Intelligent Radiologist Assistant (Lungs Tumor segmentation)

High Level Design (HLD)

Author: Tanjina Proma

Date: 12/09/2024

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

The model will use deep learning and computer vision to detect the disease. The system will automatically localize and classify various types of thoracic abnormalities from chest radiographs. It will make use of the scans that have been annotated by experienced radiologists.

So, the main objectives are as follows -

- Create a model to segment the lung tumor present in the CT scans.
- Create a portal where doctors and patient can login.
- This portal should use our model to generate a sample report based on the uploaded CT scan of Lungs.
- Doctor should be able to login and upload the CT scans in the portal and can generate report using the model in a bulk.
- Patient should be able to login and generate report too if they possess their Lungs CT scan or they should see the report generated from the hospital or doctor using our model.

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

Further improvements could involve exploring ensemble methods to combine predictions from multiple models for enhanced performance. Implementing model interpretability techniques such as Grad-CAM or SHAP can provide insights into model decisions. Integration of advanced data augmentation methods and utilization of larger datasets can improve model generalization. Employing transfer learning with more sophisticated architectures like DenseNet or ResNet may capture finer tumor features. Lastly, optimizing Docker image size and leveraging cloud-based GPU resources for faster computations can streamline the deployment process, enhancing scalability and efficiency.

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 3.8 for developing and executing the machine learning pipeline.
- TensorFlow: TensorFlow library (version 2.12.0) 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: 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

Link: https://www.kaggle.com/datasets/mohamedhanyyy/chest-ctscan-images/data

Images are not in dcm format, the images are in jpg or png to fit the model. Data contain 3 chest cancer types which are Adenocarcinoma, Large cell carcinoma, Squamous cell carcinoma, and 1 folder for the normal cell. Data folder is the main folder that contain all the step folders inside Data folder are test, train and valid.

test represent testing set consisting of **20%** of the data.
train represent training set consisting of **70%** of the data.
valid represent validation set consisting of **10%** of the data.

2.6.1 Adenocarcinoma

Adenocarcinoma of the lung: Lung adenocarcinoma is the most common form of lung cancer accounting for 30 percent of all cases overall and about 40 percent of all non-small cell lung cancer occurrences. Adenocarcinomas are found in several common cancers, including breast, prostate and colorectal. Adenocarcinomas of the lung are found in the outer region of the lung in glands that secrete mucus and help us breathe. Symptoms include coughing, hoarseness, weight loss and weakness.

2.6.2 Large cell carcinoma

Large-cell undifferentiated carcinoma: Large-cell undifferentiated carcinoma lung cancer grows and spreads quickly and can be found anywhere in the lung. This type of lung cancer usually accounts for 10 to 15 percent of all cases of NSCLC. Large-cell undifferentiated carcinoma tends to grow and spread quickly.

2.6.3 Squamous cell carcinoma

Squamous cell: This type of lung cancer is found centrally in the lung, where the larger bronchi join the trachea to the lung, or in one of the main airway branches. Squamous cell lung cancer is responsible for about 30 percent of all non-small cell lung cancers, and is generally linked to smoking.

And the last folder is the normal CT-Scan images.

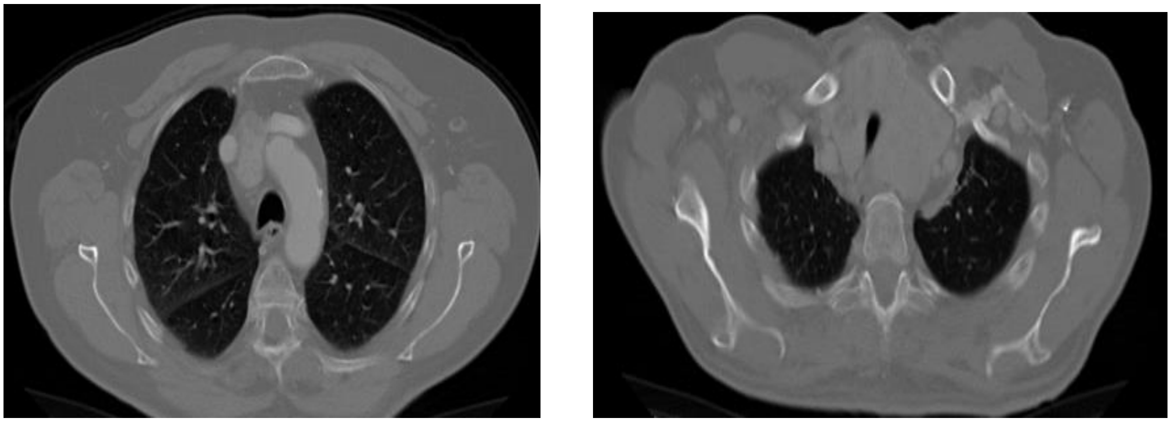

Fig: Adenocarcinoma & Large cell carcinoma

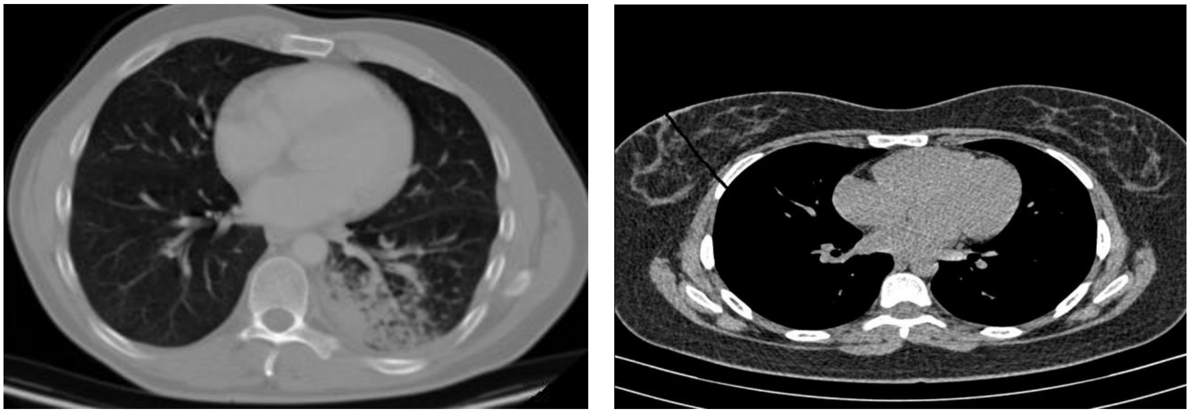

Fig: Squamous cell carcinoma & Normal

2.7 Constraints

- Computational Resources: Availability of GPU resources for efficient model training.
- Latency: Real-time inference may be subject to latency constraints.
- Data Privacy: Adherence to data privacy regulations when handling potentially sensitive content.

2.8 Assumptions

- Training Data Quality: Assumes the availability of a high-quality labeled dataset for effective model training.
- Model Generalization: Assumes the trained model can generalize well to classify diverse chest CT scan images.

3 Design Details

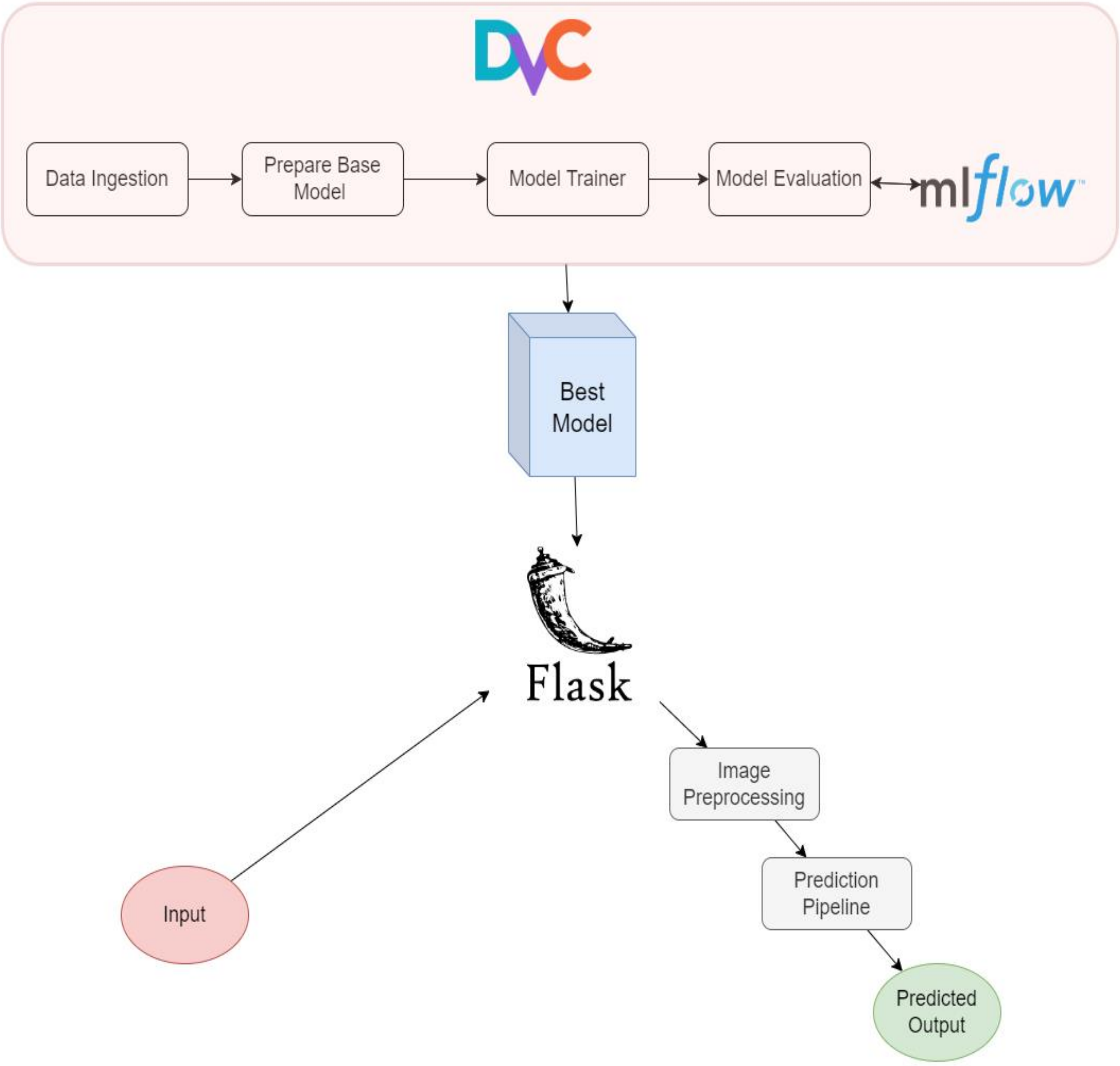

Fig: Flowchart

3.1 Process Flow

1. **Data Ingestion:**

   - Collect images
   - from Kaggle dataset link which included dataset created from BBC Images data.

2. **Prepare Base Model:**
    - Choose the VGG16 convolutional neural network architecture as the base model for feature extraction due to its effectiveness in image classification tasks.
    - Import the VGG16 model architecture and weights pretrained on ImageNet from the TensorFlow library.
    - Choose the VGG16 convolutional neural network architecture as the base model for feature extraction due to its effectiveness in image classification tasks.
    - Import the VGG16 model architecture and weights pretrained on ImageNet from the TensorFlow library.
3. **Model Training:**
    - Split the preprocessed data into training, validation, and test sets.
    - Choose the VGG16 convolutional neural network architecture as the base model for feature extraction due to its effectiveness in image classification tasks.
    - Import the VGG16 model architecture and weights pretrained on ImageNet from the TensorFlow library.
4. **Evaluation:**
    - Assess the model's performance using evaluation metrics on the validation set.
    - Fine-tune hyperparameters based on evaluation results.
    - Evaluate the model using metrics such as accuracy, sensitivity, specificity, and AUC-ROC on a validation dataset. Use MLflow to log and visualize metrics.
    - Conduct qualitative assessment by visualizing model predictions overlaid on CT scan images.
5. **Inference:**
    - Deploy the trained model to make predictions on new, unseen images.
    - Classify incoming images into predefined categories using the deployed model.
6. **Deployment Process**
    - **API endpoint:** Flask serves as access points for interacting with the model through APIs, allowing users to input data and receive processed results. Accepts inputs and categorizes them into predefined classes or labels.
    - **Containerization**: Containerize the model using Docker for reproducible deployment across different environments.
    - **MLflow Model Packaging:** Package the trained model using MLflow's model packaging capabilities.

- **Deployment Pipeline**: Create a deployment pipeline using DVC to manage data and MLflow to manage model versions.
- **Version Control**: Use Git for version control, tracking changes in code, configurations, and datasets.
- **Continuous Integration**: Integrate CI/CD pipelines with DagsHub to automate testing, building, and deployment processes.
- **Deployment**: Deploy the packaged model to production environments or cloud platforms using Docker containers.
- **Model Monitoring**: Implement monitoring mechanisms to track model performance and detect drifts in predictions over time.
- **Reproducibility**: Ensure reproducibility of experiments and results using DVC for data and MLflow for model tracking.
- **Regular Updates:** Update the model periodically with new data and retrain as necessary to adapt to evolving patterns in lung tumor characteristics.
- **Cloud** : AWS ECR (Elastic Container Registry) is a service for storing, managing, and deploying Docker container images. EC2 (Elastic Compute Cloud) provides scalable virtual servers for various computing needs on AWS.

3.2 Event Log

- **Data Ingestion Log:** Records the sources of collected images.
- **Prepare Base Model Log:** Stores information about the preprocessing steps applied to the images.
- **Training Log:** Records training progress, including metrics like loss and accuracy.
- **Evaluation Log:** Stores evaluation metrics and results.
- **Inference Log:** Logs predictions made by the deployed model along with confidence scores

3.3 Error Handling

- **Data Ingestion Errors:** Log failed attempts to collect or preprocess images, notify administrators, and retry mechanisms.
- **Prepare Base Model Errors:** Handle tokenization or formatting failures, logging details for debugging.
- **Training Errors:** Monitor for convergence issues, log and alert for anomalies in training behavior.
- **Evaluation Errors:** Alert on unexpected evaluation metric behavior, triggering reevaluation.
- **Inference Errors:** Handle errors in inference due to model or input issues, logging details for debugging and improvement.

4 Performance

4.1 Quantitative Metrics:

- **Accuracy:** The ratio of correctly classified lung tumor samples to the total number of samples. It provides an overall assessment of the model's correctness.

- **Precision:** The proportion of true positive predictions out of all predicted positive instances (both true positives and false positives). It measures the accuracy of positive predictions.

- **F1-Score:** The harmonic mean of precision and recall, providing a balance between the two metrics. It accounts for both false positives and false negatives.

- **Area Under the Receiver Operating Characteristic curve (AUC-ROC):** A performance measure that evaluates the model's ability to discriminate between positive and negative classes across various thresholds.

4.2 Qualitative Evaluation:

- **Visual Inspection:** Reviewing sample predictions to assess the model's performance qualitatively. Overlaying predicted tumor regions on CT scan images can help identify any misclassifications or areas of improvement.

- **Confusion Matrix:** A tabular representation of predicted vs. actual classifications, showing true positives, true negatives, false positives, and false negatives. It provides insights into the types of errors made by the model.

- **ROC Curve:** Plotting the true positive rate (sensitivity) against the false positive rate for different threshold values. It helps visualize the trade-off between sensitivity and specificity.

- **Precision-Recall Curve:** Plotting precision against recall for different threshold values. It provides insights into the model's performance across various decision thresholds.

4.3 Model Performance Tracking:

- **MLflow Tracking:** Utilize MLflow to log and track performance metrics, hyperparameters, and model artifacts during training and evaluation.

- **Experiment Comparison:** Compare performance across different experiments using MLflow's experiment tracking capabilities, allowing for iterative improvements.

- **Version Control:** Use DagsHub and DVC to version control datasets, code, and model configurations, ensuring reproducibility and facilitating collaboration.

5 Dashboards

5.1 KPIs (Key Performance Indicators)

- **Model Performance Metrics:** Tracking key metrics like accuracy, precision, recall, and F1-score obtained during model evaluation on the validation set. Aim for high accuracy (>90%) to ensure reliable classification performance.

- **Inference Metrics:** Minimize inference time to ensure timely and efficient processing of CT scans in clinical settings, aiming for inference times within acceptable limits (e.g., milliseconds per image).
- **Training Metrics:** Keep track of training loss, convergence, and learning rate for assessing model training progress.
- **Deployment Latency:** Minimize deployment latency to ensure rapid deployment and availability of the model for clinical use, aiming for efficient deployment processes with minimal delays.
- **Model Versioning and Management:** The ability to manage and track different versions of the model, including updates, modifications, and performance improvements. Maintain organized model versioning using DVC and MLflow, ensuring clear documentation and traceability of changes throughout the development lifecycle.

6 Conclusion

In conclusion, the integration of VGG16 Model in TensorFlow, Python, Docker, MLflow, DagsHub, and DVC facilitates the development of a robust and reproducible lung tumor classification system. Leveraging pre-trained weights, transfer learning, and continuous monitoring, the model ensures accurate classification of chest CT scans. MLflow and DVC streamline experiment tracking and data versioning, while Docker enables seamless deployment across diverse environments. Collaborative features of DagsHub enhance team collaboration and CI/CD pipelines automate testing and deployment. This comprehensive approach not only enhances diagnostic accuracy but also contributes to improved patient outcomes in lung cancer diagnosis and treatment.