Intelligent Radiologist Assistant (Lungs Tumor segmentation)

High Level Design (HLD)

Author: Tanjina Proma

Date: 1/03/2024

Table of Contents

Abstract

Lung cancer remains one of the most prevalent and lethal malignancies worldwide. Early detection and accurate classification of lung tumors are crucial for effective treatment planning and patient prognosis. With the advent of deep learning techniques, automated tumor classification from medical images has gained significant attention due to its potential to assist radiologists in clinical decision-making.

In this study, we propose a methodology for lung tumor classification from chest CT scans using the VGG16 convolutional neural network architecture implemented in TensorFlow. The VGG16 model, renowned for its deep architecture and strong feature extraction capabilities, is fine-tuned on a dataset comprising annotated CT scans to perform binary classification of lung tumors into malignant and benign categories.

The preprocessing pipeline involves normalization and augmentation techniques to enhance model generalization and robustness. Subsequently, the preprocessed CT scans are fed into the VGG16 model, which extracts high-level features from the input images. Transfer learning is employed by retraining the fully connected layers of the VGG16 model on the target dataset while keeping the convolutional base frozen.

Performance evaluation is conducted using metrics such as accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC-ROC). Additionally, qualitative assessment through visual interpretation of classification results is performed to validate the model's efficacy in distinguishing between malignant and benign lung tumors.

The experimental results demonstrate the effectiveness of the proposed approach in accurately classifying lung tumors from CT scans. The model achieves competitive performance compared to existing methods, showcasing its potential as a reliable tool for assisting radiologists in lung cancer diagnosis. Furthermore, the proposed methodology can be easily adapted and integrated into clinical workflows to facilitate early detection and personalized treatment strategies for lung cancer patients.

1 Introduction

1.1 Why the HLD?

The accurate classification of lung tumors from chest CT scans is a critical task in medical imaging, facilitating early diagnosis and appropriate treatment planning for patients with suspected or confirmed lung cancer. Deep learning techniques, particularly convolutional neural networks (CNNs), have demonstrated remarkable performance in automated image analysis tasks, including medical image classification. In this context, the proposed high-level design aims to leverage the VGG16 model, a widely-used CNN architecture, to classify lung tumors from chest CT scans with TensorFlow, a popular deep learning framework.

1.2 Objective

The primary objective of this project is to develop a robust and accurate deep learning model capable of classifying lung tumors detected in chest CT scans as malignant or benign. By harnessing the power of the VGG16 architecture and TensorFlow framework, the model seeks to assist radiologists in making timely and informed decisions regarding patient care, thereby potentially improving outcomes for individuals at risk of or diagnosed with lung cancer.

2 General Description

2.1 Product Perspective

The task of lung tumor classification from chest CT scans involves the development of a deep learning system capable of automatically identifying and categorizing lung tumors as “Adenocarcinoma", "Large cell carcinoma", "Normal" or "Squamous cell carcinoma". This process plays a crucial role in aiding radiologists and oncologists in diagnosing and treating lung cancer, which is one of the leading causes of cancer-related deaths worldwide.

2.2 Problem Statement

2.3 Proposed Solution

The proposed approach utilizes the VGG16 convolutional neural network architecture, a well-established deep learning model known for its effectiveness in image classification tasks. TensorFlow, a popular deep learning framework, is employed to implement and train the model due to its flexibility, scalability, and extensive community support.

2.4 Further Improvements

2.5 System Specifications:

Hardware Requirements:

- CPU: A multi-core processor (Intel Core i5 or equivalent) for running training and inference tasks. For faster processing, a high-performance CPU or GPU is recommended.
- GPU (optional but recommended): NVIDIA CUDA-compatible GPU (e.g., NVIDIA GeForce GTX or RTX series) with CUDA support for accelerated deep learning computations.
- RAM: At least 8 GB of RAM for processing large datasets and training deep learning models efficiently.

- Storage: Sufficient storage space for storing datasets, model checkpoints, and Docker images. SSD storage is preferred for faster data access.

Software Requirements:

- Operating System: Compatible with major operating systems such as Windows, macOS, or Linux distributions like Ubuntu.
- Python: Python 3.x (preferably 3.6 or later) for developing and executing the machine learning pipeline.
- TensorFlow: TensorFlow library (version 2.x) for building, training, and deploying deep learning models. Install TensorFlow with GPU support if using a compatible GPU.
- MLflow: MLflow library for managing the machine learning lifecycle, including experiment tracking, model packaging, and deployment. Install MLflow using pip.
- Docker: Docker engine for containerizing the application and ensuring consistent execution environments across different systems. Install Docker Desktop for Windows or Docker CE for Linux.
- DagsHub: DagsHub platform for version control, collaboration, and continuous integration of machine learning projects. Sign up for a DagsHub account and install the DagsHub CLI.
- DVC (Data Version Control): DVC for managing data versioning and reproducibility in machine learning projects. Install DVC using pip or package manager.

Dependencies and Libraries:

- NumPy, Pandas: Essential libraries for data manipulation, preprocessing, and analysis.
- scikit-learn: Library for machine learning algorithms, including data splitting, evaluation metrics, and model selection.
- Matplotlib, Seaborn: Visualization libraries for creating plots, charts, and visualizing model performance.
- OpenCV: Library for image processing and computer vision tasks, including loading and preprocessing CT scan images.

Additional Tools and Services:

- Git: Version control system for tracking changes in code, configurations, and project files.
- GitHub or GitLab: Online platforms for hosting code repositories, collaborating with team members, and managing project development.
- Docker Hub: Registry service for storing and sharing Docker images, facilitating Docker image distribution and deployment.
- MLflow Tracking Server: Centralized server for storing experiment metadata, tracking metrics, and organizing MLflow experiments.
- Continuous Integration (CI) Services: Integration with CI services like GitHub Actions for automating testing, building, and deploying machine learning models.
- Cloud Services: AWS for scalable compute resources, storage, and deployment of machine learning applications.

2.6 Data Requirements

2.7 Constraints

2.8 Assumptions

3 Design Details