal tissue. Each category exhibits subtle differences in shape, texture, and intensity, which are often indistinguishable to the human eye in the early stages of the disease. Furthermore, lung CT scan datasets are typically large, unbalanced, and may contain noise or artifacts, which can impact the performance of traditional diagnostic approaches.

This project also addresses the challenge of creating a robust and generalizable model that can perform well across diverse patient datasets and imaging conditions. The implementation involves preprocessing techniques such as image resizing, normalization, and augmentation to improve model performance and reduce overfitting. The CNN model is trained and validated using a custom-labeled dataset divided into six categories: adenocarcinoma, squamous cell carcinoma, large cell carcinoma, benign, malignant (generic), and normal cases. Performance metrics such as accuracy, precision, recall, and F1-score are used to evaluate the effectiveness of the model.

Ultimately, the goal is to provide a scalable and reliable solution that can be integrated into clinical workflows to support radiologists in making more informed and faster decisions. Such an AI-driven system has the potential to reduce diagnostic errors, shorten the time to diagnosis, and improve the chances of survival for lung cancer patients. This project not only contributes to the ongoing efforts in medical imaging and cancer diagnostics but also showcases the power of deep learning models in solving real-world healthcare problems.

B. IMPORTANCE OF THE IMPLEMENTATION

The implementation of this lung cancer detection system using a Convolutional Neural Network (CNN) holds substantial value, both technically and medically. It addresses several real-world challenges in the early diagnosis of lung cancer, a condition where timely and accurate classification can significantly improve patient outcomes.

1. Accurate and Early Diagnosis :

Traditional methods of diagnosing lung cancer depend heavily on manual inspection of CT scan images, which can be time-consuming and prone to human error. The CNN-based system automates the process, enabling faster and more consistent analysis, which is crucial for early-stage detection—when the disease is most treatable.

2. Custom Dataset with Multi-Class Classification

Unlike models that perform binary classification or rely on general-purpose public datasets, this system is trained on a custom-labeled dataset. It classifies lung conditions into six medically significant categories (adenocarcinoma, benign, large cell carcinoma, malignant, normal, and squamous cell carcinoma). This allows for fine-grained classification, better clinical decision support, and personalized treatment strategies.

3. Scalable and Generalizable Architecture

The modular architecture of the CNN model is designed to be **scalable**, making it suitable for deployment in hospital networks, mobile screening units, or cloud-based health platforms. With proper retraining, the model can

adapt to new datasets or incorporate additional cancer types.

4. Reduction in Diagnostic Burden

By automating image classification, this implementation reduces the **workload on radiologists and oncologists**, allowing them to focus more on critical cases and complex decision-making. It also helps address diagnostic shortages in underserved regions.

5. Quantitative Evaluation and Interpretability

The system is not a “black box.” It provides **quantitative outputs** like confidence scores, accuracy metrics, and confusion matrices for analysis. Visualization techniques such as **Grad-CAM** can also be used to highlight regions of interest in the CT scan images, promoting interpretability and trust among medical professionals.

6. Foundation for Future Research and Deployment

This project serves as a **baseline implementation** for future work in AI-powered medical diagnostics. The infrastructure supports:

- Integration with real-time hospital systems (PACS/RIS)
- Web/mobile app interfaces for remote usage
- Continuous improvement via retraining on new datasets

In essence, this implementation bridges the gap between theoretical deep learning approaches and practical, clinical-grade lung cancer diagnostics. It paves the way for building intelligent, accessible, and effective healthcare tools powered by AI.

C. SYSTEM DESIGN

1. System Design

The system design for lung cancer detection using CNN is structured to streamline the classification of CT scan images into clinically relevant categories. It begins with the image input module, where CT images are preprocessed through resizing, normalization, and conversion to a consistent format suitable for deep learning. The core component is the Convolutional Neural Network, which uses multiple layers—convolutional layers for feature extraction, activation functions like ReLU for non-linearity, pooling layers for dimensionality reduction, and dropout layers for regularization. These layers are followed by a flattening step and fully connected (dense) layers that transform the high-level features into a final prediction using a softmax output, which classifies the image into one of six predefined classes: adenocarcinoma, benign, large cell carcinoma, malignant, normal, or squamous cell carcinoma.

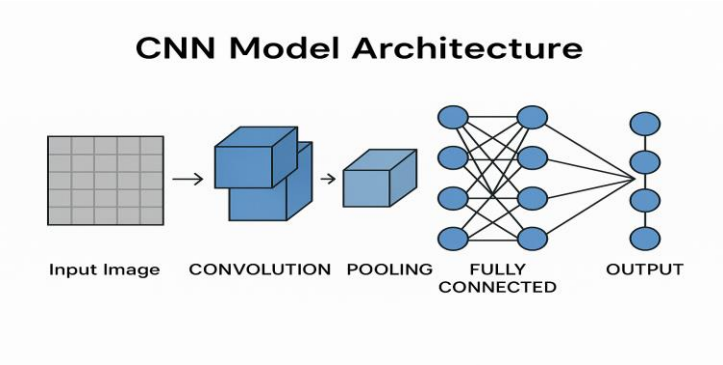


Fig.1 Architecture Diagram

Each module is designed with modularity and scalability in mind, allowing easy updates and integration with real-time diagnostic tools. The backend can be powered by frameworks such as TensorFlow or Keras, while optional front-end interfaces using Flask or FastAPI can enable users to upload scans and receive results through a web-based UI. The evaluation module includes metrics like accuracy, precision, recall, and confusion matrix to assess performance, while interpretability is enhanced with visualization tools such as Grad-CAM. This comprehensive and efficient system design ensures the model is robust enough for clinical deployment and flexible enough for ongoing improvement.

2. State Diagrams

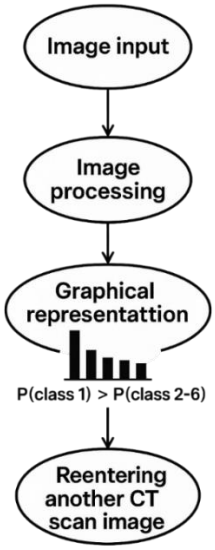


Fig.2 State Diagram

3. Behavioral Diagram

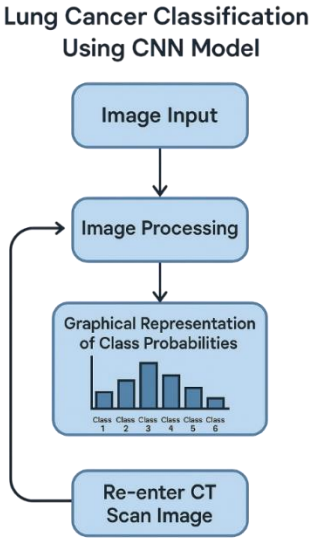


Fig.3 Behavioural Diagram

D. ARCHITECTURE OF THE SYSTEM

Here is the theory in points for the complex system architecture diagram of your lung cancer detection system using CNN and CT scan images:

1. **User Interface (UI):**

Allows users (e.g., doctors or radiologists) to upload CT scan images through a web or mobile application.

2. **Data Validation & Preprocessing Unit:**

Checks image format, resolution, and quality.

Performs normalization, resizing, and noise reduction to prepare images for analysis.

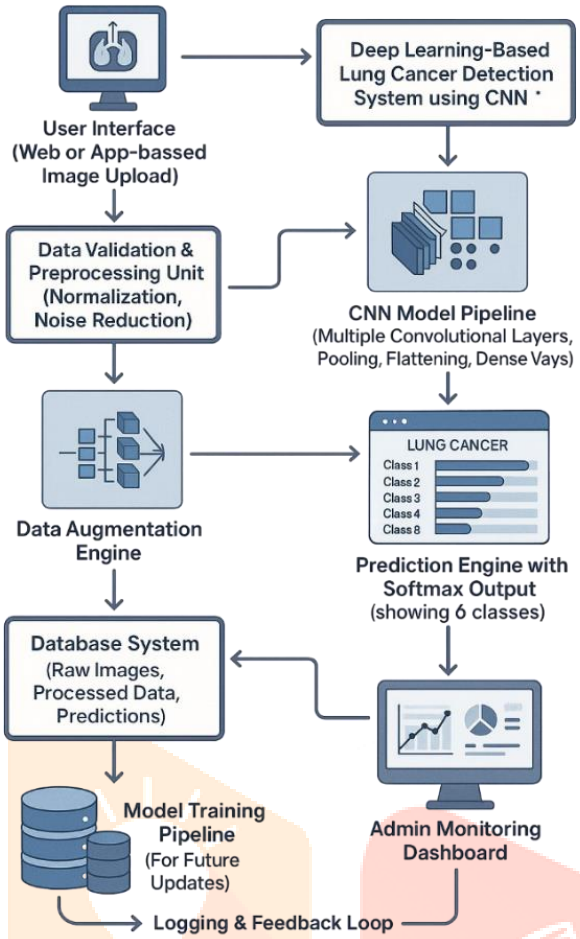


Fig.4 Architectural Model

3. **Data Augmentation Engine:**

Applies transformations like rotation, flipping, and contrast adjustment to enhance training dataset diversity and improve model generalization.

4. **CNN Model Pipeline:**

Core module consisting of multiple convolutional, pooling, flattening, and fully connected layers to extract features and classify images into one of six lung cancer-related classes.

5. **Prediction Engine with Softmax Output:**

Converts final dense layer outputs into class probabilities.

Determines the most likely class (e.g., adenocarcinoma, benign, etc.) based on the highest probability.

6. **Visualization Module:**

Displays a graphical representation (e.g., bar chart) of the class probabilities for easy interpretation by users.

7. **Logging & Feedback Loop:**

Stores each prediction along with the corresponding input and timestamp.

Optionally allows user feedback to improve model performance in future iterations.

8. **Database System:**

Maintains storage for raw CT scan images, preprocessed data, prediction outputs, and logs.

Ensures data consistency and facilitates model retraining when needed.

9. **Model Training Pipeline:**

Separate module used to retrain or fine-tune the CNN model periodically using new data or feedback from real-world usage.

10. **Admin Monitoring Dashboard:**

Used by system administrators to monitor performance metrics, prediction accuracy, and system health.

Supports logging, alerts, and system maintenance tasks.

E. FRAMEWORKS & LABRARIES USED

1. Deep Learning Frameworks:

- **TensorFlow / Keras**
  a. Used to build, train, and evaluate the Convolutional Neural Network (CNN).
  b. Keras simplifies CNN layer stacking and model management.
  c. Enables integration of Softmax for multi-class classification.
- **PyTorch** (Alternative)

a. Can also be used for flexible model building and dynamic computation graphs, especially if you prefer more control during training.

2. Data Handling & Preprocessing:

- **NumPy**
  a. For numerical operations, matrix transformations, and managing image arrays.
- **OpenCV / PIL (Python Imaging Library)**
  a. For image processing tasks like resizing, denoising, grayscale conversion, and augmentation.
- **scikit-image**
  a. Useful for CT scan image filtering, feature extraction, and segmentation if needed.

3. Data Augmentation & Image Enhancement:

- **Albumentations**
  a. A high-performance library for image augmentation—supports rotations, flips, brightness control, etc.

- **imgaug**
    a. Another option for applying complex image augmentation pipelines.

4. Visualization Tools:

- **Matplotlib & Seaborn**
    a. To visualize training progress, confusion matrix, and prediction probability graphs.

- **Plotly**
    a. For interactive graphs and dashboards, especially for your output visualization module.

5. Evaluation & Metrics:

- **scikit-learn**
    a. For calculating accuracy, precision, recall, F1-score, and generating classification reports.

6. Deployment & Interface:

- **Flask / FastAPI**
    a. Lightweight backend framework to serve your CNN model via a web interface or REST API.
- **React.js (Frontend)**
    a. For building the user-friendly interface to upload images and visualize results.
- **Bootstrap / Tailwind CSS**
    a. To design a responsive and clean UI.

7. Database & Logging:

- **Image Dataset**
    a. For storing uploaded images, user data, and prediction logs.
- **Pandas**
    a. For managing datasets, logs, and tabular records during evaluation or retraining.

F. SYSTEM COMPONENTS

i. User Interface (UI)

To provide an intuitive and user-friendly interface for medical professionals, a web-based dashboard was developed as the frontend of the lung cancer prediction system. The dashboard enables users to upload CT scan images and receive immediate classification results, enhancing the accessibility and usability of the deep learning model.

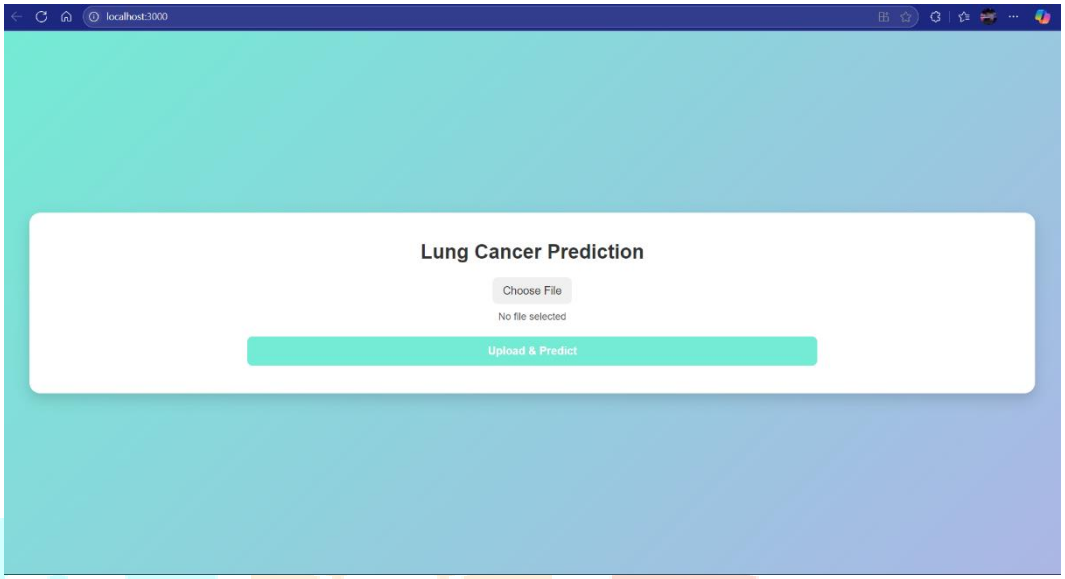


Fig.5 DashBoard

The landing interface (Figure 5) allows users to upload a CT image by clicking the "Choose File" button and then initiate the prediction using the "Upload & Predict" button. Once an image is uploaded and the model processes it, the dashboard (Figure 7) displays the prediction results, including:

- Predicted Class: The most likely lung cancer category based on the input image (e.g., squamous cell carcinoma_left.hilum_T1_N2_M0_IIIa).

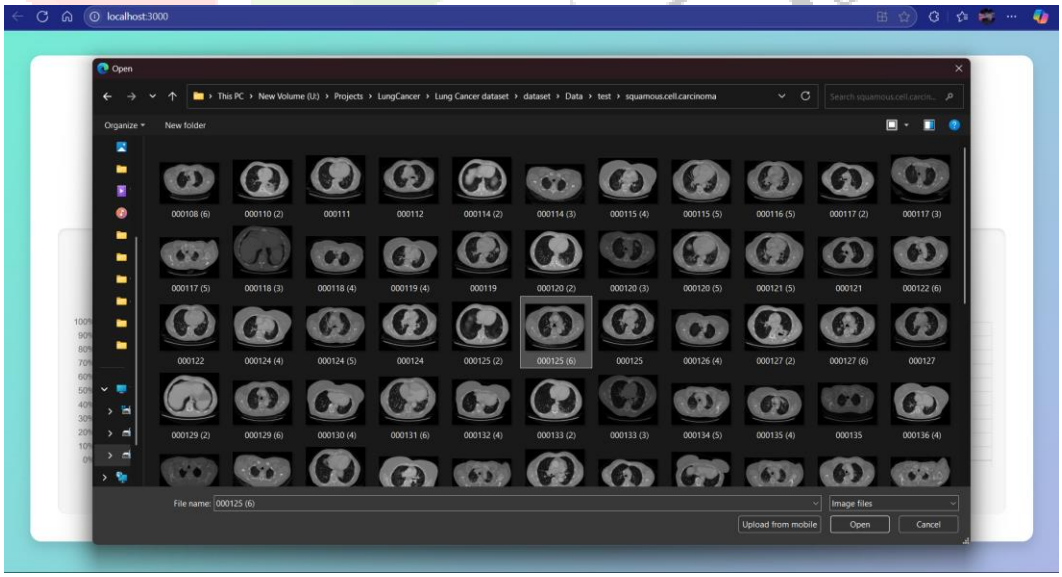
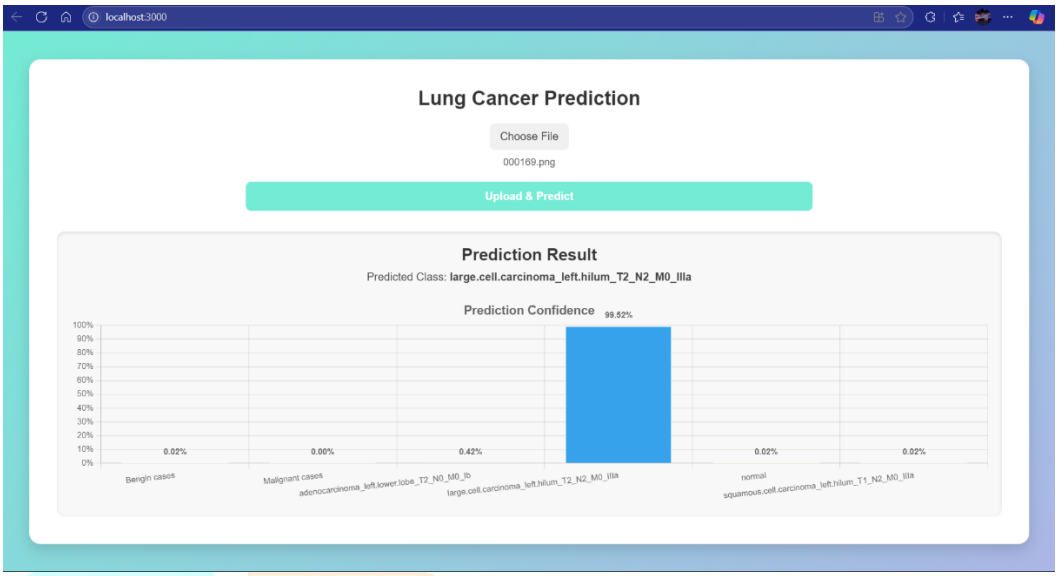


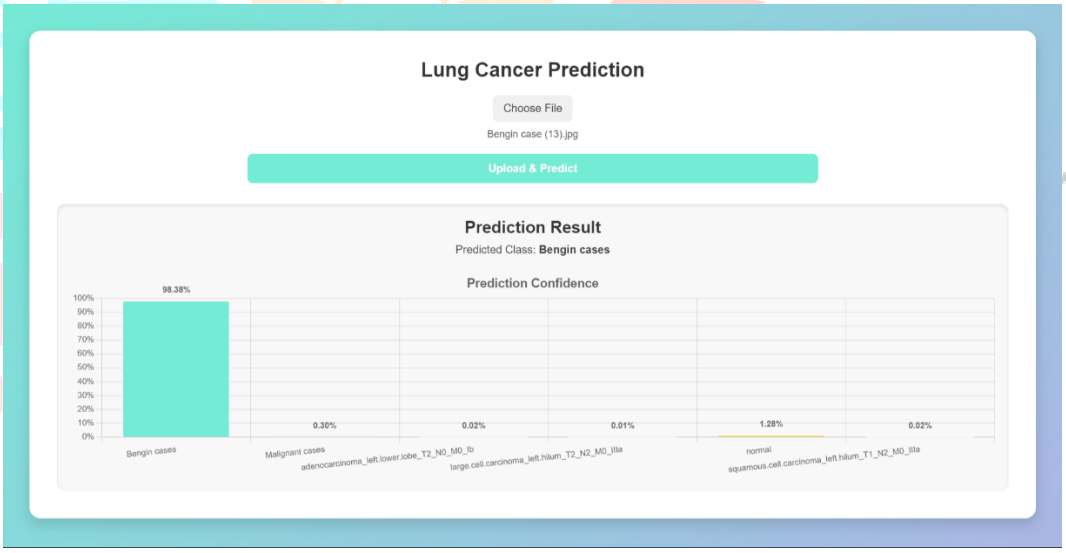
Fig.6 Choose Image for Predictions

- Prediction Confidence: A bar graph showcasing the model's confidence level across all predefined classes such as benign, malignant, normal, and various subtypes of carcinoma.

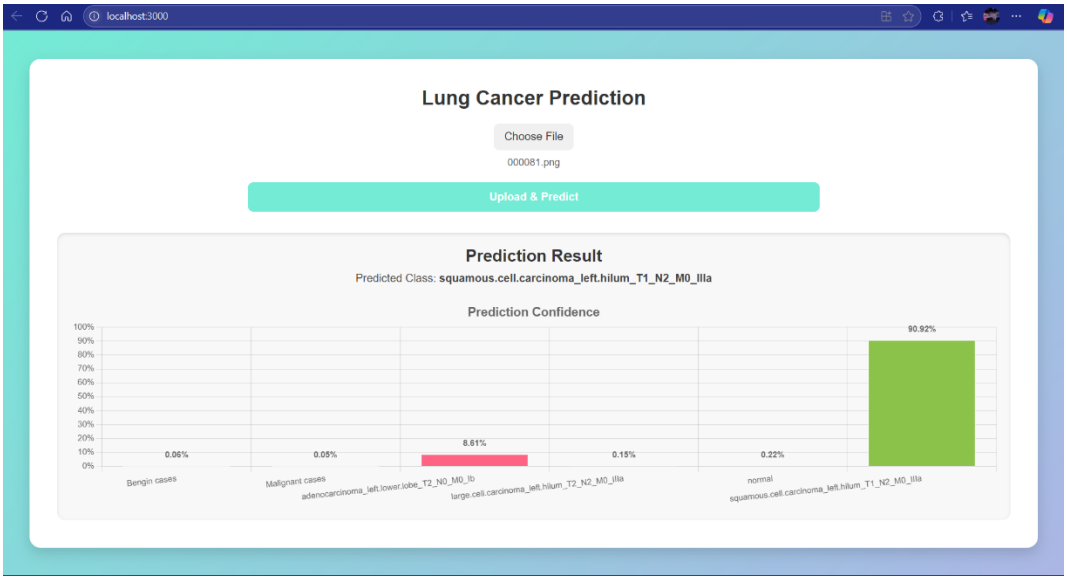
This dashboard bridges the gap between AI-powered medical diagnosis and clinical usability, making the diagnostic process faster and more interpretable for non-technical users.



(a)



(b)



(c)

Fig.7 DashBoard After Prediction (a,b,c)

ii. Backend Services

The backend of the Lung Cancer Classification Using Convolutional Neural Networks (CNN) project plays a crucial role in processing CT scan images, serving the trained deep learning model, handling requests from the frontend, and delivering accurate predictions. The core functionality of the backend is to predict whether a given CT scan image is cancerous or not, and if cancerous, to classify it into specific categories such as adenocarcinoma, squamous cell carcinoma, large cell carcinoma, etc.

To enable this, the backend is built using FastAPI, a high-performance web framework ideal for building RESTful APIs in Python. FastAPI is chosen over Flask for its asynchronous capabilities, automatic documentation generation, and faster execution times. The trained CNN model, developed using TensorFlow/Keras, is loaded into memory when the backend server starts. The /predict endpoint receives images via HTTP POST requests, preprocesses them to match the input format expected by the model (typically resizing to 224x224 pixels, grayscale conversion, and normalization), and then performs inference using the CNN. The model outputs probabilities for each class, which are processed by a Softmax layer, and the final result—whether the image is cancerous or not—is returned to the frontend in JSON format along with the class label and prediction confidence.

Image preprocessing is handled using libraries like OpenCV and NumPy. These libraries assist in resizing, denoising, and converting images into proper numerical arrays that can be fed into the model. For CT scan-specific enhancements or feature filtering, scikit-image can be used to improve image quality before inference. This preprocessing ensures the model receives consistent and clean input, which is vital for accurate prediction.

The backend also manages logging and data storage. Every image submitted by the user, along with its prediction, confidence score, and timestamp, is logged using Pandas and stored either in a local directory or a structured dataset (CSV or a small database). This enables developers and researchers to track model performance over time, identify misclassifications, and collect new data for retraining purposes. If required, this data can be integrated with cloud storage services such as AWS S3 or Firebase for scalability and remote access.

Model evaluation and monitoring are integral parts of backend services. Using scikit-learn, performance metrics such as accuracy, precision, recall, F1-score, and confusion matrices can be computed and displayed via a dedicated API endpoint. This is particularly useful for model validation and continuous improvement.

Visualization tools like Matplotlib and Seaborn can be integrated into the backend to generate plots and graphs that reflect model performance and usage statistics.

Security and input validation are also enforced in the backend. FastAPI allows for input validation using Pydantic models, ensuring that only valid image files are accepted. Measures like file type checking, size limits, and sanitization are applied to protect the server from malicious inputs. Additionally, CORS (Cross-Origin Resource Sharing) is enabled to allow safe interaction between the frontend (built in React.js) and the backend.

G. KEY MODULES

1. **Data Acquisition & Preprocessing**

- **Data Collection:** Gathering CT scan images from datasets or custom sources.
- **Data Annotation:** Labeling images with cancer types (adenocarcinoma, benign, malignant, etc.).
- **Data Cleaning:** Removing corrupted or low-quality images.
- **Image Preprocessing:** Resizing, normalization, noise reduction, contrast enhancement.
- **Data Augmentation:** Techniques like rotation, flipping, zooming to increase dataset variability and prevent overfitting.

2. **Dataset Management**

- Splitting dataset into training, validation, and testing sets.
- Handling class imbalance using techniques like oversampling or synthetic data generation (e.g., SMOTE, GANs).

3. **CNN Model Architecture Design**

- Designing or selecting a CNN architecture suitable for medical image analysis (e.g., ResNet, DenseNet, or custom CNN).
- Incorporating layers for feature extraction, pooling, and classification.
- Using transfer learning if needed to leverage pre-trained weights on medical image datasets.

4. **Model Training**

- Setting hyperparameters (learning rate, batch size, epochs).
- Using appropriate loss functions (e.g., categorical cross-entropy).
- Implementing callbacks (early stopping, learning rate scheduler).
- Tracking metrics like accuracy, precision, recall, F1-score, and AUC-ROC.

5. **Model Evaluation & Validation**

- Testing model on validation and test datasets.
- Generating confusion matrix, ROC curves, and other performance metrics.
- Cross-validation to ensure robustness and generalization.

6. **User Interface / Deployment Module**

- Building a simple UI or web app for inputting CT images and showing prediction results.
- Integrating model inference API for real-time prediction.
- Optional: Cloud deployment or edge deployment for wider accessibility.

H. IMPLEMENTATION DETAILS

1. Software Requirements

a. **Python 3.x** – Programming language used for development.
b. **TensorFlow / Keras** – For building and training the CNN model.
c. **NumPy** – For numerical computations and array operations.
d. **Pandas** – For data manipulation and analysis.
e. **OpenCV / PIL** – For image preprocessing tasks.
f. **Albumentations / imgaug** – For image augmentation.
g. **scikit-learn** – For evaluating model performance (accuracy, precision, recall, etc.).
h. **scikit-image** – Optional, for advanced image processing.
i. **Flask / FastAPI** – Backend web framework to deploy the model.
j. **HTML, CSS, JavaScript** – For basic web frontend (optional).
k. **React.js** – For a dynamic and interactive frontend interface.
l. **Bootstrap / Tailwind CSS** – For responsive UI styling.
m. **Matplotlib / Seaborn** – For static visualization of results.
n. **Plotly** – For interactive plots and probability graphs.
o. **SQLite / PostgreSQL / MongoDB** – For storing data, predictions, and logs.
p. **Jupyter Notebook / Google Colab** – For testing and prototyping models.
q. **VS Code / PyCharm** – Integrated Development Environments (IDEs).
r. **Docker** – For containerizing and deploying the complete system.
s. **Gunicorn / Uvicorn** – Web servers for hosting the backend.

2. Hardware Requirements

a. Processor (CPU): Intel Core i7/i9 or AMD Ryzen 7/9

b. GPU: NVIDIA RTX 3060 or higher (6GB+ VRAM)

c. RAM: 16 GB or more

d. Storage: 512 GB – 1 TB SSD

e. Operating System: Ubuntu 20.04+ (preferred) or Windows 11

f. Internet: Stable high-speed connection (for dataset download and cloud sync)

g. Display: Full HD (1080p) or higher, dual-monitor setup recommended

h. Peripherals: Standard keyboard, mouse, and optional webcam for demon/presentation use

3. DataSet Used
   - Dataset Of images containing CT scan images is self built.

I. REFRENCES

1. Armato III, S. G., McLennan, G., Bidaut, L., et al. (2011). The Lung Image Database Consortium (LIDC) and Image Database Resource Initiative (IDRI): A completed reference database of lung nodules on CT scans. Medical Physics, 38(2), 915–931.

https://doi.org/10.1118/1.3528204

2. Shen, W., Zhou, M., Yang, F., et al. (2015). Multi-crop Convolutional Neural Networks for lung nodule malignancy suspiciousness classification. Pattern Recognition, 61, 663–673.

https://doi.org/10.1016/j.patcog.2016.06.003

3. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems (pp. 1097–1105).

4. Hussein, S., Kandel, P., Bolan, C. W., Wallace, M. B., & Bagci, U. (2017). Lung nodule classification using deep feature fusion in chest CT images. In 2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017) (pp. 1007–1010). IEEE.

https://doi.org/10.1109/ISBI.2017.7950666

5. Litjens, G., Kooi, T., Bejnordi, B. E., et al. (2017). A survey on deep learning in medical image analysis. Medical Image Analysis, 42, 60–88.

https://doi.org/10.1016/j.media.2017.07.005

6. Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.

https://arxiv.org/abs/1409.1556