Automated Medical Image Analysis for Disease Diagnosis Using Deep Learning: A Web-Based System Implementation

Kanish Shandilya[1], Devansh Dhingra[2], Komal Kashyap[3], Pooja Singh[4]
[1,2,3] *Department of Computer Science Engineering, Mahatma Gandhi Mission's College of Engineering & Technology, Noida, NCR, India*
[4] *Guide, Department of Computer Science and Engineering, MGM's College of Engineering and Technology, Noida, India*

***Abstract*—Deep learning-based medical image analysis has shown strong potential in improving diagnostic accuracy across various clinical tasks. However, many existing works remain limited to model development and lack practical deployment frameworks. This paper presents a web-based implementation of an automated medical image analysis system that integrates multiple deep learning models into a unified platform.**

Building upon previously developed convolutional neural network (CNN) models for pneumonia detection [3] and cardiac disease classification [4], and a U-Net-based model for atrium segmentation [1], this work focuses on bridging the gap between model development and deployment. The models, trained using PyTorch in a Jupyter Notebook environment, are exported as checkpoint files and integrated into a Flask-based backend for real-time inference.

The system supports multiple diagnostic workflows and provides functionalities such as user authentication, image upload, dynamic model selection, and result visualization through confidence scores, activation heatmaps, and segmentation overlays. A database system is also incorporated to store user activity and analysis history.

The results demonstrate that deep learning models can be effectively deployed within a structured web-based environment, improving accessibility and practical usability. This work highlights the importance of system-level integration in translating machine learning models into real-world diagnostic tools.

***Index Terms*—Deep Learning, Medical Image Analysis, Web-Based System, Flask, Convolutional Neural Network, U-Net, Image Segmentation, Model Deployment**

I. INTRODUCTION

Medical image analysis plays a critical role in modern healthcare by enabling early detection and diagnosis of diseases. Traditionally, interpretation of medical images such as chest X-rays and cardiac MRI scans relies on expert analysis, which can be time-consuming and subject to variability. With the advancement of deep learning techniques, particularly Convolutional Neural Networks (CNNs) [2] and U-Net architectures [1], automated systems have demonstrated strong performance in classification and segmentation tasks [5].

In recent years, several studies have focused on developing deep learning models for specific diagnostic tasks such as pneumonia detection [3] and anatomical structure segmentation [5]. These models are capable of learning complex patterns directly from image data and have achieved high accuracy. However, a major limitation of many existing works is that they remain confined to experimental environments and lack practical deployment mechanisms for real-world use.

In prior work, a modular deep learning-based system was developed for multiple medical image analysis tasks, including pneumonia detection [3], cardiac disease classification [4], and atrium segmentation [1]. The models were trained using PyTorch [9] in a controlled environment and demonstrated reliable performance. Despite these results, the system was

limited to a model-centric implementation and did not address usability or accessibility for end users.

To overcome these limitations, this work focuses on the design and implementation of a web-based system that integrates the previously developed models into a unified, user-interactive platform. The system enables users to upload medical images, select diagnostic tasks, and obtain results in real time through a structured interface.

Unlike many existing approaches that focus solely on algorithmic performance, the proposed system emphasizes end-to-end integration, incorporating model deployment, user interaction, data management, and result visualization within a single framework. The inclusion of visual outputs such as activation heatmaps and segmentation overlays further enhances interpretability.

This work demonstrates how deep learning models can be transformed from isolated research components into usable systems, highlighting the importance of deployment and system-level design in medical image analysis applications.

II. SYSTEM ARCHITECTURE

The proposed system follows a modular architecture designed to integrate deep learning models with a user-accessible web interface. It consists of four primary components: frontend interface, backend server, model layer, and database system. These components work together to enable seamless interaction between users and the underlying models.
The overall workflow of the system is illustrated in Fig. 1. The process begins with user interaction through the web interface, where users can log in, select a diagnostic module, and upload medical images. The request is processed by the backend server, which handles routing, validation, and communication with the model layer. The selected model performs inference, and the results are returned to the user in both numerical and visual formats. The outputs are also stored in a database for future reference.

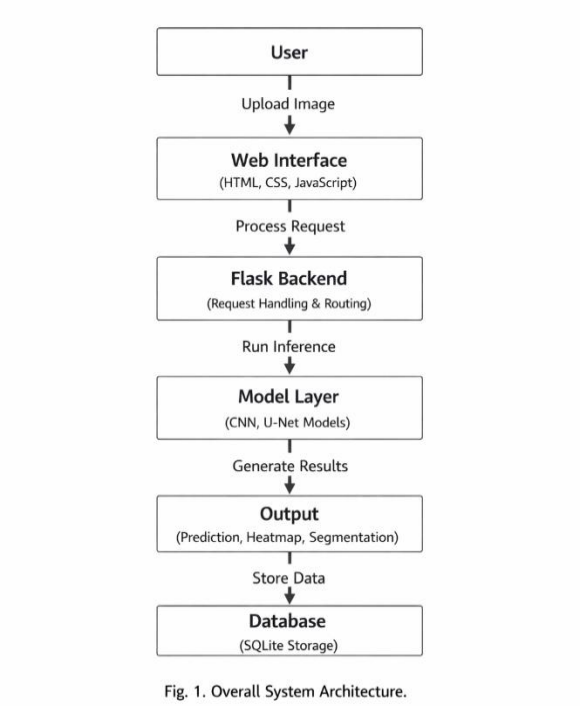

Fig. 1. Overall System Architecture.

A. Frontend Interface

The frontend is developed using HTML, CSS, and JavaScript, providing an intuitive and interactive user experience. It allows users to navigate between diagnostic modules, upload images, and view results in a structured format. The interface is designed for clarity, ensuring that users can perform analyses without requiring technical expertise.

B. Backend Server

The backend is implemented using the Flask web framework, which acts as the central controller of the system. It manages routing, request processing, and communication between system components. The backend is also responsible for user authentication, file handling, and response generation.
It identifies the selected diagnostic task and directs the input to the appropriate model. Session management is implemented to ensure secure access to the analysis workspace.

C. Model Layer

The model layer consists of pre-trained deep learning models developed using PyTorch [9]. These models are trained in a Jupyter Notebook environment and exported as checkpoint files for deployment. Separate models are used for pneumonia detection, cardiac classification, and atrium segmentation.

During runtime, the required model is dynamically selected based on user input. The data is pre-processed before being passed to the model, and the output is post-processed to generate interpretable results such as confidence scores, activation heatmaps, and segmentation masks.

D. Database System

A SQLite database is used to store user credentials and maintain a history of performed analyses. This enables persistent storage of results and allows users to access previous analyses.

The integration of a database supports user-specific workflows and enhances the practical usability of the system.

III. METHODOLOGY

The methodology focuses on integrating pre-trained deep learning models into a functional web-based pipeline. Unlike model-centric approaches, this work emphasizes the transition from model training to deployment and user interaction.

The process can be divided into four stages: model training, model export, system integration, and inference execution.

A. Model Training and Export

The deep learning models are developed and trained in a Jupyter Notebook environment using the PyTorch framework [9]. Separate models are trained for pneumonia detection [3], cardiac disease classification [4], and atrium segmentation [1].

After training, the models are saved as checkpoint files (.pth format), which store the learned parameters. These files are later used in the deployment phase, eliminating the need for retraining.

B. Model Integration

The trained models are integrated into the backend using a modular loading mechanism. During application startup, models are loaded from their respective checkpoint files and initialized for inference.

The system dynamically selects the appropriate model based on the user's chosen diagnostic task. This modular approach allows independent execution of each model and supports easy extension of the system.

C. Input Processing and Validation

The system supports different input formats depending on the task. For classification, chest X-ray images in formats such as PNG and JPG are accepted. For segmentation, medical imaging data in NIfTI format (.nii, .nii.gz) is processed [6].

Uploaded inputs are validated and pre-processed before inference. Image preprocessing includes resizing and normalization, while MRI data is processed to extract relevant slices for analysis.

D. Inference Pipeline

During inference, the validated input is passed to the selected model for prediction. The output is then post-processed to generate interpretable results.

For classification tasks, the system produces probability scores along with activation heatmaps highlighting important regions of the image [2]. For segmentation tasks, the model generates binary masks [1], which are combined with the original image to produce overlay visualizations.

E. Result Generation and Storage

The final outputs are formatted into structured results that include prediction scores, visual artifacts, and explanatory information. These results are displayed to the user through the web interface.

The system also stores analysis results in a database, allowing users to access previous analyses and maintain continuity in their workflow.

IV. IMPLEMENTATION

The system is implemented using Python and the Flask web framework, providing an efficient environment for deploying deep learning models within a web-based interface. The implementation combines model execution, user interaction, data storage, and result generation into a unified application.

A. Backend Implementation

The backend is developed using Flask, which manages routing, request handling, and communication between system components. It supports functionalities such as user authentication, workspace navigation, model execution, and report generation.

User sessions are managed to ensure secure access to the system. The backend also handles error conditions

such as unsupported file formats and invalid inputs, improving reliability during operation.

B. Model Loading and Execution

The trained models are stored as checkpoint files and loaded dynamically during runtime. A structured loading mechanism ensures that each model is initialized efficiently.

Based on the selected diagnostic task, the appropriate model is executed for inference. The system supports both classification and segmentation workflows, allowing flexible task selection within a single platform.

C. File Handling and Processing

The application includes a structured file management system for handling uploaded and generated data. Uploaded files are stored securely, and naming conventions are used to avoid conflicts.

For classification tasks, image inputs are processed using standard techniques [8]. For segmentation tasks, volumetric MRI data is handled using specialized utilities to extract relevant slices [6]. Generated outputs such as heatmaps and masks are stored separately for display and retrieval.

D. Visualization and Output Generation

To enhance interpretability, the system generates visual outputs alongside numerical predictions. For classification tasks, activation heatmaps highlight regions influencing the model's decision.

For segmentation tasks, predicted masks are combined with original images to create overlay visualizations [1]. These outputs provide clearer insight into model predictions and improve user understanding.

The system presents results in a structured format that includes prediction scores, explanations, and visual artifacts.

E. Database Integration

A SQLite database is used to manage user data and store analysis history. It maintains records of user interactions, including selected tasks and generated results.

This feature enables users to revisit previous analyses and supports a continuous workflow within the system.

F. Report Generation

The system includes a reporting feature that generates downloadable summaries of analysis results. These reports contain prediction outcomes, confidence values, and explanatory details.

This functionality extends the usability of the system by allowing results to be stored and shared outside the application.

V. RESULTS AND DISCUSSION

The developed system was evaluated based on its ability to execute multiple diagnostic tasks within a unified web-based environment and provide meaningful outputs to users. The evaluation focuses on usability, output clarity, and the effectiveness of integrating trained models into a deployable system.

A. User Interface and Interaction

The system provides a structured and user-friendly interface that allows smooth navigation between different diagnostic modules. The homepage presents an overview of the platform and guides users toward the analysis workspace.

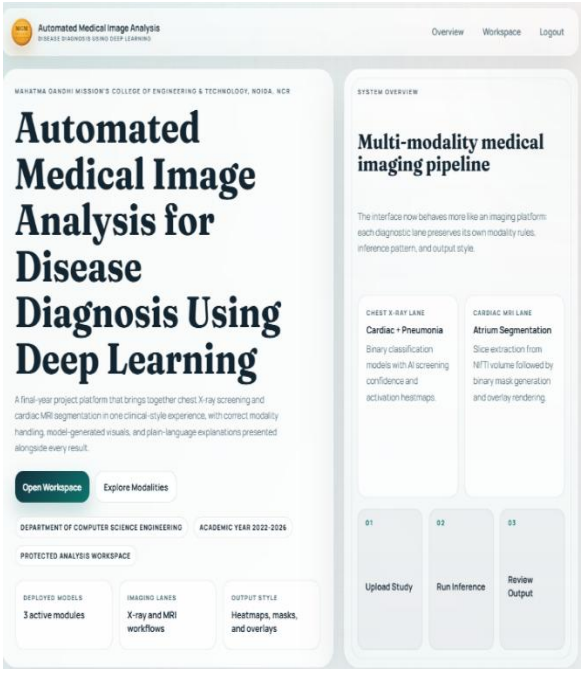

Fig. 2. Web Interface Homepage

Within the workspace, users can select from multiple diagnostic modules, including pneumonia detection, cardiac classification, and atrium segmentation.

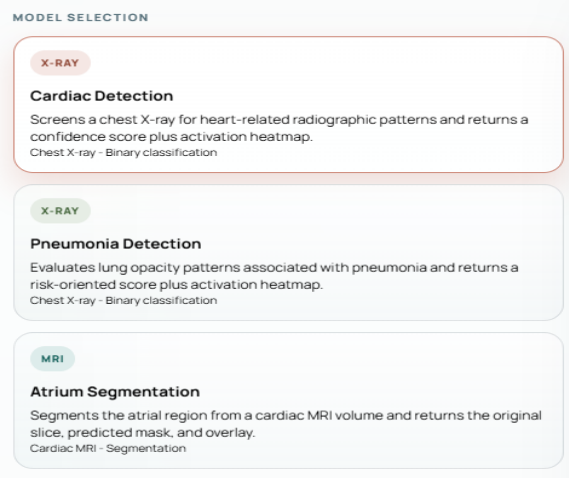

Fig. 3. Module Selection Interface

The interface design ensures that users can perform tasks with minimal effort, making the system accessible even to non-technical users.

B. Image Upload and Processing

The platform allows users to upload medical images based on the selected task. Input validation is performed to ensure that only supported formats are accepted.

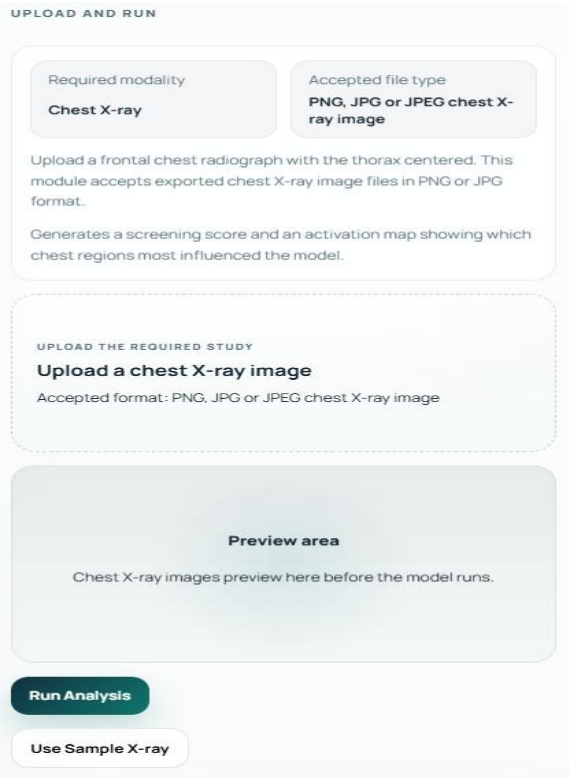

Fig. 4. Image Upload Interface

After uploading, the system processes the input and directs it to the appropriate model. The preprocessing stage ensures that the input conforms to model requirements before inference.

C. Output Generation and Visualization

The system produces outputs in both numerical and visual forms. For classification tasks, prediction results are presented along with confidence scores and explanatory text.

Fig. 5. Model Output with Confidence and Explanation

To improve interpretability, the system generates activation heatmaps that highlight the regions influencing the prediction. These visual cues provide additional insight into the model's behaviour.

For segmentation tasks, the system produces binary masks and overlay visualizations that clearly represent the predicted regions of interest [1].

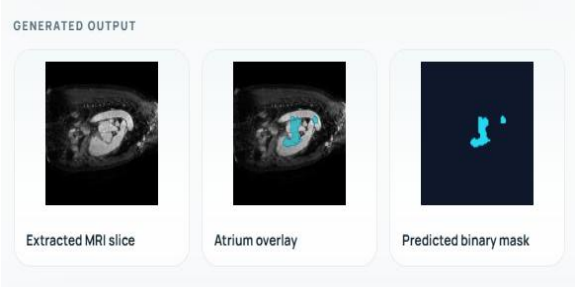

Fig. 6. Visualization of Model Prediction (Heatmap/Segmentation Overlay)

D. System Performance and Reliability

The system successfully performs inference across multiple diagnostic tasks within a single framework. The modular design allows independent execution of each model while maintaining a consistent user experience.

The inclusion of visual outputs enhances interpretability, while database integration ensures that analysis results are stored and can be retrieved when needed.

E. Discussion

The results demonstrate that deep learning models can be effectively integrated into a web-based system, making them accessible beyond research environments [5]. The platform not only delivers predictions but also presents them in a form that is easier to interpret.

A key strength of the system is its end-to-end integration, combining model execution, visualization, and user interaction within a single workflow. This approach emphasizes practical usability rather than focusing solely on model performance.

However, system performance may vary depending on input quality and computational resources. Future improvements can focus on optimizing processing efficiency and extending system capabilities.

Additionally, detailed evaluation metrics such as confusion matrix and ROC analysis are part of the model evaluation phase discussed in earlier work, and are not repeated in this system-level implementation.

VI. CONCLUSION

This paper presented a web-based implementation of an automated medical image analysis system using deep learning models. Building on previously developed models for pneumonia detection, cardiac disease classification, and atrium segmentation, the work focused on translating model-level results into a usable application.

The system integrates multiple diagnostic workflows within a single platform, allowing users to upload medical images and obtain results through an interactive interface. The use of a Flask-based backend, along with a modular model integration approach, enables efficient execution of trained models in real time. Additional features such as user authentication, analysis history tracking, and report generation improve the overall usability of the system.

A key contribution of this work is the end-to-end integration of model training, deployment, and user interaction. The inclusion of visualization techniques such as activation heatmaps and segmentation overlays further improves interpretability and helps users better understand model outputs.

The results indicate that deep learning models can be effectively deployed in a web-based environment, making them more accessible for practical use. This highlights the importance of system-level implementation in bridging the gap between research models and real-world applications.

Future work may include deployment on cloud platforms, support for larger and more diverse datasets, and improvements in processing efficiency. Extending the system to additional diagnostic tasks and enabling real-time analysis can further enhance its applicability.

REFERENCES

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," MICCAI, 2015, pp. 234–241.

[2] Alex Krizhevsky, I. Sutskever, and Geoffrey Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," NeurIPS, 2012.

[3] D. Rajpurkar et al., "CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning," arXiv:1711.05225, 2017.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," CVPR, 2016.

[5] G. Litjens et al., "A Survey on Deep Learning in Medical Image Analysis," Medical Image Analysis, vol. 42, pp. 60–88, 2017.

[6] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation," 3DV, 2016.

[7] J. Deng et al., "ImageNet: A Large-Scale Hierarchical Image Database," CVPR, 2009.

[8] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv:1409.1556, 2014.

[9] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016.

[10] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2016.