Medical Prognosis using DenseNet and ResNet

A Mini-Project Report

Submitted to the APJ Abdul Kalam Technological University

in partial fulfillment of requirements for the award of degree

Bachelor of Technology

in

Computer Science and Engineering (Artificial Intelligence and Machine Learning)

by

Abhimanyu Pradeep(SCT21AM004)

Abineet Thampi(SCT21AM007)

Harrel Jacob Alex(SCT21AM031)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SREE CHITRA THIRUNAL COLLEGE OF ENGINEERING TRIVANDRUM

KERALA

May 2024

DEPT. OF COMPUTER SCIENCE & ENGINEERING

SREE CHITRA THIRUNAL COLLEGE OF ENGINEERING TRIVANDRUM

2023- 24

CERTIFICATE

This is to certify that the report entitled **Medical Prognosis with DenseNet and ResNet** submitted by **Abhimanyu Pradeep(SCT21AM004), Abineet Thampi(SCT21AM007), Harrel Jacob Alex(SCT21AM031)** to the APJ Abdul Kalam Technological University in partial fulfillment of the B.Tech. degree in Computer Science and Engineering (Artificial Intelligence and Machine Learning) is a bonafide record of the mini-project work carried out by him under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Prof. Linita Ann Koruthu
(Project Guide)
Assistant Professor
Dept.of CSE
SCT College of Engineering
Trivandrum

Prof. Binu Rajan M R
(Project Coordinator)
Assistant Professor
Dept.of CSE
SCT College of Engineering
Trivandrum

Prof. Rejimol Robinson R R
Associate Professor and Head
Dept. of CSE
SCT College of Engineering
Trivandrum

DECLARATION

I hereby declare that the mini-project report Medical Prognosis using DenseNet and ResNet, submitted for partial fulfillment of the requirements for the award of degree of Bachelor Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Prof. Linita Ann Koruthu.

This submission represents our ideas in our own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources.

I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Abhimanyu Pradeep
Abineet Thampi
Harrel Jacob Alex

TRIVANDRUM
09-05-2024

Acknowledgment

I take this opportunity to express our deepest sense of gratitude and sincere thanks to everyone who helped us to complete this work successfully. I express my sincere thanks to Prof. Rejimol Robinson R R, Head of Department, Computer Science & Engineering, Sree Chitra Thirunal College of Engineering for providing us with all the necessary facilities and support. I gratefully acknowledge the contributions of Prof. Binu Rajan M R, our project coordinator, whose support and assistance were instrumental in the completion of this proposed work.

I would like to place on record our sincere gratitude to our project guide, Prof. Linita Ann Koruthu, Assistant Professor, Computer Science & Engineering, Sree Chitra Thirunal College of Engineering for the guidance and mentorship throughout this work.

Abhimanyu Pradeep
Abineet Thampi
Harrel Jacob Alex

Abstract

The use of artificial intelligence and machine learning has further progressed advancements in modern society. The applications of A.I are far ranging and have almost infinite potential. It is growing to be a major game-changer in the field of medical science. Massive amounts of data which is now widely available has made the use of computational networks more viable than ever now.

Applications of artificial intelligence in the medical field encompass tasks like patient diagnosis, enhancing communication between healthcare providers and patients, and the transcription of medical documents such as prescriptions. Medical image analysis is another avenue which is made possible through deep learning machines. While computers excel in executing tasks efficiently, modern algorithms have now reached accuracies comparable to human experts in various medical disciplines. Medical images and scan reports may often be difficult to understand for the common populace.

Our proposed work aims to solve that problem by using a Convolutional Neural Network(CNN) to analyze various x-ray images and MRI scans and extract information. The analysis is then used to provide a meaningful prognosis for the health complication(if it exists) which is inferred from the data. The additional information can be used for further diagnosis and for patients to better understand their own conditions at a first glance

Contents

Acknowledgment ..... **i**
Abstract ..... **ii**
1 Introduction **1**
1.1 Motivation 1
1.2 Purpose of the project 2
1.3 Deep Learning Frameworks 2
1.4 Transfer learning 3
1.5 Intended Audience and Document Overview 3
1.6 Scope 4
1.7 Limitations 5
2 Literature Review **6**
2.1 Computer-aided detection (CADe) and diagnosis (CADx) system for lung cancer with likelihood of malignancy 6
2.2 Deep Learning Applications in Medical Image Analysis 6
2.3 Label-Efficient Deep Learning in Medical Image Analysis: Challenges and Future Directions 7
2.4 Explainable Artificial Intelligence (XAI) in Deep Learning-based Medical Image Analysis 7
2.5 Problem Statement 8
2.6 Proposed Solution 9
3 Methodology **10**
3.1 Data Acquisition: 10
3.2 Data Preprocessing: 11
3.3 Selection of a Pre-trained Model and Training: 12
3.4 Model Architecture: 14
3.5 Data Splitting: 14
3.6 Transfer Learning Training: 14
3.7 Evaluation: 15
4 Implementation **16**
4.1 Development Tools and Libraries 16
4.1.1 TensorFlow 16
4.1.2 Pytorch 17
4.1.3 Streamlit 18
4.1.4 Numpy 19

4.1.5 Matplotlib....20
4.2 Development Process....21
4.2.1 Data Collection and Preprocessing....21
4.2.2 Implementation of DenseNet for Pneumonia detection....24
4.2.2 Implementation of ResNet for Brain Tumor detection....26
4.2.1 Integration with User Interface:....27
5 Results & Evaluation....29
Result: 1....29
Result: 2....30
Result: 3....31
Result: 4....32
Evaluation....33
Pneumonia Detection (DenseNet121) Model....33
Brain Tumor Detection (ResNet50) Model....34
6 Conclusion....35
References....36

Chapter- 1

Introduction

In today's world of rapidly evolving healthcare technologies, the need for accurate, efficient, and swift medical diagnoses is paramount. Traditional methods of analyzing medical images often rely on manual interpretation by trained professionals. While valuable, this approach can be time-consuming and susceptible to human error. This project aims to address these limitations by creating a sophisticated Artificial Intelligence (AI) system.

This AI system is designed to analyze a variety of medical imaging modalities, encompassing X-rays, MRI scans, and potentially other formats. By leveraging deep learning algorithms, the system will be able to process these images and extract crucial information. This information will then be used to provide a meaningful prognosis – essentially an educated prediction – about any potential health complications identified in the data.

The ultimate goal of this project is to create a valuable tool that can assist healthcare professionals in the diagnostic process. By automating a portion of the image analysis workload and potentially reducing the risk of human error, this AI system has the potential to streamline diagnoses, improve efficiency, and ultimately contribute to better patient outcomes.

1.1 Motivation

Early detection of pneumonia and brain tumors is critical for successful treatment and improved patient outcomes. However, traditional methods of analyzing medical images, like chest X-rays and brain scans, can be time-consuming and susceptible to human error.

The project will leverage deep learning techniques, a powerful subset of AI. By training AI models on large datasets of labeled medical images, we can equip them to recognize the specific features associated with these diseases in X-rays and scans.

Overall, the project is motivated by the potential of AI to revolutionize medical image analysis by making it faster, more accurate, and more consistent, leading to better healthcare outcomes for patients.

1.2 Purpose of the project

Medical images and scan reports may often be difficult to understand for the common populace. This project aims to solve that problem by using a Convolutional Neural Network(CNN) to analyze various x-ray images and MRI scans and extract information. The analysis is then used to provide a meaningful prognosis(if there is one) for the health complication which is inferred from the data. The additional information can be used for further diagnosis and for patients to better understand their own conditions at a first glance.

1.3 Deep Learning Frameworks

This mini-project relies on deep learning, a subfield of machine learning, specifically using convolutional neural networks like DenseNet and ResNet. CNNs are powerful for image analysis tasks. They can be trained on large datasets of medical images (X-rays and CT scans) labeled with the presence or absence of pneumonia or brain tumors.

The mini-project utilizes deep learning frameworks like TensorFlow and PyTorch for model development and training. These frameworks provide tools and libraries specifically designed for building, training, and deploying deep learning models. Features of deep learning include:

- **1. Automatic Feature Extraction:** Unlike traditional machine learning methods that require manual feature engineering, deep learning models can automatically learn relevant features from the data itself. This is particularly beneficial for complex data like medical images where handcrafted features might be challenging to define.
- **2. Representation Learning:** Deep learning architectures like DenseNet and ResNet are adept at learning hierarchical representations of data. Lower layers capture basic features like edges and textures, while higher layers combine these features to form more complex representations relevant to the task
- **3. Non-Linearity:** Deep learning models often employ non-linear activation functions in their layers. This allows them to model complex relationships between input data and output predictions, crucial for capturing the intricacies of medical images.

1.4 Transfer learning

Transfer learning is a powerful technique in deep learning that allows us to leverage knowledge gained from one task to improve performance on a related but different task. Imagine training a massive image recognition model on millions of general images (cats, dogs, cars). This model learns powerful features for recognizing objects and patterns in images. Transfer learning lets us take this pre-trained model and "fine-tune" it for a new task, like medical image analysis. By keeping the core architecture and learned features, we can train the model much faster and potentially achieve better accuracy on the new task compared to starting from scratch. This approach saves time, resources, and ultimately benefits the development of specialized AI models for various applications.

1.5 Intended Audience and Document Overview

This app can be used by patients and the general public, as they could upload medical scans, and the system could provide insights or connect them with relevant medical resources.

Early and accurate diagnoses can lead to:

- Earlier treatment initiation.

- Improved treatment outcomes and potentially higher survival rates.
- Reduced risk of complications.

This proposed work is also tailored for radiologists, pulmonologists, oncologists, and other specialists who routinely analyze medical images for diagnosis. It also aims to provide them with an AI tool that can assist in analyzing chest X-rays and brain scans, potentially leading to:

- Faster image analysis and interpretation.
- Improved accuracy in detecting pneumonia and brain tumors.
- Increased confidence in diagnoses.

1.6 Scope

Focus: This mini-project focuses on developing deep learning models for medical image analysis. It utilizes a DenseNet121 architecture from TensorFlow trained on lung X-rays to detect pneumonia. A separate ResNet model from PyTorch, trained on brain CT scans, is used for brain tumor classification. While the system doesn't provide medical diagnosis, it can analyze uploaded scans and potentially offer insights or connect users with relevant medical resources. The proposed work highlights the potential of deep learning for medical image analysis tasks.

Features:

- Image Analysis:
    - Analyze lung X-rays for pneumonia using a DenseNet model trained with TensorFlow.
    - Analyze brain CT scans for brain tumors using a ResNet model trained with PyTorch.
- User Interface: Allow users to upload medical scans (X-rays and CT scans).
- Output:
    - Provide insights into the uploaded image, potentially indicating the likelihood of pneumonia or brain tumor presence.
    - Connect users with relevant medical resources (optional).

- Expanding functionality to handle multiple images or integrate with a larger system.
- Implementing data pre-processing pipelines within the program.
- Incorporating metrics to evaluate model performance and assess prediction reliability.

Deliverables:

- Trained DenseNet and ResNet models.
- Functional user interface for image upload and analysis results.
- Documentation of the development process, data sources, and evaluation methods.

1.7 Limitations

- **Data Quality and Bias**: The accuracy of the models heavily relies on the quality and representativeness of the training data. Biases in the data, such as an overrepresentation of certain types of pneumonia or brain tumors, can lead to biased predictions. Also Models trained on specific datasets might not perform well on unseen data with different characteristics. Expanding and diversifying the datasets can improve generalizability.
- **Black Box Nature:** Deep learning models can be complex and difficult to interpret. Understanding how a model arrives at a specific prediction can be challenging, limiting explainability and trust in the system's reasoning. The models might incorrectly identify pneumonia or brain tumors (false positives) or miss actual cases (false negatives). These errors can have significant consequences for patients.
- **User Dependence:** The system relies on users uploading high-quality medical scans for accurate analysis. Issues like poor image resolution or artifacts can affect the model's performance

Chapter- 2

Literature Review

2.1 Computer-aided detection (CADe) and diagnosis (CADx) system for lung cancer with likelihood of malignancy

Computer programs that are designed to assist radiologists in the detection and diagnosis of lung cancer. CADe and CADx systems for the detection and diagnosis of lung cancer have been important areas of research in recent decades. CADe systems focus on finding suspicious lesions in lung images, such as nodules on chest X-rays or CT scans. CADx systems go a step further by not only detecting suspicious lesions but also trying to determine whether they are likely to be benign (harmless) or malignant (cancerous). CADe systems do not present the radiological characteristics of tumors, and CADx systems do not detect nodules and do not have good levels of automation.A rule-based classifier and Support Vector Machine (SVM) have been used to eliminate false positives.

2.2 Deep Learning Applications in Medical Image Analysis

There are different types of medical imaging and tens of millions of examinations have been conducted worldwide. Both 2-D and 3-D structures of organs have been studied to differentiate and understand what is normal and abnormal. CNNs are well-suited to perform such image recognition tasks. It is the most popular ML algorithm as it preserves local image relations, while performing dimensionality reduction.Some of the prominent supervised learning algorithms used are Convolutional Neural Networks(CNNs), Recurrent Neural Networks(RNNs), Transfer Learning with CNNs. Unsupervised learning algorithms used include Autoencoders, Restricted Boltzmann Machines and Deep Belief Networks, Generative Adversarial Networks

2.3 Label-Efficient Deep Learning in Medical Image Analysis: Challenges and Future Directions

The research provides a comprehensive overview of label-efficient learning methods in the context of medical image analysis (MIA). It covers various approaches, including semi-supervised, self-supervised, multi-instance, active, and annotation-efficient learning strategies. The survey examines over 300 recent papers and categorizes the methods into different schemes, such as generative data augmentation, human-in-the-loop interaction, and omni-supervised learning.

It also addresses the challenges of generalization across domains and datasets, the need for benchmarking, and the application of expert knowledge to refine model outputs. The document highlights the importance of foundation models and discusses the potential future directions for label-efficient learning in MIA. Additionally, it emphasizes the significance of hybrid methods and the use of diverse learning methodologies in the field.

The paper also provides a taxonomy based on learning schemes and offers guidance for future research in label-efficient learning. Overall, the document aims to shed light on the progress and challenges in label-efficient learning in MIA and provide a roadmap for future advancements in the field.

2.4 Explainable Artificial Intelligence (XAI) in Deep Learning-based Medical Image Analysis

The paper highlights the importance of human-grounded evaluation in XAI, which involves simpler human experiments to assess the quality of explanations. It also discusses the increasing significance of XAI in high-stakes decision-making, particularly in the medical field. The survey identifies trends in XAI research, such as the combination of multiple forms of explanation, and provides examples of holistic approaches in medical image analysis.

Furthermore, the research delves into specific XAI techniques used in medical imaging, such as textual explanation through image captioning and the use of human-generated sentences as ground truth for training. It also discusses the challenges and critiques of XAI, including the potential lack of faithfulness and detail in explanations provided by black box models.

In conclusion, the research provides a comprehensive overview of the use of XAI in deep learning-based medical image analysis, discussing evaluation methods, current trends, and future perspectives for XAI in this field. It surveys 223 papers and categorizes them according to an XAI framework, providing a valuable resource for understanding and applying XAI techniques in medical imaging.

2.5 Problem Statement

Early and accurate diagnosis of brain tumors and pneumonia is crucial for improving patient outcomes. However, traditional methods of diagnosis, such as visual analysis of medical scans by radiologists, can be time-consuming, subjective, and susceptible to human error. This can lead to delayed diagnosis and treatment, potentially impacting patient survival rates and quality of life. The increasing availability of medical image data (CT scans, X-rays) creates an opportunity to utilize advanced technologies for more efficient and objective analysis. However, manually analyzing large amounts of medical images is impractical.

.

2.6 Proposed Solution

This proposed work addresses these challenges by developing a deep learning-based medical image analysis application. This application aims to:

- Automate the analysis of brain scans and chest X-rays to detect brain tumors and pneumonia.
- Assist radiologists by providing objective and potentially faster insights into the presence or absence of these conditions.
- Offer a user-friendly interface for healthcare professionals to interact with the system and receive predictions.

By leveraging deep learning and artificial intelligence, this mini-project has the potential to improve the accuracy and efficiency of medical image analysis, ultimately contributing to better patient care.

Chapter- 3

Methodology

In this chapter, we elaborate on the techniques used in the creation of the medical prognosis tool. An in-depth explanation of each part of the process is provided for gaining a deeper insight into the methodology adopted here.

3.1 Data Acquisition:

There are multiple data sources available for the sourcing of data. Medical institutions such as hospitals, clinics, and other healthcare facilities can be a source of medical images like X-rays, CT scans, or MRIs.

Public datasets, which are publicly available medical image datasets exist, often curated by research institutions or organizations. These datasets can be a valuable resource for training and evaluating the model. It includes:

- MNIST: Handwritten digit images
- CIFAR-10/100: Small image classification datasets with diverse classes (not medical, but useful for benchmarking)
- Kaggle Datasets: Many medical image classification challenges are hosted on Kaggle, providing datasets and evaluation metrics.

3.2 Data Preprocessing:

Pre-defined Transformations

Using pre-defined transformations allows us to combine multiple transformations into a single pipeline, making the pre-processing code more concise and easier to manage. We can define different transformation pipelines for training and testing and reuse them throughout the code. By using pre-defined transformations, we ensure consistent data pre-processing for both training and testing phases, which is important for reliable model evaluation. Pre-defined transformations offer more flexibility through different functionalities

Transformation Pipeline

This pipeline consists of the following steps applied sequentially to the image loaded from the provided path:

- Resize: Resizes the image to the size required.
- CenterCrop: Takes a central square crop of required size from the resized image. This ensures the model receives a fixed-size input.
- ToTensor(): Converts the PIL image format to a PyTorch tensor.
- Normalize: Normalizes the pixel values of the tensor based on the provided mean and standard deviation. This is a common practice in image classification tasks to improve model performance.
- Test Function: The testing transformations follow a similar structure but with some key differences

Custom Preprocessing

Implements a simpler pre-processing approach. Such as, resizing the image directly to the target size using the resize method from PIL. Converting the PIL image to a NumPy array. Normalizing pixel values by dividing each color channel by 255 (assuming the image has values between 0 and 255). This is a simpler normalization approach. Expanding dimension (batch size) by adding a new dimension of size 1 at the beginning of the array to represent a batch of one image.
Custom pre-processing provides more control over specific normalization methods used.

3.3 Selection of a Pre-trained Model and Training:

Our mini-project utilized pre-trained models for both brain tumor and pneumonia detection, capitalizing on the advantages of transfer learning:

- **Reduced Training Time:** Pre-trained models have already learned low-level features from massive datasets, significantly reducing the training time required compared to building a model from scratch.
- **Improved Performance:** By leveraging the pre-trained knowledge, these models can often achieve better performance on smaller datasets specific to our tasks (brain tumor and pneumonia classification).

Training Process

- **Data Preparation:**
    - **Data Acquisition:** Medical image datasets for brain tumors and pneumonia were collected (hospitals, public repositories).
    - **Data Labeling:** Medical experts labeled the images for presence/absence of tumors (different types for brain tumors) and pneumonia.
    - **Preprocessing:** Libraries have been used for:
        - Random cropping, flipping, resizing (data augmentation)
        - Normalization

- **Model Selection and Fine-tuning:**
    - **Brain Tumor Detection (PyTorch):**
        - A pre-trained ResNet50 model is loaded using PyTorch's functionalities.
        - The model is split into two parts:
            - **Frozen Layers:** The initial layers (responsible for learning low-level features) were frozen to leverage their pre-trained knowledge.
            - **Fine-tunable Layers:** The final layers (responsible for high-level features specific to the task) were set to trainable mode.
        - The prepared brain tumor dataset was split into training, validation, and testing sets (e.g., 80%/10%/10%).
        - An optimizer (e.g., Adam) and a loss function (e.g., cross-entropy) were chosen.
        - During training:
            - Training data was fed through the frozen layers of the pre-trained model.
            - The fine-tunable layers learned to adapt the pre-trained features for brain tumor classification based on the training data.
            - The model's predictions on the validation set were monitored to prevent overfitting (e.g., using early stopping).
        - Training stopped when the validation performance plateaued or started to decline.
    - **Pneumonia Detection (Keras/TensorFlow):**
            - A pre-trained model on chest X-ray datasets was loaded using Keras.
            - The model has been fine-tuned on a pneumonia-specific dataset using a similar approach as brain tumor detection (splitting data, freezing layers, training with optimizer and loss function).

3.4 Model Architecture:

- Import pre-trained model: Load the pre-trained model architecture, excluding the final classification layers.
- Add weight to ResNet according to different scans. Weight obtained from image net.
- Freeze pre-trained layers: Set the weights (parameters) of the pre-trained layers to non-trainable (frozen) to prevent them from changing during training. These layers act as the feature extractor.
- Add new layers: Design and add new fully-connected layers specific to your classification task (e.g., disease vs. healthy tissue).

3.5 Data Splitting:

- Divide your preprocessed medical image data into training, validation, and testing sets.
- Training Set (Largest): This subset (typically 60-80% of the data) is used to train the model. The model learns patterns and relationships between image features and their corresponding labels (presence/absence of tumors or pneumonia) from the training data.
- Validation Set (Medium): This subset (typically 10-20% of the data) is used to monitor the model's performance during training. It helps prevent overfitting, where the model performs well on the training data but poorly on unseen data. The model's performance on the validation set is used to adjust hyperparameters (learning rate, optimizer settings) during training.
- Testing Set (Smallest): This unseen subset (typically 10-20% of the data) is used to evaluate the final model's generalizability after training is complete. The model's performance on the testing set provides an unbiased estimate of how well it performs on new data it hasn't encountered before.

3.6 Transfer Learning Training:

- Train the modified CNN model on the training set.

- The model propagates the input image through the frozen pre-trained layers (feature extractor).
- New features specific to the task are extracted in the newly added layers.
- The final layers perform classification based on the learned features.
- Use the validation set to monitor training progress and adjust hyperparameters (learning rate, optimizer) if needed to prevent overfitting.

3.7 Evaluation:

This system analyzes medical scans uploaded by users and offers insights or connects them to relevant resources. Here's how we can evaluate such a model trained with Convolutional Neural Networks (CNNs):

Performance Metrics:

- **Accuracy:** This is a core metric, reflecting how often the model makes correct classifications or predictions on unseen data. Common metrics include:
    - Sensitivity (True Positive Rate): How often the model correctly identifies a positive case (e.g., disease presence).
    - Specificity (True Negative Rate): How often the model correctly identifies a negative case (e.g., disease absence).

Chapter- 4

Implementation

4.1 Development Tools and Libraries

4.1.1 TensorFlow

TensorFlow is an open-source software library developed by Google for numerical computation and large-scale machine learning. It's a powerful tool that allows you to build and train various machine learning models, especially deep learning models.

Provides the foundation for building and deploying machine learning models. It offers tools for data loading, preprocessing, model building, training, and evaluation.

Here are some key aspects of TensorFlow:

- **High-level API (Keras):** Includes Keras, a high-level API that simplifies the process of building and training models. Keras provides pre-built building blocks for common neural network layers and activation functions, allowing you to focus on the model's architecture and logic.
- **Open-source:** Freely available for anyone to use and contribute to, fostering a large and active developer community.
- **Widely used:** A popular choice for machine learning projects due to its extensive capabilities and large user base. This translates to plenty of resources, tutorials, and community support available online.

- **Scalability:** Handles small-scale projects on personal computers to large-scale deployments on powerful computing clusters.
- **Community and Resources:** Benefits from a vast developer community that contributes to its ongoing development and provides a wealth of learning resources and tutorials.

4.1.2 Pytorch

PyTorch is a machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, originally developed by Meta AI and now part of the Linux Foundation umbrella.
Functionality of Pytorch includes**:**

- **Dynamic Computation Graph:** Unlike TensorFlow's static computational graph, PyTorch utilizes a dynamic approach. This means the model's computational graph is built and modified during runtime, offering greater flexibility for research and experimentation.
- **Pythonic Interface:** PyTorch leverages Python syntax, making it familiar and easy to learn for programmers already comfortable with Python. This can streamline development and prototyping compared to frameworks with a steeper learning curve.
- **Rich Ecosystem:** PyTorch has a growing ecosystem of libraries and extensions that support various tasks like computer vision, natural language processing (NLP), and scientific computing. This expands its capabilities and simplifies development for specific applications.

Here’s how Pytorch is utilized :

- **Research and Prototyping:** PyTorch's dynamic nature and ease of use make it a popular choice for researchers and developers who want to experiment with new deep learning architectures and ideas.

- **Computer Vision:** Tasks like image classification, object detection, and image segmentation are well-supported by PyTorch and its ecosystem of libraries.
- **Natural Language Processing:** Analyzing text data, machine translation, and sentiment analysis are common applications for PyTorch in NLP.
- **Generative Models:** Creating new data like images or text can be achieved using generative models built with PyTorch.

4.1.3 Streamlit

Streamlit is a Python library used for building interactive web applications for machine learning and data science. It is used here to create the user interface for the vector-based search engine, allowing users to interact with the system through a web browser.
Streamlit plays a crucial role in the development of the Medical Prognosis with DenseNet and ResNet by enabling the creation of an interactive and user-friendly interface. Its simplicity, versatility, and seamless integration with Python libraries make it an ideal choice for building web applications for machine learning and data science tasks.
Here's how Streamlit is leveraged :

- **Interactive Web Applications**: Streamlit enables the creation of interactive web applications directly from Python scripts, without the need for HTML, CSS, or JavaScript. This simplifies the development process and allows for rapid prototyping of user interfaces for machine learning and data science applications.
- **User Interface Development**: Streamlit provides a simple and intuitive API for building user interfaces using familiar Python syntax. Developers can easily create various UI components such as buttons, sliders, text inputs, and plots to facilitate user interaction with the search engine.
- **Real-time Updates**: Streamlit offers automatic reactive updates, allowing the user interface to dynamically update in response to user inputs or changes in the underlying data. This enables real time exploration and visualization of search results, enhancing the user experience and providing immediate feedback.

- **Integration with Data Processing**: Streamlit seamlessly integrates with data processing libraries such as Pandas, allowing developers to incorporate data analysis and visualization directly into the user interface. This facilitates data exploration and interpretation within the search engine interface, empowering users to gain insights from the search results.

- **Deployment**: Streamlit simplifies the deployment of web applications, providing built-in support for deploying applications to various platforms such as Streamlit Sharing, Heroku, or Docker containers. This ensures that the vector-based search engine can be easily deployed and accessed by users via a web browser, without the need for complex setup or configuration.

4.1.4 Numpy

NumPy is a fundamental package for scientific computing in Python. It is used for handling numerical operations efficiently, particularly during the vectorization and indexing processes. NumPy's role here is:

- **Data Loading and Preprocessing**: NumPy provides efficient tools for loading various data formats, potentially including medical image datasets in formats like DICOM (Digital Imaging and Communications in Medicine). While specialized medical image processing libraries might be used for advanced tasks, NumPy offers a foundation for basic loading functionalities.

- **Vectorization**: NumPy's vectorized operations enable the application of mathematical operations on entire arrays of data without the need for explicit looping. This significantly improves the efficiency and speed of computations.

- **Indexing and Data Manipulation**: NumPy's array indexing capabilities are crucial for accessing and manipulating data stored in arrays. It provides intuitive syntax for slicing, indexing, and reshaping arrays, allowing for seamless data manipulation tasks such as filtering documents, extracting embeddings, or performing similarity calculations.

- **Performance Optimization**: NumPy's underlying implementation is highly optimized and written in C, ensuring fast execution of numerical operations. This performance optimization is essential for maintaining the responsiveness and scalability of the search engine, particularly when handling large-scale datasets and complex computations.

- **Normalization:** NumPy allows for normalization of image pixel values to a specific range (e.g., 0-1 or -1 to 1) to ensure consistent data representation for the machine learning models. This can improve the training process and potentially lead to better model performance.

4.1.5 Matplotlib

While Matplotlib isn't directly involved in the core machine learning model training, it plays a crucial role in understanding the data, evaluating model performance, and communicating the project's findings in this medical image analysis mini-project focusing on pneumonia detection using X-rays.
Here's how matplotlib is utilized:

1. Data Exploration and Understanding:

- **Visualizing Sample Images:** At the outset, Matplotlib allows you to visualize individual X-ray images. By creating grayscale or heatmap representations, you can gain insights into the data distribution, identify variations in lung opacity associated with pneumonia, and explore potential patterns that might inform the model development process.

2. Preprocessing Insights (Optional):

- **Distribution of Pixel Intensities:** Matplotlib can help visualize the distribution of pixel intensities within the X-ray images. This can be crucial for data normalization techniques, where you might choose to normalize pixel values based on the observed distribution to ensure consistent data representation for the models.

3. Model Performance Evaluation:

- **Confusion Matrix:** After training the DenseNet and ResNet models, Matplotlib is instrumental in creating confusion matrices. These visualizations depict how many X-ray images were correctly classified (True Positives: pneumonia identified correctly, True Negatives: healthy lungs identified correctly) and how many were misclassified (False Positives: healthy lungs identified as pneumonia, False Negatives: pneumonia missed by the model). Analyzing the confusion matrix helps identify potential biases or weaknesses in the models' performance.

- **ROC Curves:** Matplotlib can be used to generate Receiver Operating Characteristic (ROC) curves. These curves plot the true positive rate (TPR) against the false positive rate (FPR) for various classification thresholds. By analyzing the ROC curve, you can assess the trade-off between sensitivity (correctly identifying pneumonia) and specificity (correctly identifying healthy lungs).

- **Loss Curves:** Matplotlib can visualize the training and validation loss curves over training epochs. These plots depict how the model's loss (difference between prediction and true label) changes during training. Analyzing the loss curves helps identify issues like overfitting (model memorizes training data but performs poorly on unseen data) or underfitting (model fails to learn the underlying patterns).

4.2 Development Process

4.2.1 Data Collection and Preprocessing

This mini-project tackles two distinct medical image analysis tasks: pneumonia detection in lung X-rays and brain tumor detection in CT scans. Due to the inherent differences between the data modalities (X-ray vs. CT scan) and the chosen deep learning frameworks (TensorFlow for DenseNet and PyTorch for ResNet), separate data collection and preprocessing strategies are necessary.

Data Collection:

- **Public Datasets:** While leveraging publicly available datasets is an initial approach, it's crucial to acknowledge the limitations. Here are some considerations:
    - **Data Quality and Bias:** Public datasets might have inconsistencies in labeling, image quality variations, or biases that don't reflect real-world patient populations.
    - **Data Anonymization:** Ensure the datasets comply with Health Insurance Portability and Accountability Act (HIPAA) regulations to protect patient privacy.
    - **Dataset Options:** Here are some potential sources, keeping in mind the aforementioned limitations:
        - **Pneumonia:** ChestX-ray8, NIH Chest X-ray dataset, Pneumonia Detection Challenge (Kaggle)
        - **Brain Tumor:** BraTS (Brain Tumor Segmentation) dataset, ISLES (Ischemic Stroke Lesion Segmentation) dataset
- **Collaboration with Medical Institutions (Preferred):** Partnering with hospitals or imaging centers offers several advantages:
    - **Access to High-Quality Labeled Data:** Medical professionals can provide high-quality, labeled data specific to the local patient population and disease patterns, leading to potentially better model performance.
    - **Data Anonymization Expertise:** Medical institutions have established procedures for anonymizing patient data while preserving its diagnostic value.

Data Preprocessing:

Preprocessing for Lung X-ray (Pneumonia Detection):

1. **Data Formatting:** Ensure all X-ray images are in a consistent format (e.g., JPEG, PNG) and have the same bit depth (usually 8-bit grayscale).

2. **Normalization:** Normalize pixel intensity values to a specific range (e.g., 0-1 or -1 to 1) for consistent data representation during model training. Libraries like NumPy can be used for this task.

3. **Resizing:** Resize all X-rays to a standard size appropriate for the DenseNet model's input layer. This reduces computational complexity and ensures all images are processed consistently.

4. **Data Augmentation (Optional):** Consider using TensorFlow's *ImageDataGenerator* to create variations of existing X-rays (flipping, rotation, zoom). This helps address limited data issues and improves model generalizability.

5. **Data Splitting:** Divide the dataset into training, validation, and testing sets (common split: 80%/10%/10%). The training set is used to train the DenseNet model, the validation set monitors training progress and prevents overfitting, and the testing set evaluates the model's performance on unseen data.

Preprocessing for CT Scan (Brain Tumor Detection):

1. **Data Formatting:** Ensure all CT scans are in a standardized format (e.g., DICOM) commonly used for medical imaging. Specialized libraries like Pydicom might be needed for handling DICOM files.

2. **Normalization (Intensity or HU):** CT scans have varying intensity scales depending on the acquisition protocol. Normalization techniques can involve either intensity normalization (similar to X-rays) or Hounsfield Unit (HU) standardization, which considers tissue density information within the CT scan.

3. **Resizing or Cropping:** CT scans can be large 3D volumes. Depending on the ResNet model's architecture, you might need to resize the entire scan or crop specific regions of interest containing the brain.

4. **Data Augmentation (Optional):** Similar to X-rays, data augmentation can be beneficial, but with considerations for 3D data manipulations within the *ImageDataGenerator* or other specialized libraries.

5. **Data Splitting:** Split the CT scan dataset into training, validation, and testing sets, similar to the X-ray data.

4.2.2 Implementation of DenseNet for Pneumonia detection

DenseNet is a convolutional neural network (CNN) architecture known for its feature reuse and densely connected layers. This design improves feature propagation and potentially reduces the number of parameters compared to traditional CNNs.
TensorFlow provides pre-trained DenseNet models like DenseNet121 or DenseNet201. This mini-project used DenseNet121 for detection of pneumonia

1. Model Selection and Pre-trained Weights:

- **DenseNet121:** The proposed work utilizes DenseNet121, a convolutional neural network (CNN) architecture known for its depth and efficiency. DenseNet's dense connectivity pattern allows feature reuse and improves gradient flow during training.

- **Pre-trained Weights:** Consider using pre-trained DenseNet121 weights on ImageNet (a large image dataset). These weights provide a strong foundation for image recognition tasks and can be fine-tuned for pneumonia detection on the lung X-ray dataset.

2. Model Architecture:

- **Base Model:** Load the pre-trained DenseNet121 model from TensorFlow using *tf.keras.applications.DenseNet121*. By default, the pre-trained weights are loaded, but the model is set to non-trainable (freezing the weights).

- **Compile the Model:** The optimizer selected is Adam with learning rate 0.0001, binary cross-entropy for loss function , and metrics (e.g., accuracy) during model compilation.

3. Model Training:

Training Process: Train the model using the training set. Given 20 epochs, during each training epoch, the model:

- Forwards batches of X-ray images through the network.
- Calculates the loss (difference between prediction and true label) based on the output of the final layer.
- Backpropagates the loss to update the weights of the newly added layers (fine-tuning) while keeping the pre-trained weights frozen.
- Monitors training progress and performance metrics (accuracy, loss) on the validation set to prevent overfitting.

4. Model Evaluation:

- **Testing Set:** Once training is complete, evaluate the model's performance on the unseen testing set, val set(10-20% of the original data). Metrics of accuracy is taken in this model.

6. Saving the Model:

- Save the trained model for future use in making predictions on new lung X-rays using *model.save()* function.

4.2.2 Implementation of ResNet for Brain Tumor detection

ResNet (Residual Network) is a popular deep learning architecture known for its effectiveness in handling deep neural networks. PyTorch offers a convenient way to leverage pre-trained ResNet models for various image classification tasks, including medical image analysis.

1. Model Selection and Pre-trained Weights:

- **ResNet:** The proposed work leverages a ResNet architecture, a popular CNN known for its ability to handle deep networks effectively by introducing skip connections that facilitate gradient flow during training. We use ResNet50 in this case

- **Pre-trained Weights:** I pretrained ResNet weight *IMAGENET1K_V1*. These weights provide a strong foundation for image recognition and can be fine-tuned for brain tumor detection on the CT scan dataset. PyTorch offers functionalities to load pre-trained models.

2. Model Architecture:

Base Model: Load the pre-trained ResNet model from PyTorch using appropriate functions (e.g.,*torchvision.models*). By default, the pre-trained weights are loaded, but the model is set to non-trainable (freezing the weights)

Compile the Model: The optimizer selected is Stochastic Gradient Descent (SGD) optimizer, CrossEntropyLoss for loss function , and metrics (e.g., accuracy) during model compilation.

3. Model Training:

Training Process: Train the model using the training set. Given 50 epochs, during each training epoch, the model:

- Forwards batches of X-ray images through the network.
- Calculates the loss (difference between prediction and true label) based on the output of the final layer.
- Backpropagates the loss to update the weights of the newly added layers (fine-tuning) while keeping the pre-trained weights frozen.
- Monitors training progress and performance metrics (accuracy, loss) on the validation set to prevent overfitting.

4. Model Evaluation:

- **Testing Set:** Once training is complete, evaluate the model's performance on the unseen testing set, val set(10-20% of the original data). Metrics of accuracy is taken in this model

6. Saving the Model:

- Save the trained model for future use in making predictions on new lung X-rays using *torch.save()* function

4.2.1 Integration with User Interface:

Once the fine-tuned ResNet model is trained and saved, it can be integrated with a user interface that allows users to upload CT scans. In this case we use streamlit library to create a UI where users could upload medical scans, and the system could provide insights or connect them with relevant medical resources generated from genai.

Streamlit UI allows users to upload medical images for analysis. It offers two functionalities:

- **Pneumonia Detection:** Users can upload lung X-rays. Clicking the "Check for pneumonia" button triggers analysis using a pre-trained Keras model (*pmod.keras*). The predicted outcome (likely pneumonia or normal) is displayed.

- **Brain Tumor Classification:** Uploading a brain CT scan allows users to click the "Check for brain tumor" button. A pre-trained PyTorch ResNet50 model (*tumormod.pth*) loaded on the GPU (if available) analyzes the image. The predicted outcome (likely brain tumor or normal) is displayed.

Additional Features:

- **About Section:** An expandable section provides information about the app's purpose and the types of scans it can handle (CT brain, X-ray lung).

- **Result Description:** If a diagnosis is predicted, the app displays it prominently and calls a function (*getres*) from the main module to retrieve a descriptive markdown text about the condition. This text is then formatted and displayed using Streamlit's markdown functionality.

- **Error Handling:** Uploading an unsupported image type or encountering errors during analysis results in an appropriate message being displayed.

Chapter- 5

Results & Evaluation

Now I present the results or outcomes of the implemented medical image analysis model in predicting the disease and providing accurate reports on the disease predicted. By uploading a series of x-rays and ct scans we can analyze the accuracy of the model and its uses. Each result is accompanied by analysis.

Result: 1

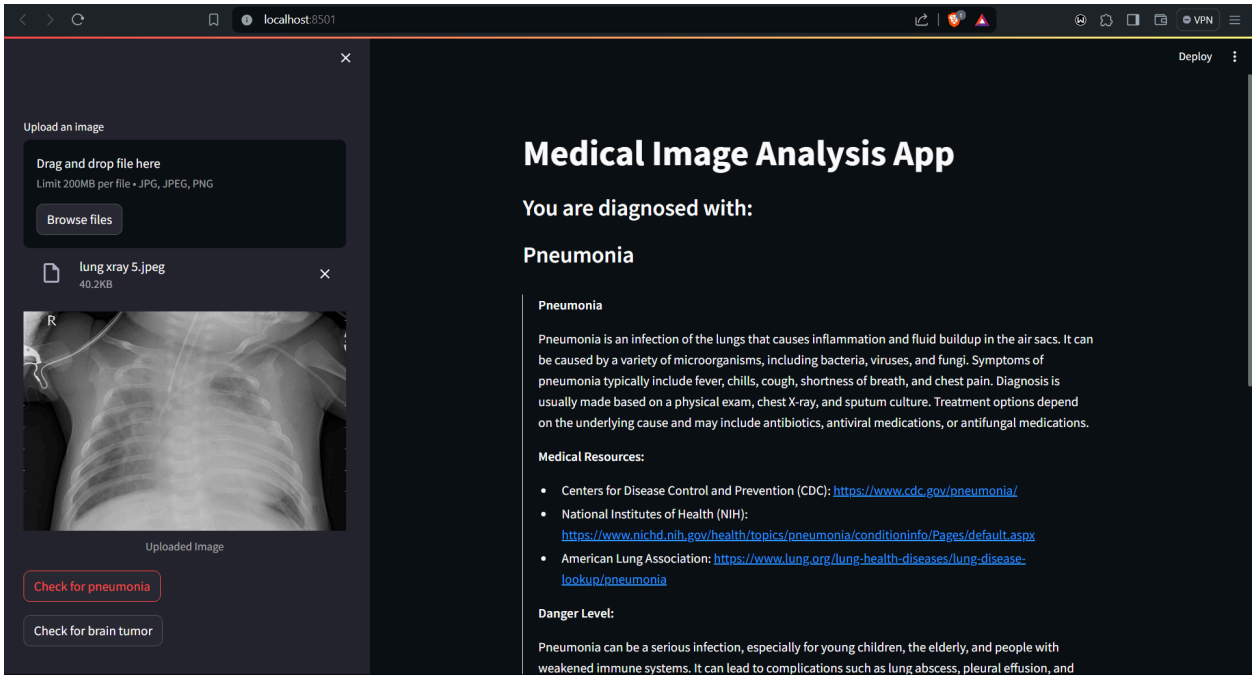


Figure 5.1: Example using X-ray of a pneumonia affected lungs

Here's a breakdown of the result:

- An X-ray confirmed Pneumonia of lungs is given as input by uploading it via the ‘Browse files’ button.
- The uploaded X-ray is checked for pneumonia by pressing the appropriate button.
- The outcome shows that the patient is correctly diagnosed with Pneumonia and is provided a report on the disease. Here model accurately predicted the result
- The report generated gives a description about the disease as well as medical resources and danger level of the disease

Result: 2

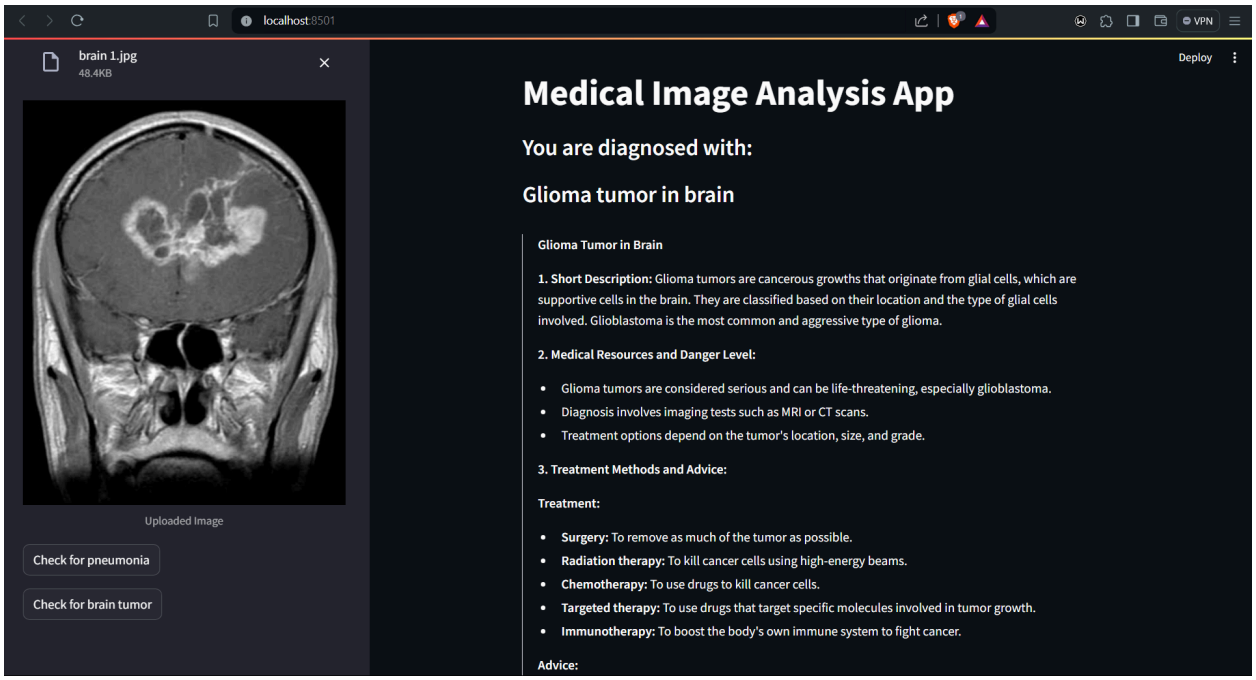


Figure 5.1: Example using CT scan of a Glioma affected brain

Here’s a breakdown of the result:

- A CT scan confirmed Glioma of the brain is given as input by uploading it via the ‘Browse files’ button.
- The uploaded CT scan is checked for tumor by pressing the appropriate button.
- The outcome shows that the patient is correctly diagnosed with brain tumor Glioma tumor and is provided a report on the disease. Here model accurately predicted the result
- The report generated gives a description about the disease as well as medical resources, danger level of the disease, Treatment methods and advice

Result: 3

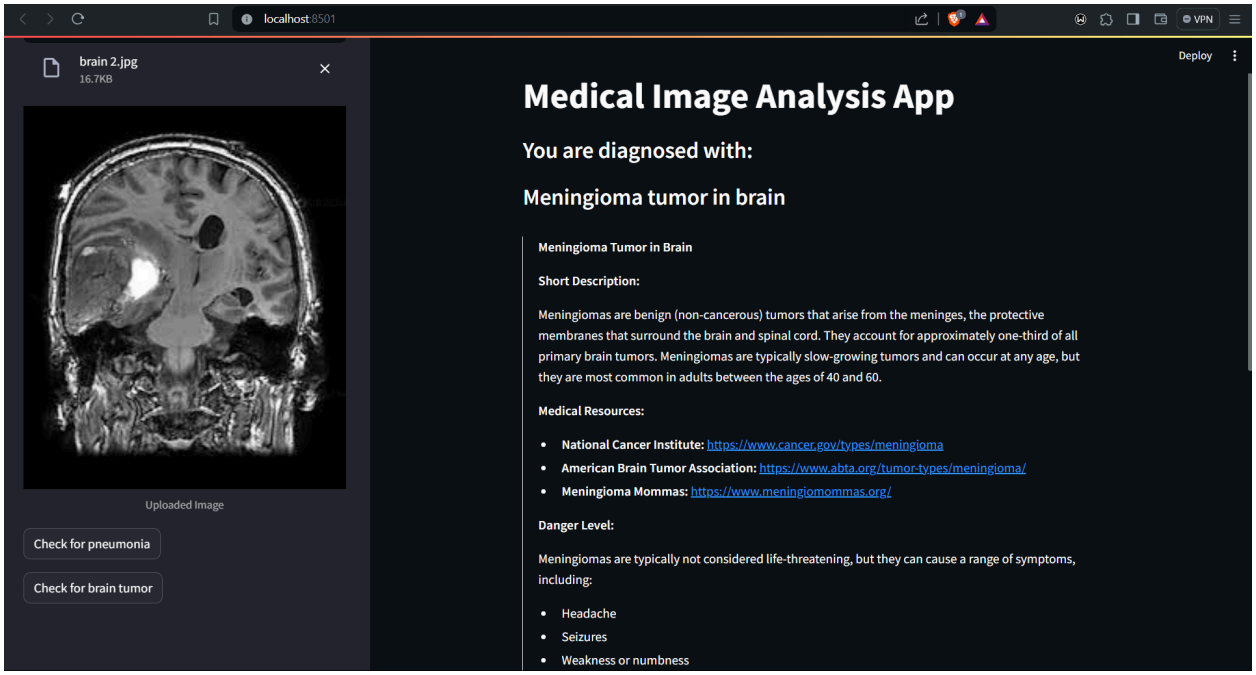
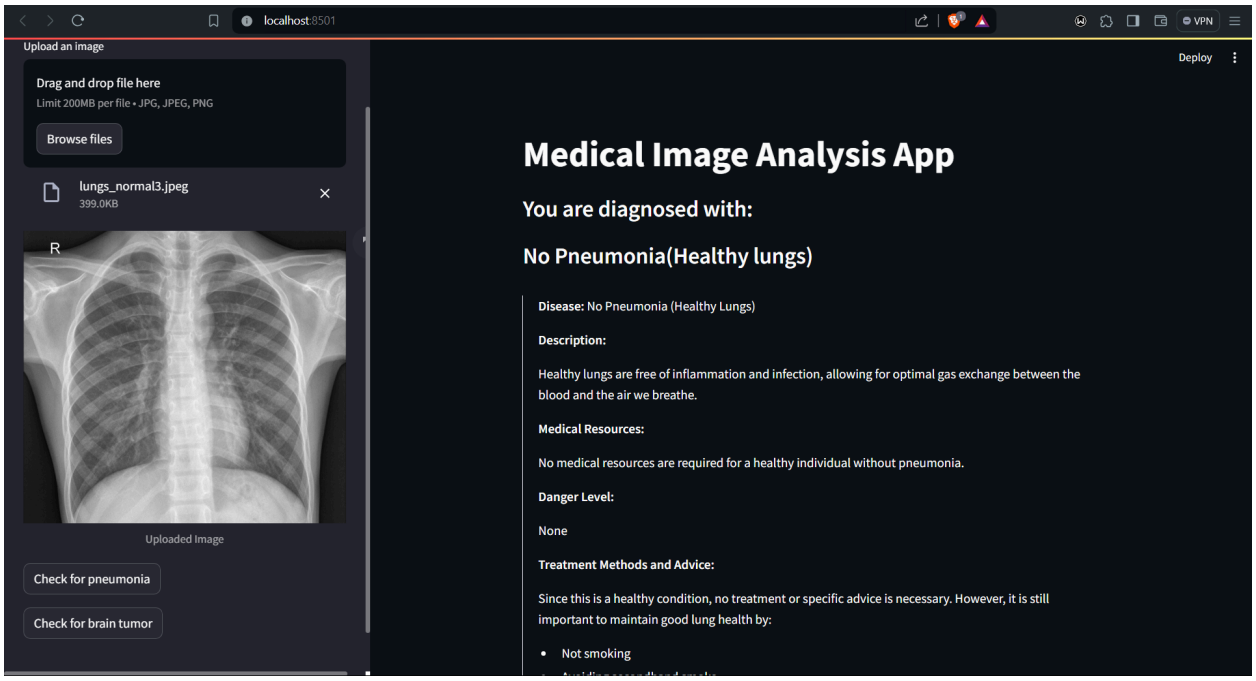


Figure 5.1: Example using CT scan of a Meningioma affected brain

Here’s a breakdown of the result:

- A CT scan confirmed Meningioma of the brain is given as input by uploading it via the ‘Browse files’ button.
- The uploaded CT scan is checked for tumor by pressing the appropriate button.
- The outcome shows that the patient is correctly diagnosed with brain tumor Meningioma tumor and is provided a report on the disease. Here model accurately predicted the result
- The report generated gives a description about the disease as well as medical resources, and danger level of the disease

Result: 4



Here's a breakdown of the result:

- An X-ray confirmed Pneumonia of lungs is given as input by uploading it via the 'Browse files' button.
- The uploaded X-ray is checked for pneumonia by pressing the appropriate button.
- The outcome shows that the patient is correctly diagnosed with Pneumonia and is provided a report on the disease. Here model accurately predicted the result
- The report generated gives a description about the disease as well as medical resources and danger level of the disease

Evaluation

Pneumonia Detection (DenseNet121) Model

```
loss, accuracy = pmodel.evaluate(test_data)
print('The accuracy of the model on test dataset is',
    np.round(accuracy*100))
✓ 45.5s

20/20 ━━━━━━━━━━━━━━━━━━━━ 45s 2s/step - binary_accuracy: 0.8155 - loss: 0.3757
The accuracy of the model on test dataset is 83.0
```

The model shows an accuracy of 83.0% and loss of 0.3757. It showcases a higher accuracy compared to other models. This evaluation metric can be used in refinements to build upon the model further and enhance the performance.

Brain Tumor Detection (ResNet50) Model

```
Epoch 5/49
----------
Training Loss: 0.3974 Acc: 0.8641
Testing Loss: 1.0390 Acc: 0.7995
```

```
Training complete in 48m 52s
Best val Acc: 0.799492
```

The model shows an accuracy of 79.9492% and testing loss of 1.0390. This evaluation metric can be used in refinements to build upon the model further and enhance the performance.

Chapter- 6

Conclusion

In conclusion, this mini-project has successfully developed a medical image analysis application capable of assisting healthcare professionals in detecting brain tumors and pneumonia.

By leveraging deep learning techniques, the application offers valuable insights into medical scans, potentially leading to earlier diagnoses and improved patient outcomes.
A pre-trained ResNet50 model was fine-tuned for brain tumor classification.
A pre-trained Keras/TensorFlow model was integrated for pneumonia detection. A web application interface was created using Streamlit to allow users to upload medical scans and receive predictions for both brain tumors and pneumonia. This interface simplifies interaction for healthcare professionals.

The application's ability to detect brain tumors and pneumonia can empower healthcare professionals to make informed decisions and potentially improve patient care. By addressing the limitations and pursuing the aforementioned future directions, this work has the potential to make a lasting contribution to the field of medical diagnosis.

References

[1]Daniel Greenfield, Sean Wilson, “Artificial Intelligence in Medicine: Applications, Implications, and Limitations”, Article from Harvard University Science in the News, 2019

[2]Pronnoy Dutta, Pradumn Upadhyay,Madhurima De, R.G. Khalkar, ”Medical Image Analysis using Deep Convolutional Neural Networks: CNN Architectures and Transfer Learning”, IEEE 2020 International Conference on Inventive Computation Technologies(ICICT), 2020

[3]Marleen de Bruijne,”Machine learning approaches in medical image analysis: From Detection to Diagnosis”, International Research paper from Erasmus Universiteit Rotterdam, 2016

[4]Meghavi Rana and Megha Bhushan, “Machine learning and deep learning approach for medical image analysis: diagnosis to detection”, Article from National Institute of Health, 2022

[5]Dinggang Shen, Guorong Wu, and Heung-Il Suk, “Deep Learning in Medical Image Analysis”, Annual Review of Biomedical Engineering, 2017

[6]Bas H.M. van der Velden, Hugo J. Kuijf, Kenneth G.A. Gilhuijs, Max A. Viergever. “Explainable artificial intelligence (XAI) in deep learning-based medical image analysis”, Image Sciences Institute, University Medical Center Utrecht, Utrecht, The Netherlands

[7]Kanadpriya Basu, Ritwik Sinha, Aihui Ong, and Treena Basu, “Artificial Intelligence: How is It Changing Medical Sciences and Its Future?”, Journal from the National Library of Medicine, 2020