Cairo University
Faculty of Computers and Artificial
intelligence Department of Computer Sciences

Mobile HPC Systems

Supervised by

Dr. Ahmed Shawky

Implemented by

20210049	*Ahmed Yehia*
20210133	*Rana Essam*
20210413	*Mohannad Hisham*
20210430	*Noureldin Ahmed*
20210500	*Merna Islam*

Graduation Project
Academic Year 2024-2025
Final Year Documentation

Chapter 1: Introduction

Abstract

High-Performance Computing (HPC) has significantly enhanced data processing and computational efficiency, enabling advancements in various fields, including healthcare. This project explores the integration of HPC into mobile systems for breast cancer tumor volume estimation using MRI scans. By leveraging Convolutional Neural Networks (CNNs) and TensorFlow Lite, the system automates tumor seed detection, reducing reliance on manual input and improving segmentation accuracy.

A key feature of this solution is its offline functionality, allowing seamless operation in hospitals and diagnostic centres with limited or no internet access. The mobile HPC approach ensures real-time, on-premises analysis, addressing concerns related to data privacy, latency, and dependency on cloud-based systems. This innovation supports healthcare professionals by streamlining the diagnostic workflow, reducing workload, and enabling faster, more precise assessments.

By bridging the gap between mobile computing and medical imaging, this project empowers clinicians with a portable, high-performance tool for early cancer detection and treatment planning. The proposed solution aims to enhance diagnostic accessibility, particularly in resource-constrained settings, ultimately improving patient outcomes and advancing the role of mobile HPC in medical applications.

Background

High-Performance Computing (HPC) plays a transformative role in solving complex, data-intensive problems by providing immense computational power. HPC's effectiveness lies in its ability to process massive datasets and execute computations in parallel, drastically reducing the time required for analysis. It enables breakthroughs by handling challenges that demand both speed and accuracy. With the advancement of mobile computing, the integration of HPC capabilities into mobile systems offers new opportunities for real-time, on-the-go problem-solving.

This project explores the application of mobile HPC systems in the medical field, specifically in the estimation of breast cancer tumor volume using MRI images. A critical aspect of this project is ensuring offline functionality, allowing the system to operate effectively in environments with no or limited internet access, such as hospitals in remote or resource-constrained settings. By eliminating the need for cloud connectivity, the solution empowers healthcare providers with real-time, on-site diagnostic capabilities.

Main Area

The core focus of this project lies in demonstrating the effectiveness of High-Performance Computing (HPC) on mobile systems, particularly in the healthcare sector. HPC has traditionally been associated with large-scale computing clusters and data centres, but recent advancements in mobile computing power have enabled the deployment of complex computational tasks on portable devices. This project aims to showcase how mobile HPC can be leveraged to perform advanced data processing tasks efficiently, proving its feasibility in real-world applications.

One of the most critical applications of mobile HPC is in medical imaging, where timely and accurate analysis can significantly impact patient outcomes. By harnessing mobile HPC capabilities, the project implements an advanced diagnostic tool that processes MRI images to estimate tumor volume. Machine learning techniques, particularly those optimized with TensorFlow Lite, ensure precise tumor seed detection, enabling automated and accurate tumor volume assessment. The ability to perform such computations offline makes this solution highly beneficial for healthcare providers operating in regions with limited or no internet access.

Beneficiary

The primary beneficiaries of this project include:

- **Doctors:** can benefit from quick and accurate tumor analyses, enabling more informed decision-making for treatment and monitoring.
- **Patients:** Breast cancer patients can receive faster and more accurate diagnoses, improving the chances of early intervention and successful treatment.
- **Hospitals and Diagnostic Centres:** Especially those in remote or resource-limited regions, can utilize the portable and efficient system to enhance their diagnostic capabilities, even in the absence of reliable internet connectivity.

Motivation

High-Performance Computing (HPC) has revolutionized numerous fields, from scientific research to financial modelling. However, its potential in mobile systems remains underutilized. With the increasing processing power of mobile devices, there is a growing need to explore how HPC can be effectively integrated into mobile applications. The motivation behind this project is to demonstrate that mobile HPC is not just a theoretical concept but a practical and impactful solution, particularly in critical domains such as healthcare.

Breast cancer remains one of the most prevalent and deadly cancers worldwide, with an estimated **2.3** million new cases and **685,000** deaths in **2020**. In Egypt, it is the most common cancer among women, accounting for approximately **38%** of all cancer cases in the country.

Early detection and precise assessment of tumor characteristics are crucial for effective treatment and improved patient outcomes. Traditional diagnostic methods, while effective, can be resource-intensive and slow, making them less accessible in underprivileged areas.

By applying mobile HPC to breast cancer diagnostics, this project seeks to bridge the gap between cutting-edge computational capabilities and real-world healthcare needs. The implementation of mobile HPC in this domain not only proves the feasibility of running complex medical imaging algorithms on portable devices but also highlights its potential to improve healthcare accessibility. By integrating efficient image processing, machine learning, and offline functionality, the project empowers healthcare providers with a portable, accurate, and cost-effective solution for early breast cancer detection and assessment.

To achieve scalable and efficient parallel processing across multiple mobile devices, this project also incorporates gRPC (Google Remote Procedure Call) as a communication protocol. gRPC enables fast, reliable, and structured messaging between devices, allowing them to dynamically share computational tasks. Using gRPC, one mobile device can act as a coordinator, distributing matrix-based image processing workloads to several worker devices over a local network. This architecture not only demonstrates the practicality of real-time distributed computing on mobile platforms but also significantly accelerates complex computations like convolutional neural network inference and fuzzy logic-based tumor volume estimation. The seamless integration of gRPC supports the project's vision of decentralized, offline-capable medical diagnostics that leverage the collective power of mobile devices in resource-constrained environments.

Problem Definition

In sensitive fields such as healthcare, there is a growing demand for **on-premises**, **portable**, and **high-performance** computing solutions. This need arises from the limitations of alternatives like cloud and edge computing, which can incur high operational costs and remain vulnerable to disruptions in internet connectivity. For medical applications in particular, reliance on remote systems introduces risks to data privacy, latency in critical workflows, and dependency on external infrastructure—factors that are often unacceptable in time-sensitive or security-conscious environments.

This challenge is exemplified in the development of a **breast cancer volume estimation system**, the focus of this project. Accurate tumor volume measurement requires segmenting cancerous regions across MRI slices using a fuzzy logic-based system, where intensity thresholds differentiate malignant tissues from healthy ones. However, current implementations rely on clinicians to manually select a "seed point" in *every slice* to initialize segmentation—a repetitive and error-prone task that delays diagnosis. To address this, our project introduces an **automated seed detection component** that integrates with the fuzzy system, eliminating manual input while preserving segmentation accuracy. By streamlining this process, the solution directly supports the need for portable, on-premises tools that reduce clinician

workload, ensure data privacy, and operate reliably in connectivity-limited clinical environments.

To further enhance performance and portability, especially when processing large image datasets or performing computation-intensive tasks such as MRI segmentation and tumor volume estimation, the system leverages **Google Remote Procedure Call (gRPC)** for distributed computing across multiple mobile devices. gRPC enables efficient, scalable communication between a central coordinator device and several worker devices over a local network, allowing computation to be dynamically offloaded and shared without depending on cloud services. This not only accelerates processing but also maintains full data locality, ensuring patient information never leaves the trusted environment. The inclusion of gRPC strengthens the system's ability to deliver real-time results with low latency, increased fault tolerance, and robust scalability, all while operating entirely offline.

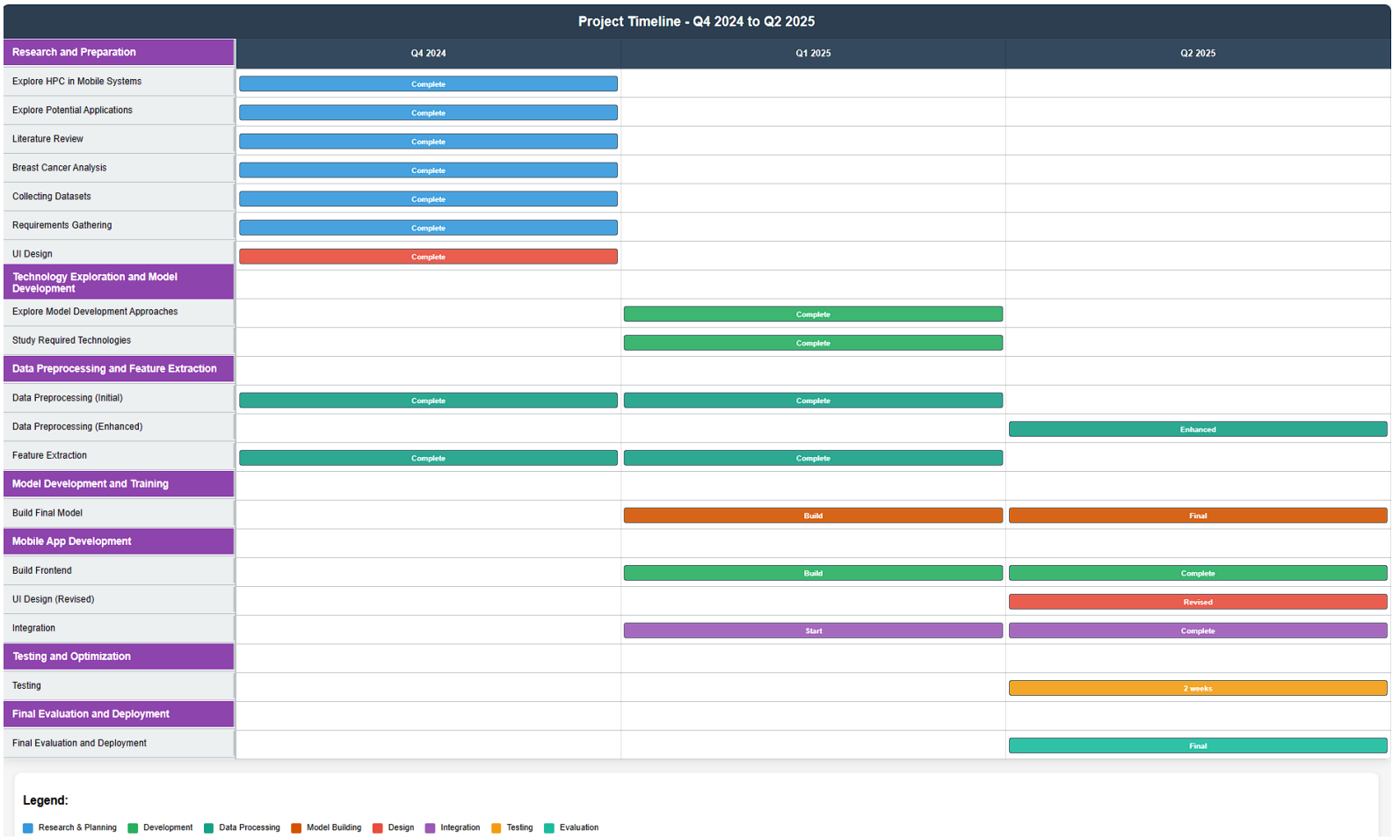
Project Objective

The goal of this project is to highlight the importance of achieving high-performance computing (HPC) on mobile devices and to demonstrate practical approaches for making this possible. As mobile devices become increasingly powerful and ubiquitous, enabling them to handle computationally intensive tasks opens new opportunities in fields such as medical imaging, real-time analytics, and on-device AI. This project explores strategies for optimizing performance, reducing latency, and leveraging hardware acceleration to bring HPC capabilities directly to mobile platforms without relying on external servers or constant internet connectivity.

Gantt chart of project time plan

No.	Task Title	Description	Status
1	Explore HPC in Mobile Systems	Research how High-Performance Computing (HPC) is used in mobile systems, including benefits, limitations, and use cases.	Completed
2	Explore Potential Applications	Investigate real-world applications that use HPC concepts, such as GIS, real-time simulations, and medical imaging systems.	Completed
3	Literature review	Collect and study previous research similar to the project to understand existing solutions, challenges, and methodologies.	Completed
4	Breast Cancer Analysis	Study how radiologists detect breast cancer in MRI scans and identify key image features that support diagnosis.	Completed
5	Collecting Datasets	Search and select appropriate MRI datasets suitable for tumor detection and segmentation tasks.	Completed
6	Explore model development approaches	Explore various model development strategies, including traditional rule-based methods and modern techniques like CNNs.	Completed
7	Study the needed Technologies	Studied Docker, TensorFlow Lite, Kotlin, Ultralytics YOLO, and gRPC for model deployment, mobile development, and distributed processing.	Completed
8	Data Preprocessing (Initial)	Perform the initial cleaning, normalization, and preparation of MRI images for use in model training.	Completed
9	Data Preprocessing (Enhanced)	Improve the preprocessing pipeline to enhance image quality, standardize inputs, and ensure compatibility with parallel processing frameworks.	
10	Feature Extraction	Extracting features unique to the tumor region in the MRI images after preprocessing through texture analysis.	Completed
11	Build the final model	Construct the final model using selected techniques and validate it to ensure optimal performance on the dataset.	Completed
12	Build frontend	Develop the Kotlin-based mobile application interface for interacting with the prediction model.	Completed

13	UI Design (Revised)	Refine and update the UI to integrate the model, seed point prediction, and fuzzy logic system into the app.	Completed
14	Integration	Integrate the trained model and fuzzy logic system into the app for seamless user interaction and real-time processing.	Completed
15	Build grpc	Implement the network communication layer using gRPC to enable distributed task sharing among multiple mobile devices.	Completed
16	Testing	Test the app's performance across different patient groups and conditions to evaluate accuracy, latency, and usability.	Completed
17	UI Design	Create the initial design layout and structure for the mobile application UI.	Completed
18	Requirements	Identify and document both functional and non-functional requirements of the system.	Completed
19	Diagrams	Design all necessary diagrams, including architecture, data flow, use case, and system interaction diagrams.	Completed


Project Timeline - Q4 2024 to Q2 2025
Research and Preparation
Q4 2024
Q1 2025
Q2 2025
Explore HPC in Mobile Systems
Complete
Explore Potential Applications
Complete
Literature Review
Complete
Breast Cancer Analysis
Complete
Collecting Datasets
Complete
Requirements Gathering
Complete
UI Design
Complete
Technology Exploration and Model Development
Explore Model Development Approaches
Complete
Study Required Technologies
Complete
Data Preprocessing and Feature Extraction
Data Preprocessing (Initial)
Complete
Complete
Data Preprocessing (Enhanced)
Enhanced
Feature Extraction
Complete
Complete
Model Development and Training
Build Final Model
Build
Final
Mobile App Development
Build Frontend
Build
Complete
UI Design (Revised)
Revised
Integration
Start
Complete
Testing and Optimization
Testing
2 weeks
Final Evaluation and Deployment
Final Evaluation and Deployment
Final
Legend:
Research & Planning
Development
Data Processing
Model Building
Design
Integration
Testing
Evaluation

Project development methodology

1. Modular Development Approach

The project was designed using a modular and iterative development methodology. The core components—Region of Interest (ROI) detection, seed point identification, and fuzzy logic–based tumor volume estimation—were developed and tested independently. This separation allowed for focused validation, easier debugging, and flexibility in system updates.

---

2. Model Training and Conversion

The ROI detection model was implemented using YOLOv8 and trained on annotated breast MRI images. Once trained, the model was converted to TensorFlow Lite (.tflite) format to support lightweight and efficient inference on mobile or resource-constrained environments. A second YOLOv8 model was trained specifically for **seed point detection**, targeting the tumor's central location within the detected region of interest. The final stage, **fuzzy logic–based tumor volume estimation**, was fully implemented in Kotlin, ensuring seamless integration within the main application and real-time execution on the client device.

---

3. Kotlin Application Integration

All system components were integrated into a Kotlin-based application that acts as the main user interface. This application is responsible for:

- Managing the workflow between components
- Processing input MRI images
- Displaying results to the user
- Providing an intuitive and responsive user experience

---

4. Execution Mode Switching

A key feature of the main Kotlin application is its ability to **allow users to manually configure the execution mode** for each major component—ROI detection, seed point localization, and fuzzy logic processing—based on device capabilities and user preferences.

- **GPU acceleration** can be independently **enabled or disabled by the user** for each of the two YOLOv8 models: one for detecting the Region of Interest (ROI) and another for seed point identification. This provides users with control over inference performance, allowing for high-speed processing on supported devices or power-efficient CPU execution when needed.

- **Fuzzy logic processing** can also be toggled between **concurrent execution using Kotlin coroutines** or **sequential execution**, depending on whether the user prioritizes speed or resource conservation.

This user-driven configurability makes the application highly adaptable, offering optimized performance for high-end environments and efficient operation on lower-resource or battery-sensitive devices. It also enhances usability by letting users tailor the system's behavior to their specific workflow or device limitations.

---

5. gRPC-Enabled Version

In addition to the standalone offline version, a **gRPC-based version** of the application was developed to support **distributed local processing** across multiple mobile or edge devices—**without relying on cloud infrastructure or internet connectivity**.

This version enables:

- **Distributed Task Processing:** Computation tasks are automatically split and distributed across multiple devices in the local network, allowing parallel processing of large medical datasets.

- **Real-Time Coordination:** A central coordinator manages task distribution, monitors worker status, and aggregates results in real-time using gRPC communication protocols.

- **Dynamic Worker Discovery:** Devices automatically discover and register with the coordinator using mDNS (multicast DNS) service discovery, enabling plug-and-play scalability.

- **Fault-Tolerant Execution:** Failed tasks are automatically redistributed to available workers, ensuring computation completion even if individual devices become unavailable.
- **Live Progress Monitoring:** Real-time event streaming provides immediate feedback on computation progress, worker status, and task completion across all participating devices.
- **Network-Independent Operation:** All communication occurs over local network protocols (gRPC over HTTP/2), eliminating dependency on internet connectivity or cloud services.
- **Scalable Performance:** Adding more devices to the network linearly increases computational capacity, allowing the system to handle larger datasets or achieve faster processing times.
- **Cross-Platform Compatibility:** The gRPC implementation supports heterogeneous device environments, enabling collaboration between different types of mobile and edge devices.Distributed Task Processing: Computation tasks are automatically split and distributed across

The used tools in the project (SW and HW)

1.1.1. Frontend Technologies

1. **Kotlin:**

A modern, statically typed programming language developed by JetBrains. It runs on the Java Virtual Machine (JVM) and is fully interoperable with Java. Known for its concise syntax, Kotlin offers enhanced safety features like null-safety and immutability. It is officially supported for Android development and widely used for mobile, web, and backend applications.

1.1.2. Machine Learning Technologies

1. **YOLOv8s (You Only Look Once – Version 8, Small)**

It's a cutting-edge object detection model known for its real-time performance and high accuracy. YOLOv8s is part of the YOLO (You Only Look Once) family, which is widely adopted in computer vision applications for its ability to detect objects efficiently in a single forward pass.

The **"s"** variant represents a **small and lightweight version** of the model, making it well-suited for deployment in mobile and resource-constrained environments.

In our system, two separate YOLOv8s models were fine-tuned on medical imaging data:

2. **ROI Detection:** The first model detects the region of interest (tumor area) in sagittal breast MRI slices.

3. **Seed Point Detection:** The second model identifies a representative seed point inside the detected tumor region.

4. **TensorFlow Lite**

A framework that enables the deployment and execution of machine learning models on mobile and embedded devices. It facilitates the use of machine learning technologies (such as models built with TensorFlow) in environments with limited resources, providing efficient inference for tasks like image classification, speech recognition, and more on mobile or edge devices. The integration of TensorFlow Lite ensures that the system can run entirely offline, making it suitable for use in healthcare settings with low or no internet access.

2.1.3. Google Remote Procedure Call (gRPC)

gRPC is a high-performance, open-source framework developed by Google for enabling efficient communication between distributed systems. It uses Protocol Buffers (protobuf) for defining service contracts and serializing structured data, making it more efficient than traditional REST APIs in terms of speed and bandwidth usage.

In mobile or distributed computing environments, gRPC allows different components (e.g., mobile clients, backend servers, worker nodes) to communicate in a type-safe, scalable, and efficient manner. It supports bi-directional streaming, authentication, load balancing, and deadlines/timeouts.

In this system, gRPC is used to coordinate tasks between mobile devices in a distributed computation setting. It facilitates seamless communication between a coordinator and worker devices, enabling them to exchange matrix data and results efficiently during tasks like distributed matrix multiplication or image processing in a federated or peer-to-peer setup.

Report Organization (summary of the rest of the report)

The remainder of this report is organized into the following chapters:

- **Chapter 2: Related Work**

  This chapter provides a review of previous research related to fuzzy segmentation methods, seed point detection strategies, and the use of machine learning in medical imaging. It examines fuzzy connectedness approaches for tumor volume estimation, including both

manual and automatic seed selection techniques. It also explores machine learning concepts such as radiomics and texture analysis in breast MRI, and highlights practical implementations of TensorFlow Lite in real-world mobile applications. This chapter establishes the foundation upon which our system is built and identifies the specific gaps our project aims to address—namely, eliminating manual input, ensuring offline functionality, and supporting distributed computation on mobile devices.

- **Chapter 3: System Analysis**

  This chapter outlines the detailed specification of the proposed system. It begins with a description of the functional requirements, including automatic Region of Interest (ROI) detection, seed point localization using YOLO model, fuzzy logic-based tumor volume estimation, result visualization, and the gRPC-based distribution of computational tasks across devices. Non-functional requirements such as performance optimization, energy efficiency, device compatibility, security, and usability are also discussed. Finally, the chapter presents use case diagrams that illustrate how users interact with the system to perform tasks like ROI detection, seed point localization and volume estimation.

- **Chapter 4: System Design**

  This chapter presents the architectural and structural design of the system. It includes the system component diagram outlining the relationships between the YOLO models, fuzzy logic module, gRPC communication layer, and user interface. Class diagrams describe the main software entities involved in handling MRI data, seed point information, and distributed tasks. Sequence diagrams illustrate key processes such as seed detection, segmentation execution, and result display. An Entity-Relationship Diagram (ERD) is provided to describe the data storage structure. The chapter also includes mobile GUI designed with Jetpack Compose, focusing on usability, accessibility, and clinical readability of outputs.

- **Chapter 5: Implementation and Testing**

  This chapter describes the implementation of the system on Android devices using Kotlin. It details the integration of TensorFlow Lite for YOLO-based seed detection, the fuzzy logic algorithm for tumor volume calculation, and gRPC for real-time distributed computation across nearby mobile devices. Screenshots and code samples illustrate the working modules. The chapter also includes test cases that validate individual components and the system as a whole. These tests cover the accuracy of seed detection, performance of segmentation and volume estimation, responsiveness of the UI, and correctness of multi-device coordination via gRPC under offline conditions.

Chapter 2: Related Work

❖ Multiple research papers have similar work. Our target is to combine the best parts of these papers and produce the most accurate results.

1. A Fuzzy Tumor Volume Estimation Based on Fuzzy Segmentation of MR Images:[1] This paper inspired us by introducing a method to measure tumor size in MRI images using a fuzzy connectedness approach. Their technique creates a fuzzy map of the tumor area and allows users to adjust the level of certainty with an Alpha Cut setting. This helps doctors make better treatment decisions by providing a more flexible way to measure tumors. Inspired by their work, we use a similar idea in our project to find the best seed point in breast cancer MRI images. However, one limitation of our approach is that we need an expert to manually select the seed point, which can be time-consuming and may introduce human error. To improve this, we plan to use an automatic seed selection method, like the one introduced in another study, to make the process more efficient and accurate.
2. An Improved Fuzzy Connectedness Method for Automatic Three-Dimensional Liver Vessel Segmentation:[2] This paper uses fuzzy connectedness for segmentation too. However, the first approach requires an expert to manually select the seed point, while the second paper presents an improved method that automatically generates a seed using an adaptive algorithm. Instead of relying on an expert, we plan to apply the automatic seed selection method from the second paper to the first approach. This way, we can make the process more efficient and reduce human involvement while still benefiting from fuzzy connectedness for accurate segmentation.
3. Machine Learning in Breast MRI:[3] From this paper, we gained a broader understanding of how machine learning is applied to breast MRI analysis. It introduced key concepts such as radiomics, which involves extracting large amounts of data from medical images to support decision-making. One important aspect is texture analysis, which looks at patterns and variations in the image to help distinguish between different types of tissues. These insights have helped us see how feature extraction techniques can be used to improve accuracy in medical imaging, providing more detailed and reliable information for diagnosis and treatment planning.
4. A Fuzzy-C-Means Approach for Tissue Volume Estimation in Brain MRI Images:[4] Fuzzy C-Means (FCM) is another method used for volume estimation in medical images. It works by grouping similar pixels based on their intensity, which helps in separating different tissues like white matter, gray matter, and cerebrospinal fluid in brain MRI scans. However, FCM mainly relies on pixel intensity and does not consider how pixels are connected. This can sometimes lead to less accurate results, especially when the image has noise or unclear boundaries. In contrast, **Fuzzy Connectedness** (FC) looks at both intensity and the relationship between pixels, making it better at capturing the actual shape and structure of

objects in the image. Studies have shown that FC is more reliable and produces more consistent results in tumor volume estimation.

5. VSCO:[5] VSCO used TensorFlow Lite to develop the “For This Photo” feature, which uses on-device machine learning to identify what kind of photo someone is editing and then suggest relevant presets from a curated list.
6. WPS:[6] WPS Office implements multiple business scenarios, such as on-device image recognition and image OCR based on TensorFlow Lite.

Chapter 3: System Analysis

Project specification

Stakeholders

Doctors:
Doctors are the primary users of the Breast Cancer Application. They will use the app to easily and accurately identify the tumor volume, helping them make informed decisions efficiently.

Medical Experts:
To ensure accuracy in tumor volume detection, we will seek feedback from specialist doctors and medical professionals throughout the development process. Their insights will help refine the application, improving its reliability and effectiveness for clinical use.

Project Sponsors:
We aim to collaborate with breast cancer institutions that can provide private datasets for training our model. Although these organizations may not directly fund the project, their data contributions are essential for developing an effective application.

Regulatory Authorities:

Regulatory authorities ensure that the application complies with healthcare and data privacy standards. This includes adhering to guidelines for medical software and protecting sensitive patient data.

Functional Requirements

1. **ROI Detection**
    - **Description:** The app shall automatically detect the Region of Interest (ROI) in an MRI scan.
    - **Details:**
        - The detection algorithm will identify areas containing potential tumor tissues.
        - The system will highlight the identified ROI for subsequent processing steps.
        - The detection process must handle different MRI scan formats and resolutions.
2. **Seed Point Localization**
    - **Description:** The app shall employ a Seed Detection Model to precisely locate seed points within the identified ROI.
    - **Details:**

- The model will analyze the ROI to determine initial points that represent key tumor regions.
- It must operate with high sensitivity to ensure that no potential tumor areas are overlooked.
- The seed points shall be validated through predefined criteria to reduce false positives.

3. Tumor Volume Estimation

- **Description**: The app shall use a Fuzzy Logic System to estimate the tumor's volume based on the detected seed points.
- **Details**:
  - The fuzzy logic algorithm will incorporate uncertainty handling to provide robust volume estimations.
  - The system will process spatial data from the seed points to compute an accurate tumor volume.
  - Volume estimation must be provided in clinically relevant units (e.g., cubic centimeters).

4. **Results Display and Reporting**

- **Description:** The app shall display the tumor volume estimation and relevant analysis results to the healthcare provider.
- **Details:**
  - The user interface will present the tumor volume alongside a visual overlay on the MRI scan for context.
  - Additional metrics or visual aids (e.g., confidence intervals, segmentation boundaries) may be provided to assist in clinical decision-making.
  - The system will support exporting results in standard medical report formats (e.g., PDF, DICOM SR) for integration into patient records.

5. **Distributed Computation**

- **Description:** The system shall automatically discover worker devices and distribute computation tasks across the local network using gRPC communication protocol.
- **Details:**
  - Worker devices automatically register with the coordinator using mDNS service discovery

- Coordinator splits large computation tasks into smaller chunks based on available worker capacity
- Tasks are assigned to workers via gRPC remote procedure calls
- Workers execute assigned tasks independently using local machine learning models
- Results are returned to coordinator for final aggregation and combination

6. **Real-Time Monitoring**

- **Description:** The system shall provide real-time progress updates and worker status monitoring during distributed computation execution.
- **Details:**
    - Real-time event streaming shows task assignment, completion, and error status
    - Users can view live computation logs and performance metrics
    - Worker device status (online/offline) is displayed in real-time
    - System provides immediate feedback on computation milestones and completion

7. **Fault Tolerance**

- **Description:** The system shall detect worker failures and automatically redistribute failed tasks to available workers to ensure computation completion.
- **Details:**
    - Automatic detection of worker device failures or network disconnections
    - Failed tasks are automatically reassigned to other available workers
    - System maintains stability even when individual workers become unavailable
    - Retry mechanisms ensure task completion with alternative workers
    - Graceful degradation when worker count decreases during computation

Non-functional Requirements

1. **Performance & Computational Efficiency**
    - **Seed detection and preprocessing** should leverage multi-core CPU processing for faster execution.
    - The app should ensure **efficient thread management** to maximize CPU utilization without degrading overall system performance.
    - **GPU acceleration** should be used primarily for segmentation and volume estimation.

2. **Reliability & Stability**
    - The app should **dynamically adjust CPU core usage** based on available system resources.
    - If the device has limited CPU cores, the system should **gracefully degrade performance** without failure.
    - Ensure **consistent results** across different mobile chipsets (ARM, Snapdragon, etc.).

3. **Battery & Thermal Management**
    - **CPU-based tasks (seed detection, preprocessing)** should be optimized to minimize battery consumption.
    - Implement **adaptive processing** to reduce strain on lower-end devices while maintaining accuracy.

4. **Security & Data Privacy**
    - Implement **access control mechanisms** to restrict unauthorized usage.
    - Data is used **on-premises**

5. **Scalability & Maintainability**
    - The software architecture should allow **easy updates and enhancements**, such as integrating improved AI models.
    - It should support **different MRI machines and formats** with minimal configuration.
    - The system should allow for future expansion to detect other tumor types.

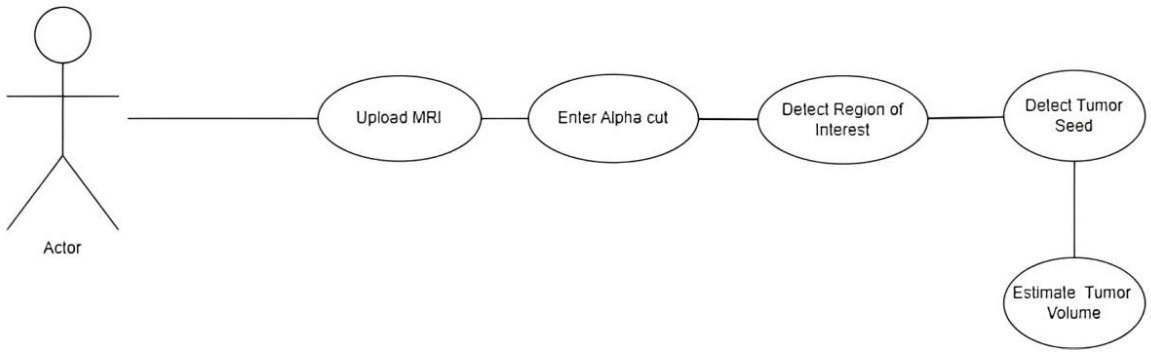
6. **Usability & Accessibility**
    - **Intuitive UI & Touch Optimization** – The app should have a user-friendly interface requiring minimal training, with smooth touch-based interactions (e.g., swipe, pinch-to-zoom).

- **Clear Tumor Visualization** – Provide high-quality segmentation display with zoom, pan, and annotation features to aid diagnosis.
- **Accessibility Features** – Support responsive UI, large tap targets, dark mode, and high-contrast themes for better usability in medical environments.

7. Portability & Compatibility

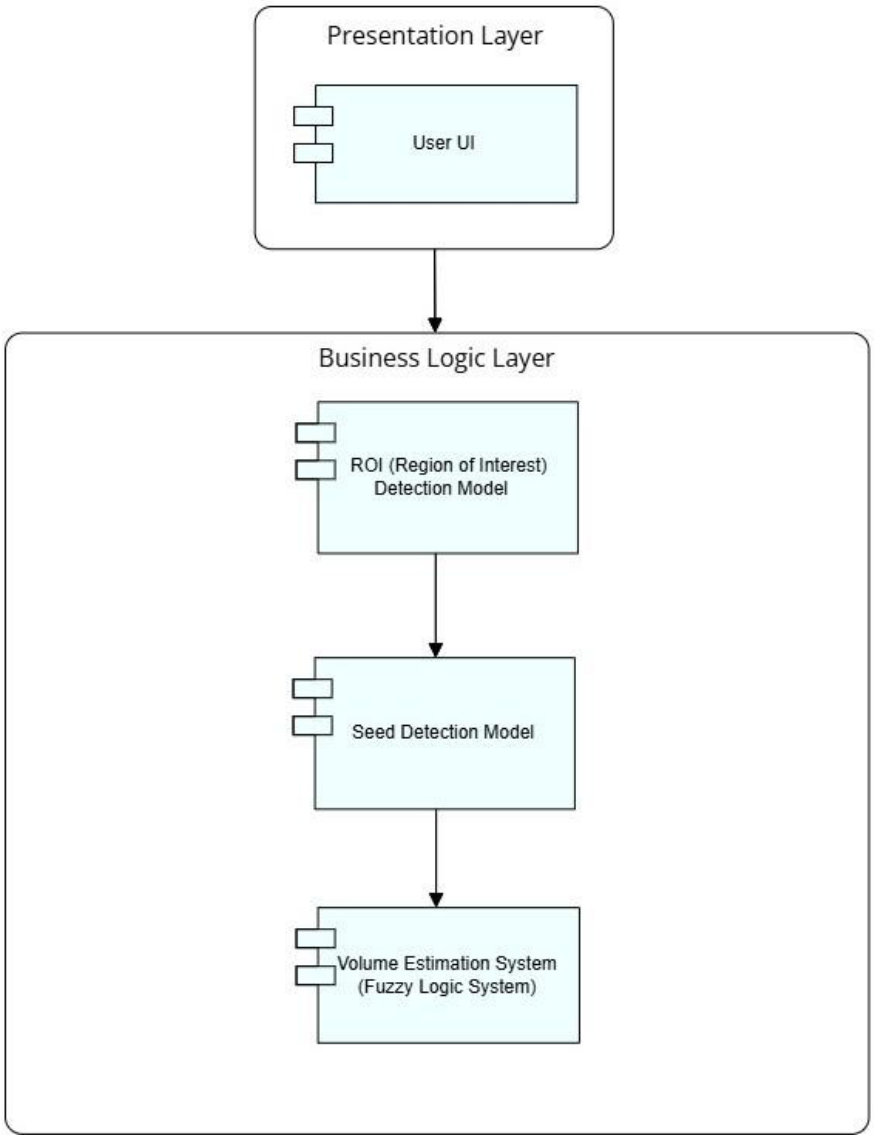
- **Android Compatibility** – The app should support a wide range of Android devices, from mid-range to high-end models.
- **Adaptive UI** – Ensure proper scaling across different screen sizes (phones, tablets) without UI distortion.
- **Efficient Storage & Performance** – Maintain a low storage footprint and optimize performance for devices with limited memory.

Use Case Diagram

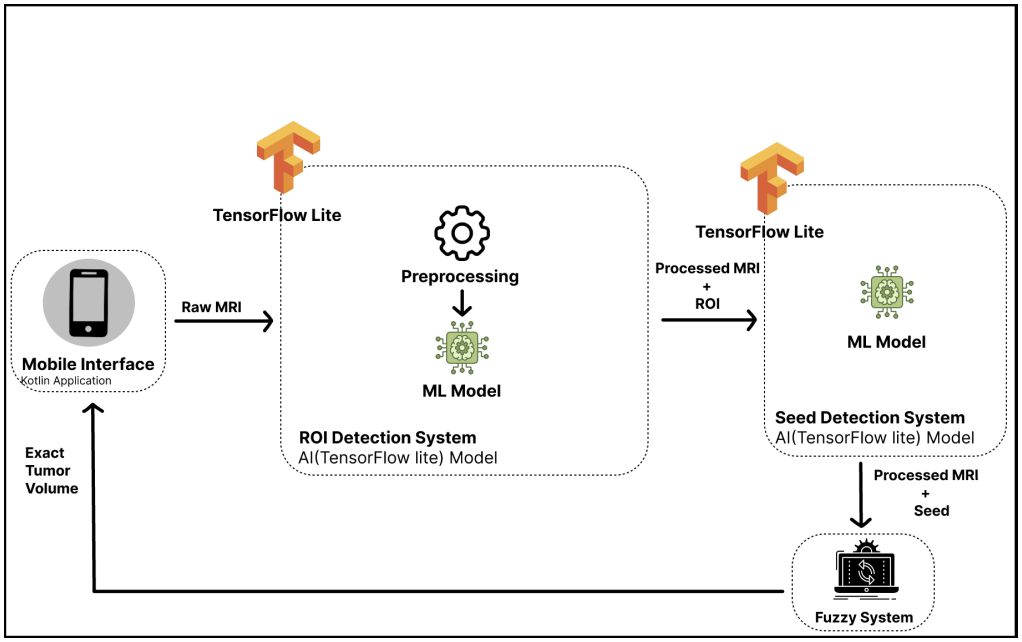


Chapter 4: System Design

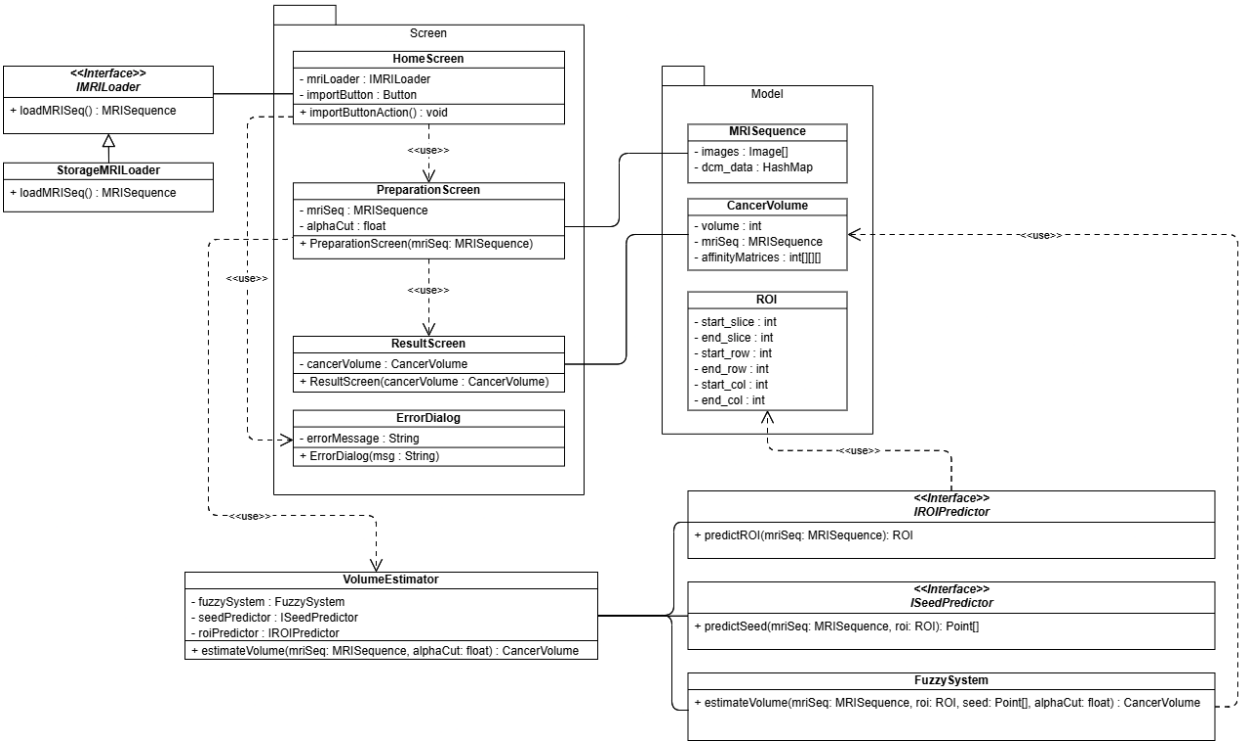
System Architecture



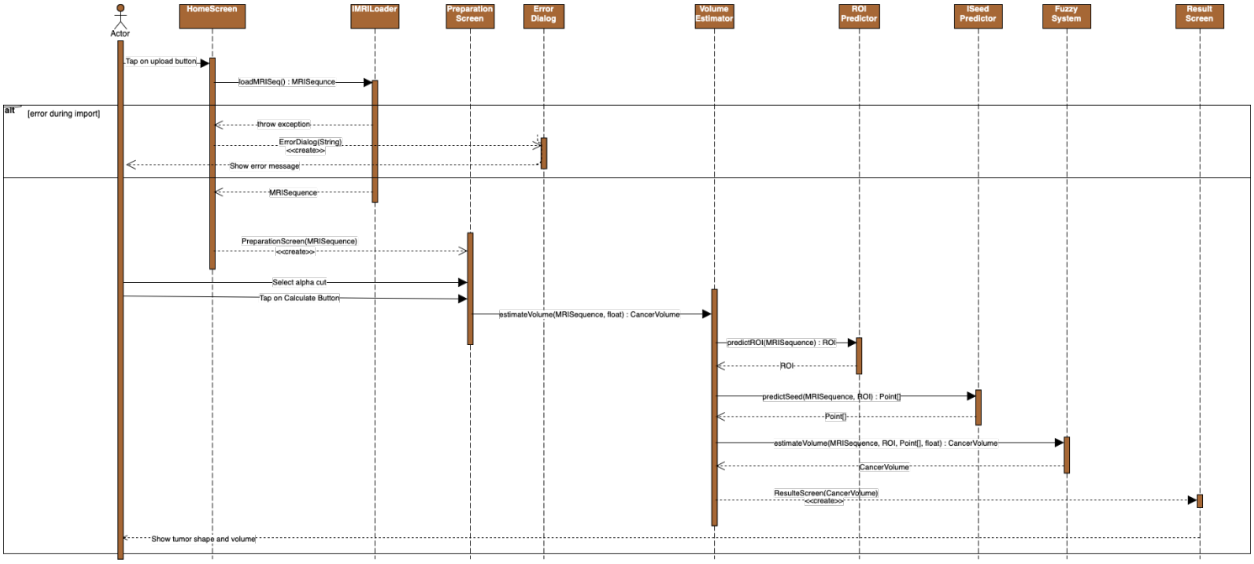
System Component Diagram



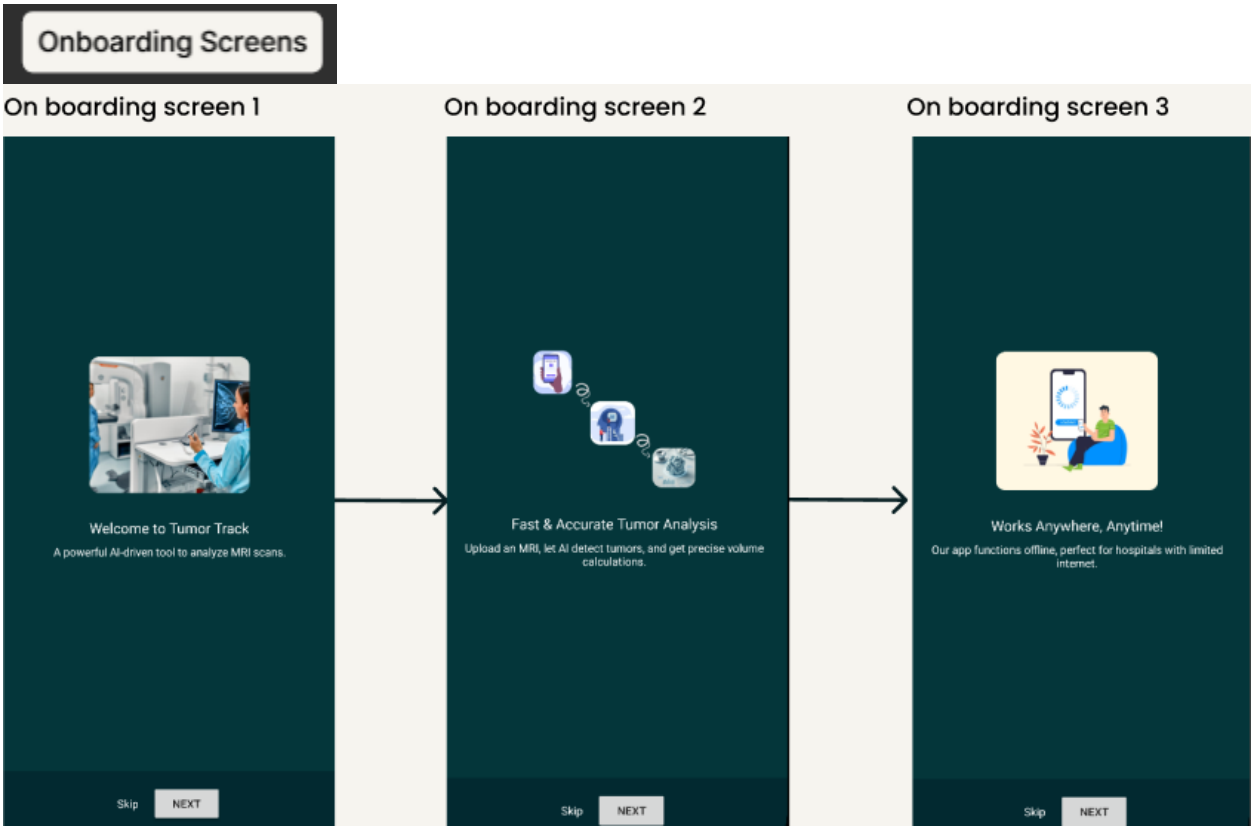
Class Diagram

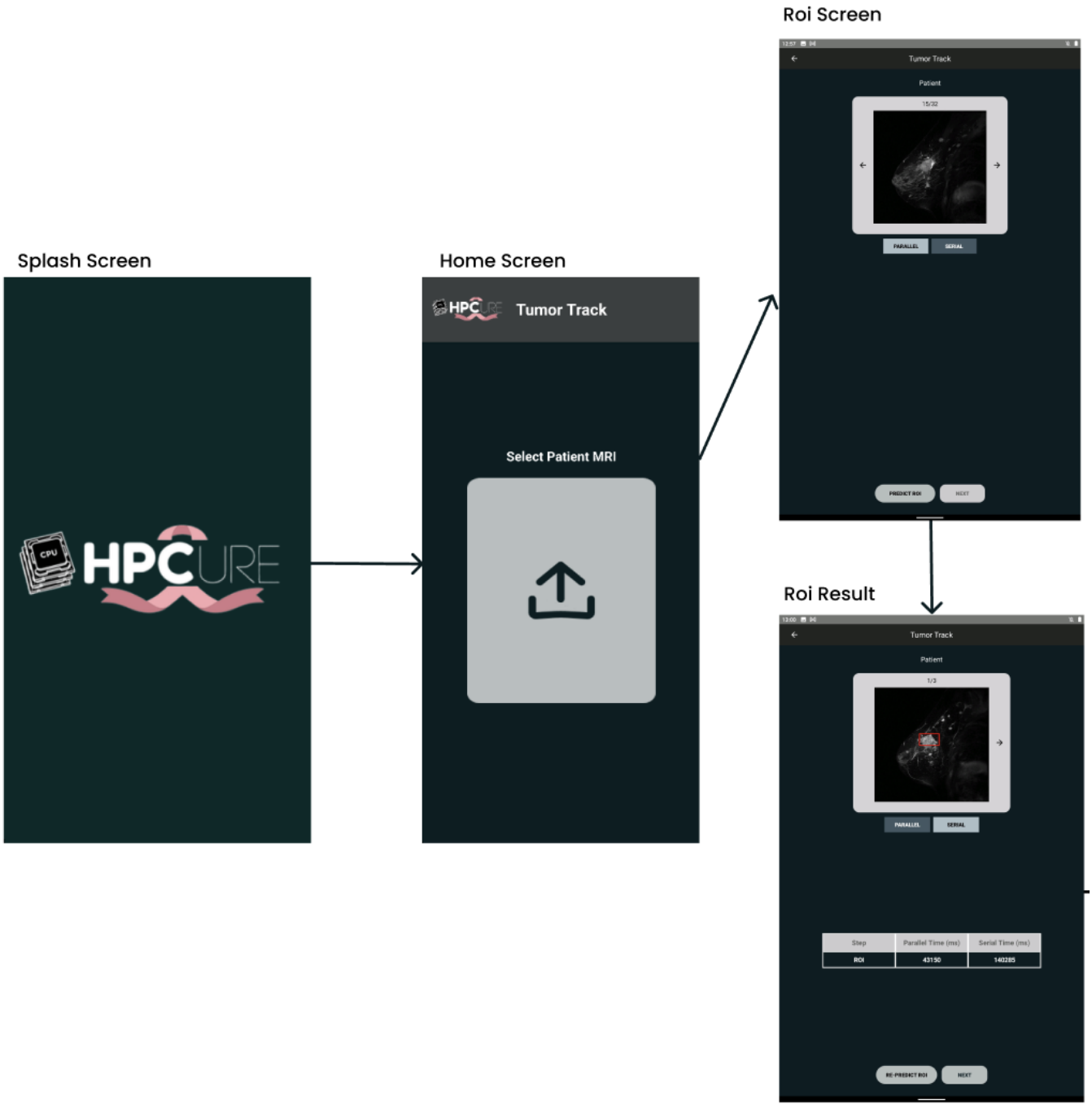


Sequence Diagram

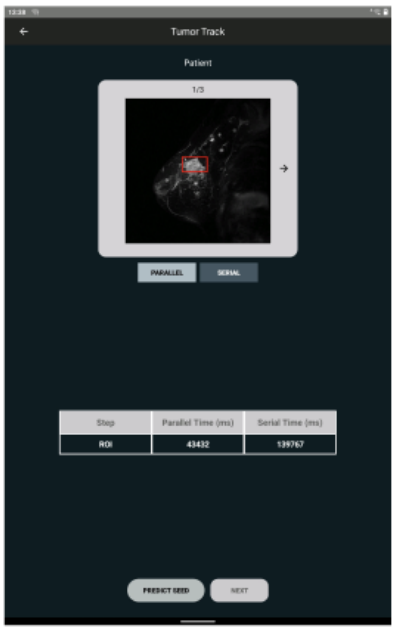


System GUI Design

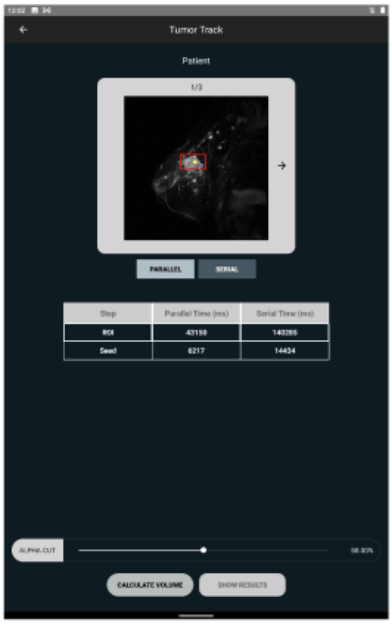



Roi Screen
Tumor Track
Patient
15/32
PARALLEL
SERIAL
PREDICT ROI
NEXT
Splash Screen
CPU
HPCURE
Home Screen
HPCURE
Tumor Track
Select Patient MRI
Roi Result
Tumor Track
Patient
1/3
PARALLEL
SERIAL
Step
Parallel Time (ms)
Serial Time (ms)
ROI
43150
140285
RE-PREDICT ROI
NEXT

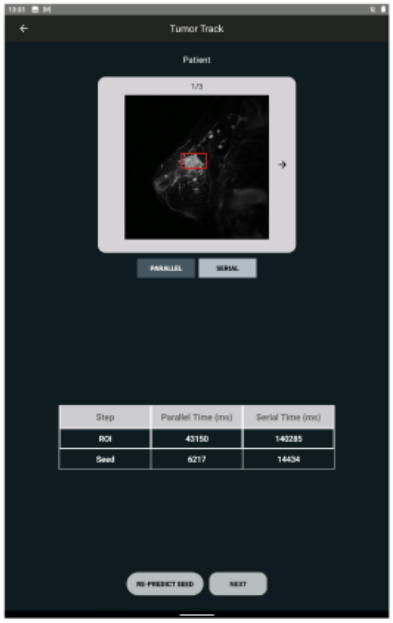
Seed Screen



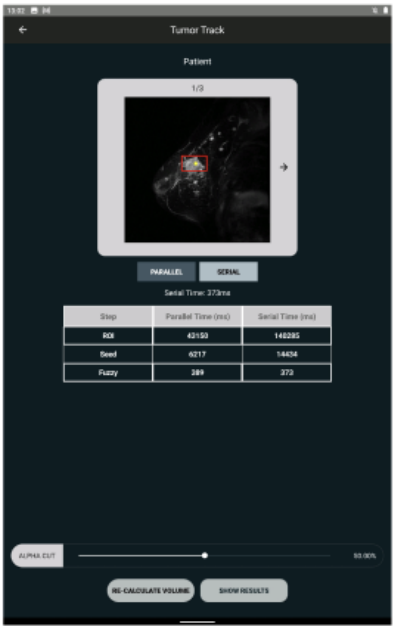
Fuzzy Screen



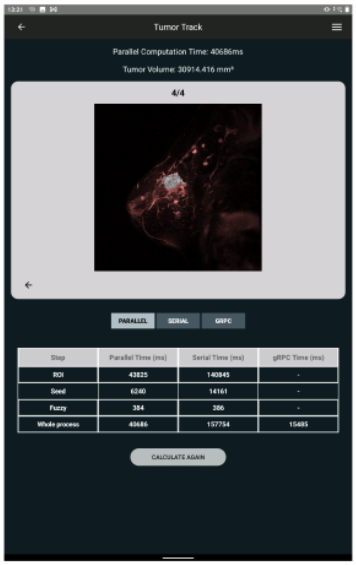
Seed Result



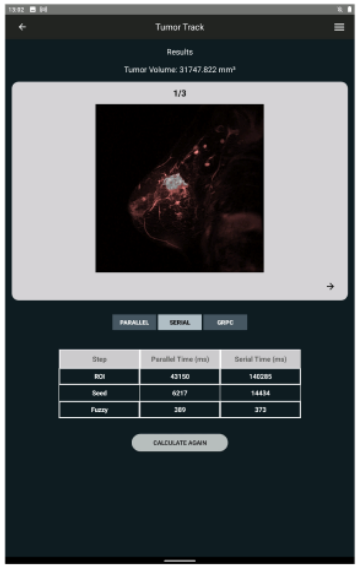
Fuzzy Screen Result



Result Screen (whole process)

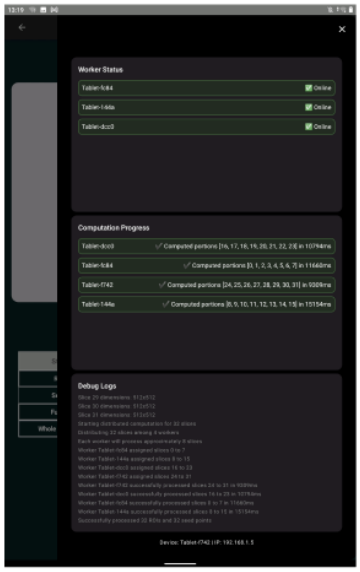


Result Screen



Grpc processing

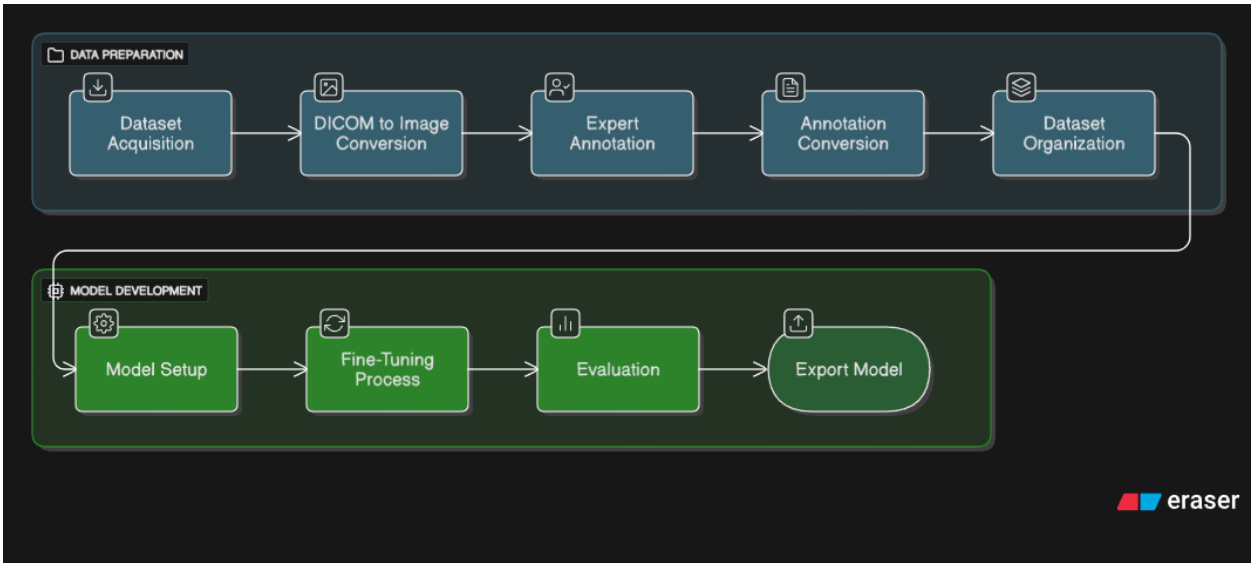
Grpc Computed



Chapter 5: Implementation and Testing

- Model Development

To clarify the structure and sequence of the model development process, the following diagram presents the complete training pipeline for both the **ROI detection** and **seed point detection** models. It illustrates the major stages—from dataset acquisition and preprocessing, to expert annotation, YOLO-format conversion, model fine-tuning, and final deployment in **TensorFlow Lite** format. This overview provides context for the more detailed descriptions that follow in the subsequent sections.



The core objective of this project is to estimate the **tumor volume** in breast MRI scans by leveraging a combination of deep learning models and a **fuzzy logic system**. The training phase was divided into two key model components that serve as inputs to the fuzzy estimation stage: (1) a **Region of Interest (ROI) detection model** and (2) a **seed point detection mechanism**. Each of these components plays a critical role in preparing the input required for accurate and interpretable tumor volume estimation.

The general flow of the system is as follows:

1. **Region of Interest Detection**: Identifies the bounding box that likely contains a tumor. This box is defined by coordinates (`x_min`, `x_max`, `y_min`, `y_max`) on the MRI slice.

2. **Seed Point Detection**: Determines a point within the tumor (the seed point), located inside the detected ROI.

3. **Fuzzy Volume Estimation**: The bounding box (ROI) and seed point are used as inputs to a fuzzy inference system, which calculates the estimated tumor volume based on spatial, geometric, and contextual rules.

Dataset Source and Preprocessing

The training dataset was obtained from the Advanced MRI Breast Lesions collection, which is publicly available through The Cancer Imaging Archive (TCIA). This collection provides a comprehensive set of high-resolution DICOM-format breast MRI scans, which include studies acquired in the sagittal plane—a crucial factor for this project.

The sagittal imaging position was specifically chosen because it offers a clearer anatomical view of breast tissue distribution and tumor morphology across slices. This orientation is particularly suitable for region localization and volumetric estimation, making the dataset well-aligned with the goals of this project.

To prepare the data for model training:

- DICOM files were converted into standard image formats (such as PNG) after preprocessing steps, including intensity normalization and contrast enhancement, to improve model interpretability.

- Tumor regions were manually annotated by Dr. Rana from the National Cancer Institute, ensuring clinically accurate bounding boxes for each MRI slice.

- The resulting annotations were transformed into the YOLO format, which encodes bounding boxes using normalized coordinates along with corresponding class labels.

This prepared dataset enabled the model to be trained effectively on sagittal breast MRI slices, contributing to accurate tumor localization in the later stages of the **system.**

1. Region of Interest (ROI) Detection Model:

The Region of Interest (ROI) detection model was implemented using YOLOv8s, a real-time object detection architecture known for its balance between high accuracy and computational efficiency. This model was selected to ensure reliable detection performance while maintaining compatibility with deployment on resource-constrained systems.

a. Model Fine-Tuning and Training

The **YOLOv8s model** was fine-tuned on the annotated breast MRI dataset using a **transfer learning** strategy. This approach involves adapting a pre-trained model—originally trained on a large-scale dataset such as COCO—to a domain-specific task with limited data. In this case, the goal was to detect tumor regions in sagittal breast MRI slices.

The fine-tuning process utilized the previously annotated dataset, which had been formatted into YOLO-compatible structure. Images were resized to **512 × 512 pixels** during training to ensure consistency while preserving sufficient spatial resolution necessary for detecting medical features.

---

b. Training Configuration and Approach

Model Initialization: Training began with pre-trained weights from the YOLOv8s model, which provided a strong feature extraction backbone.

Epochs: The model was trained over **25 epochs**, a duration chosen based on early experimentation to ensure convergence while avoiding overfitting.

Image Size: All images were resized to **512 × 512** , balancing the need for spatial detail with computational efficiency.

Batching and Optimization: The training process used standard YOLO optimization settings, including adaptive learning rates and stochastic gradient descent with momentum.

---

c. Evaluation Metrics

The performance of the ROI detection model was evaluated using standard object detection metrics to assess both accuracy and robustness. These metrics help quantify how effectively the model localizes tumor regions in sagittal breast MRI slices.

Key Metrics

- **Precision**
  Measures the proportion of true tumor detections among all predicted bounding boxes. High precision indicates that most detected regions are indeed tumors, minimizing false positives.

- **Recall**
  Reflects the model's ability to identify all actual tumor regions. A high recall value means the model successfully detects the majority of tumors, minimizing false negatives.

- **Mean Average Precision (mAP)**
  A comprehensive metric that combines both precision and recall:

- **mAP@0.5** evaluates accuracy at a single Intersection over Union (IoU) threshold of 0.5.

- **mAP@0.5:0.95** averages performance across IoU thresholds from 0.5 to 0.95, providing a stricter and more realistic assessment.

Model Performance Summary

The YOLOv8s model demonstrated strong performance across all metrics:

- **Precision** and **recall** improved steadily throughout training, indicating that the model learned to accurately and consistently identify tumor regions.

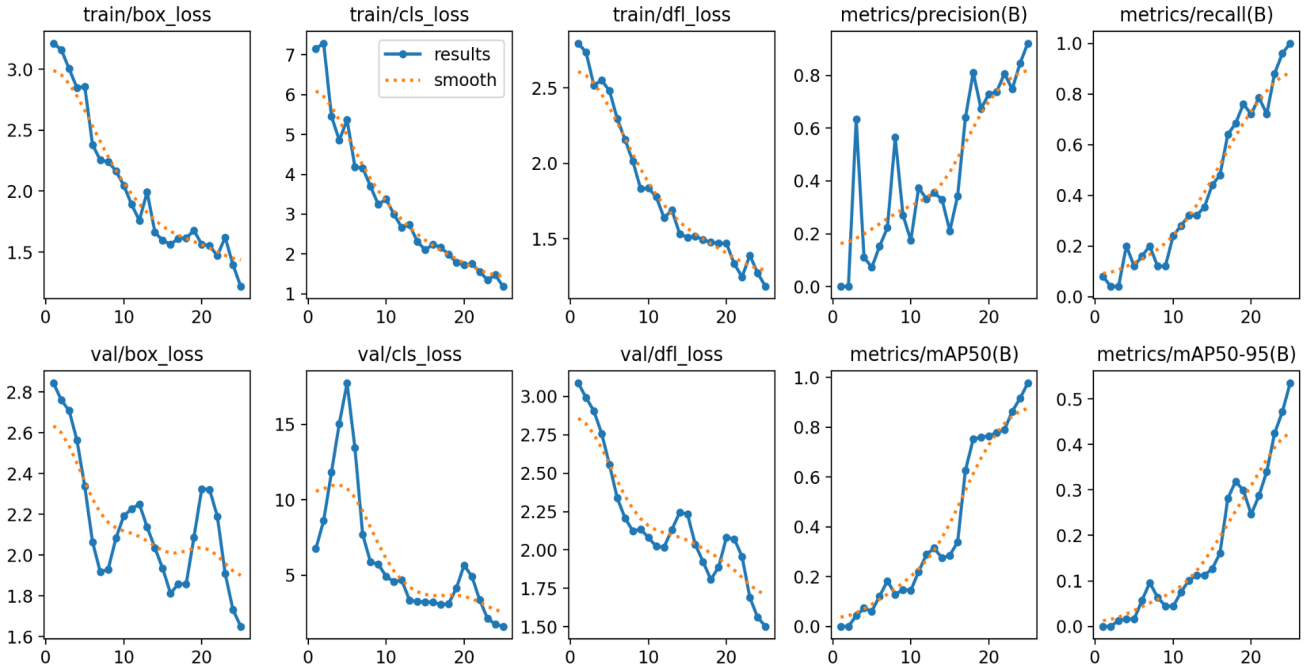
- **mAP@0.5** reached approximately **0.95**, suggesting excellent bounding box alignment at moderate overlap thresholds.

- **mAP@0.5:0.95** reached about **0.55**, demonstrating strong generalization even at stricter evaluation standards.

These results confirm the model's reliability and robustness across diverse MRI slices, making it a strong foundation for the tumor volume estimation pipeline.

Training Curve Visualization

The figure below shows the evolution of the loss functions and performance metrics over the 25 training epochs



d. Validation Results and Visualization

To qualitatively evaluate the performance of the ROI detection model, we selected representative validation images from the dataset. For each example, we present a side-by-side comparison between the ground truth annotations (provided by a domain expert) and the model's predictions.

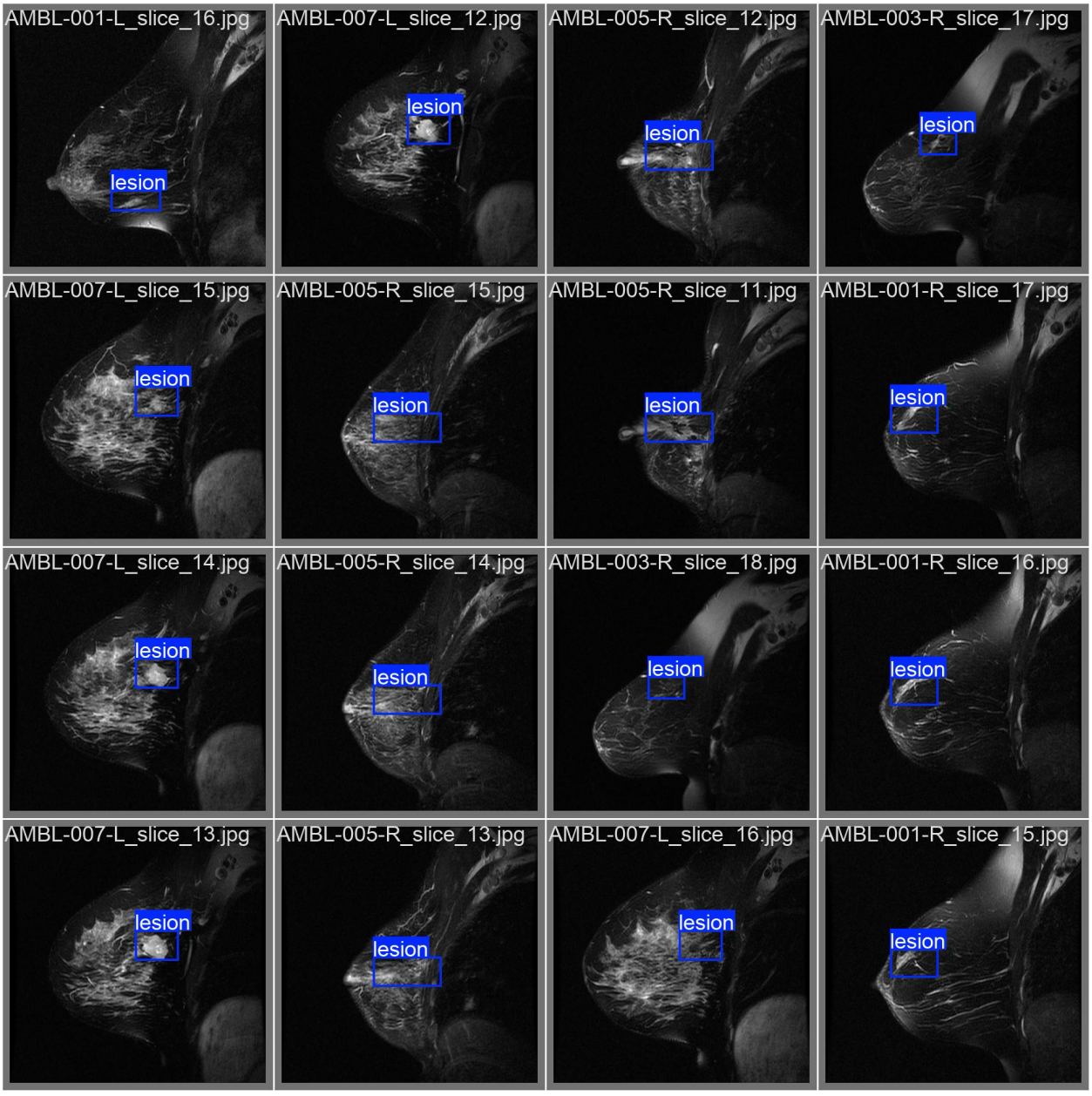
This visual comparison helps to demonstrate how accurately the YOLOv8s model localizes tumor regions in sagittal breast MRI slices after fine-tuning.

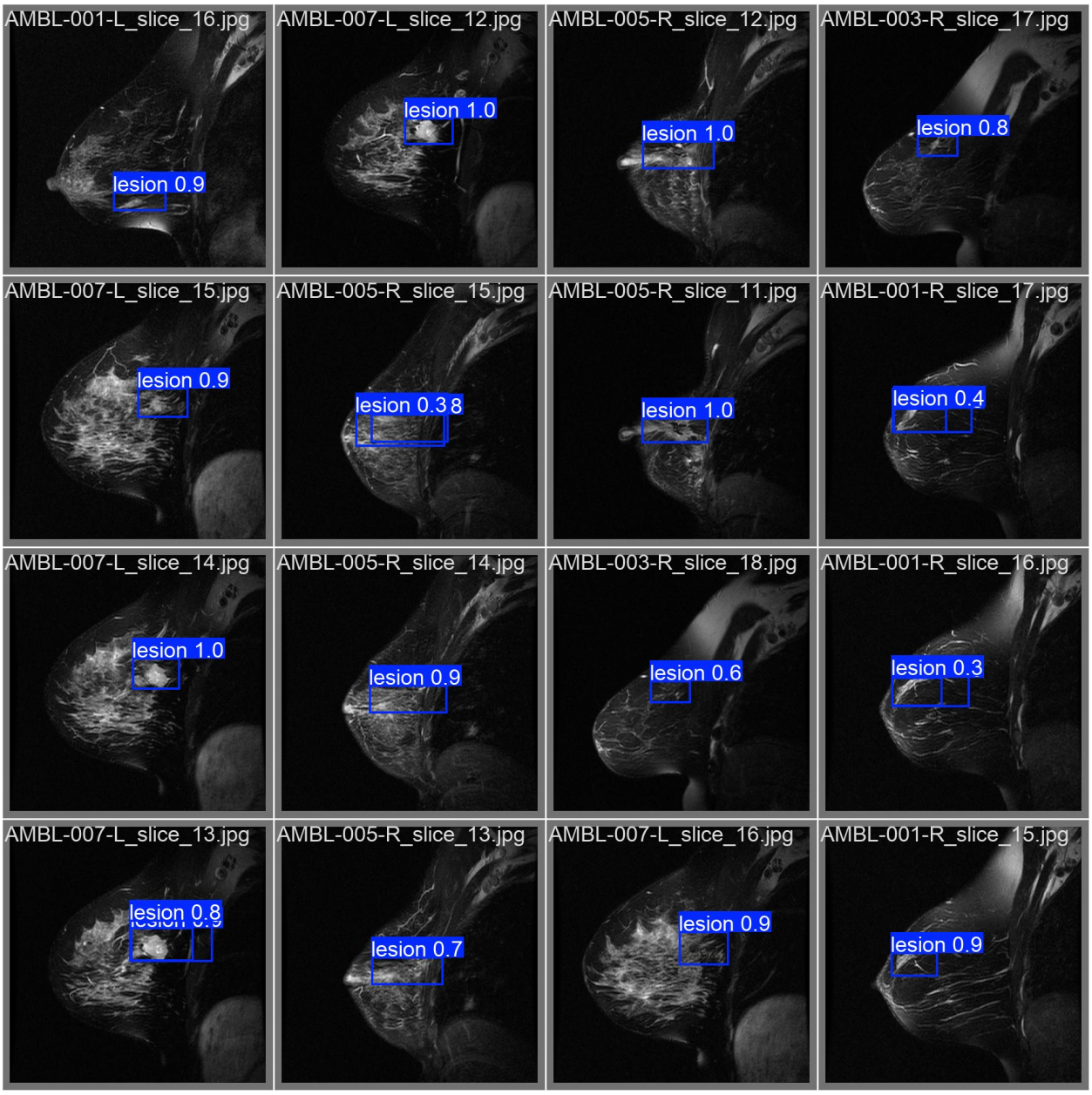
Ground Truth Labels

The following image shows expert-annotated ground truth bounding boxes for selected validation MRI slices. These precise labels served as the reference standard for both model training and evaluation.

Model Predictions

The following image displays predicted bounding boxes and corresponding confidence scores produced by the YOLOv8s model. The strong alignment with expert annotations highlights the model's effectiveness in accurately detecting tumor regions.




AMBL-001-L_slice_16.jpg
lesion 0.9
AMBL-007-L_slice_12.jpg
lesion 1.0
AMBL-005-R_slice_12.jpg
lesion 1.0
AMBL-003-R_slice_17.jpg
lesion 0.8
AMBL-007-L_slice_15.jpg
lesion 0.9
AMBL-005-R_slice_15.jpg
lesion 0.3 8
AMBL-005-R_slice_11.jpg
lesion 1.0
AMBL-001-R_slice_17.jpg
lesion 0.4
AMBL-007-L_slice_14.jpg
lesion 1.0
AMBL-005-R_slice_14.jpg
lesion 0.9
AMBL-003-R_slice_18.jpg
lesion 0.6
AMBL-001-R_slice_16.jpg
lesion 0.3
AMBL-007-L_slice_13.jpg
lesion 0.8
AMBL-005-R_slice_13.jpg
lesion 0.7
AMBL-007-L_slice_16.jpg
lesion 0.9
AMBL-001-R_slice_15.jpg
lesion 0.9

2. Seed Point Detection Model:

The same model used for Region of interest detection (YOLOv8s) was used for Seed point detection for the same reasons. But the Pose task was used to involve identifying of a specific point.

a. Model Fine-Tuning and Training

To fine tune the model on a relatively small data and still get a good results, the slices containing tumor was cropped to only the region of interest and then scaled to **512 × 512**, also this crop allowed us to use more annotated data in a different MRI position (axial position) since after the crop the MRI position becomes less significant to the view.

b. Training Configuration and Approach

Model Initialization: Training began with pre-trained weights from the YOLOv8s model.

Epochs: The model was trained over **100 epochs**, since the pre-trained was designed to identify keypoints of human body which is not very relevant to the seed detection use case.

Hardware: Since only manageable size of annotated data was used the training was possible on a personal machine, but also **100 epochs** is relatively big load on the CPU so it was made sure that also GPU was utilized (GTX 1650) yielding in 2 hours of the final training.

Image Size: All images were resized to **512 × 512** , balancing the need for spatial detail with computational efficiency.

Batching and Optimization: The training process used standard YOLO optimization settings, including adaptive learning rates and stochastic gradient descent with momentum.

c. Evaluation Metrics

The seed prediction model, fine-tuned using YOLOv8s in pose estimation mode, was evaluated with precision-focused metrics to assess how accurately and consistently it identifies tumor seed points in sagittal and axial breast MRI slices.

Key Metrics

- **Precision**

  Indicates the percentage of correctly predicted seed points among all predictions. The model consistently maintained high precision, ensuring most detected seeds were accurate and minimizing false positives.

- **Recall**

  Measures the model's ability to detect all actual seed points. High recall values reflect robust detection, minimizing the risk of missed tumor seeds.

- **Mean Average Precision (mAP)**

  A balanced metric that considers both precision and recall across thresholds:

- **mAP@0.5 (Pose):**

  Assesses accuracy at a 0.5 IoU threshold. The model reached ~0.995, suggesting highly accurate seed localization at moderate overlap.\

- **mAP@0.5:0.95 (Pose)**

  Evaluates performance across stricter IoU thresholds. The model maintained ~0.995, reflecting exceptional robustness even under rigorous evaluation standards.

Model Performance Summary

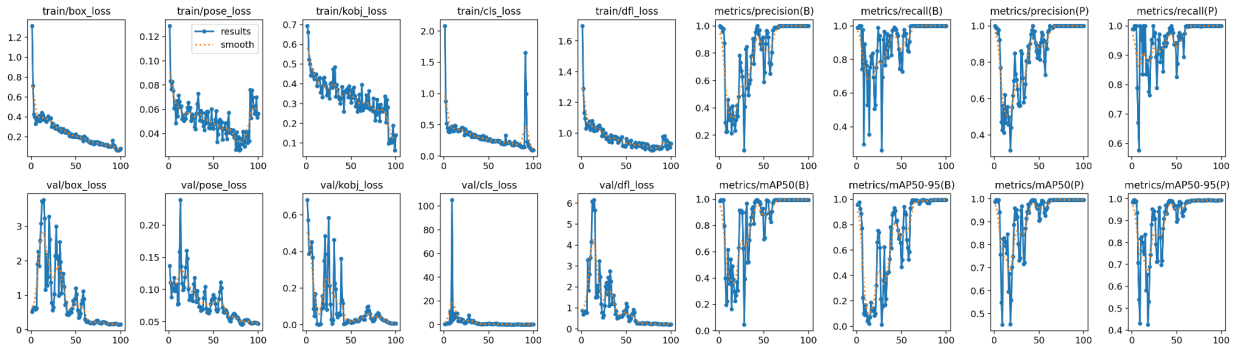
The YOLOv8s pose-based model demonstrated strong and stable performance across all evaluation metrics:

- **Precision remained consistently above 0.98 across training epochs.**
- **Recall reached 1.00 early in training, ensuring comprehensive seed detection.**
- **mAP@0.5 and mAP@0.5:0.95 for pose predictions both stabilized around 0.995, indicating nearly perfect localization accuracy even under tight IoU thresholds.**

These results validate the model's high reliability and precision in detecting tumor seeds, making it a suitable component in the broader tumor volume estimation and segmentation pipeline.

Training Curve Visualization

The figure below shows the evolution of the loss functions and performance metrics over the 25 training epochs**.**



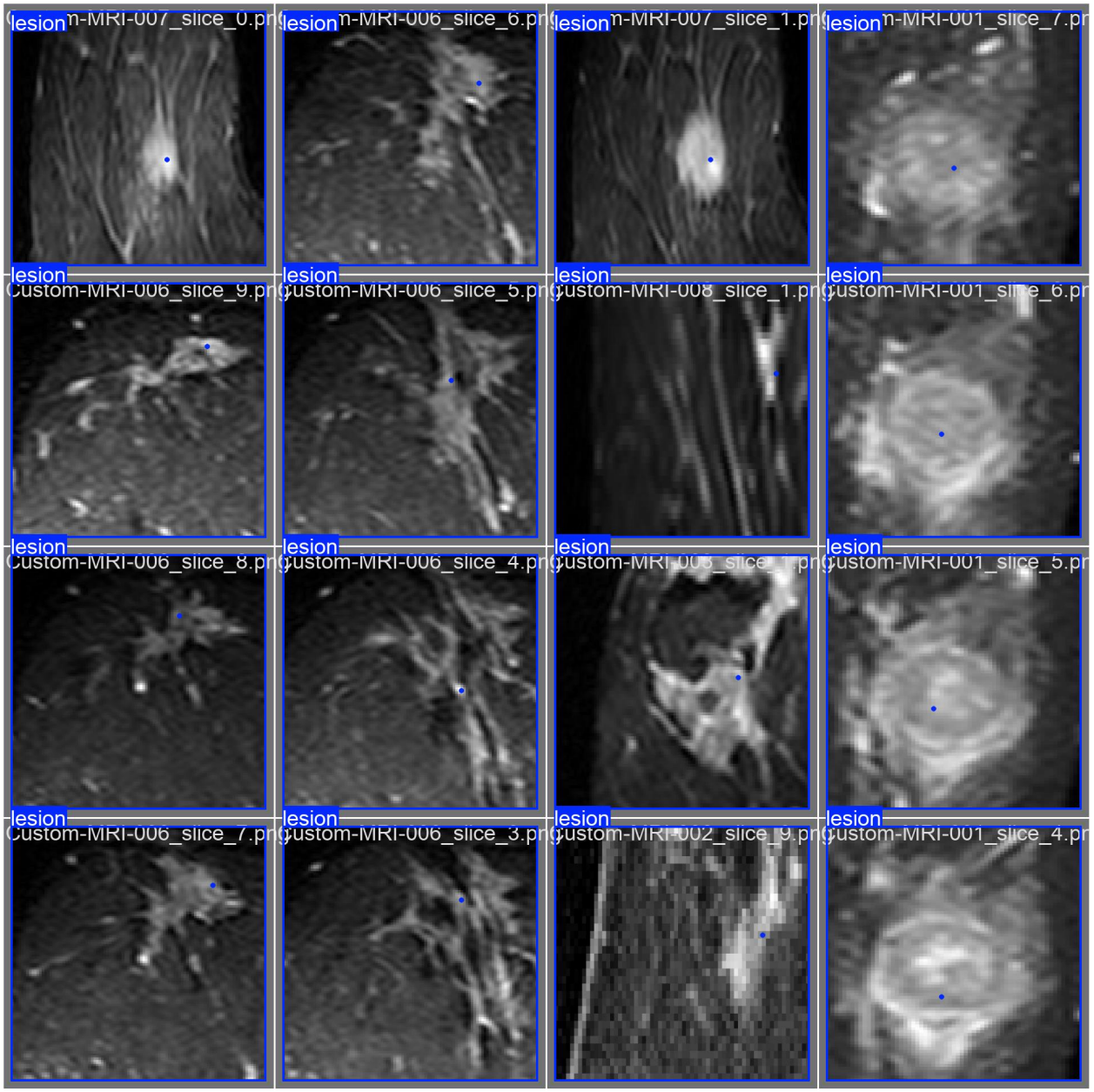
d. Validation Results and Visualization

To qualitatively evaluate the performance of the Seed detection model, we selected representative validation images from the dataset. For each example, we present a side-by-side comparison between the ground truth annotations (provided by a domain expert) and the model's predictions.

This visual comparison helps to demonstrate how accurately the YOLOv8s model predicts the seed point of the tumor in the cropped sagittal and axial breast MRI slices after fine-tuning.

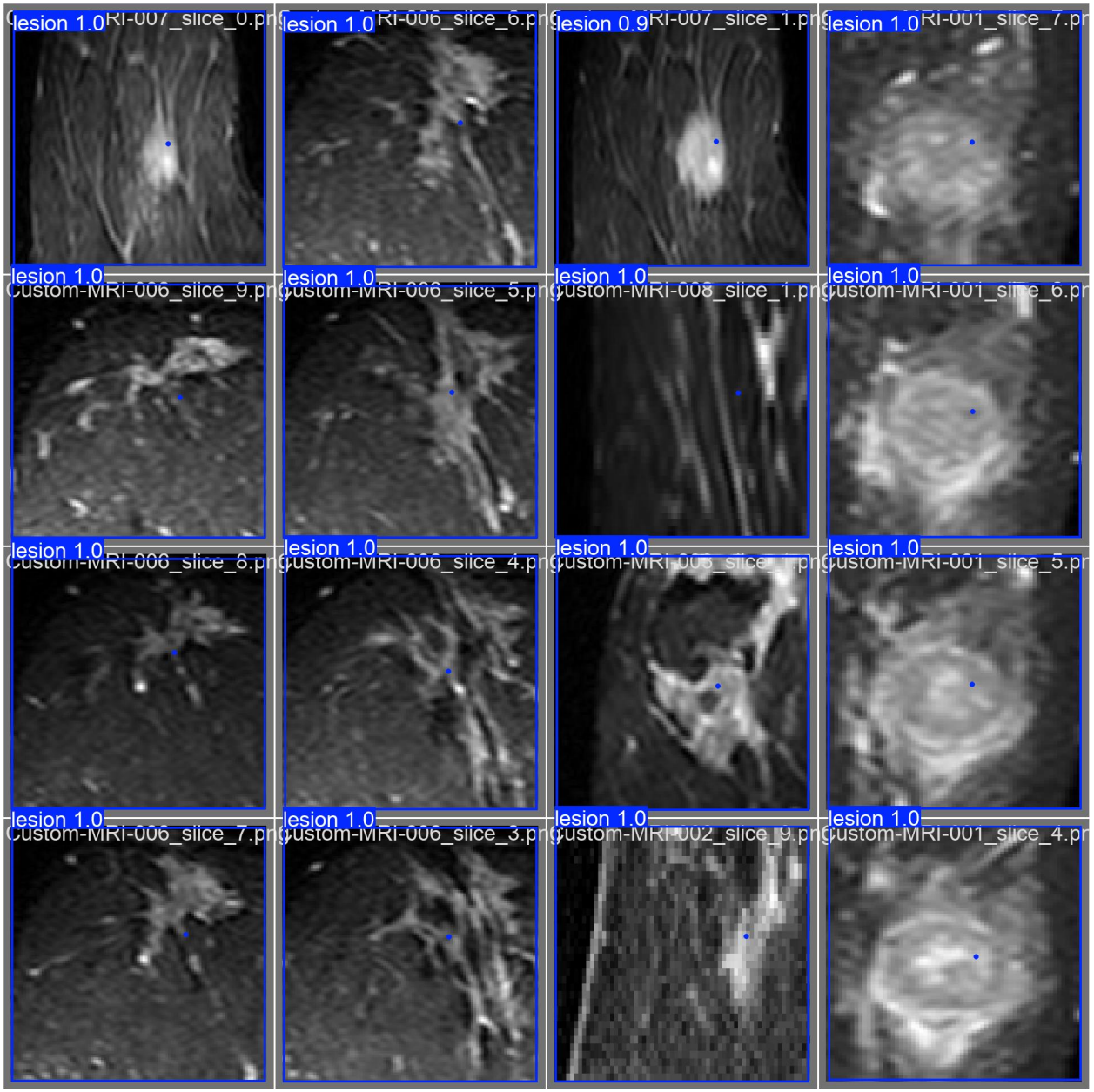
Ground Truth Labels

The following image shows expert-annotated ground truth seed points of tumor for selected validation MRI slices. These precise labels served as the reference standard for both model training and evaluation.



Model Predictions

The following image displays predicted tumor seed points produced by the YOLOv8s model. Since most of the points already lies on the tumor we can consider the model effective.



Integration of TensorFlow Lite for Models

Overview

This document describes the integration of two TensorFlow Lite models into a Kotlin Android application for medical image analysis. The system consists of:

ROI Predictor Model: Identifies regions of interest (ROIs) in MRI breast scans

Seed Predictor Model: Locates specific seed points within the identified ROIs

The implementation provides both sequential and parallel execution strategies to accommodate different performance requirements and device capabilities.

Model Integration Architecture

1. **ROI Predictor Implementation**

Purpose: Processes MRI sequences to detect regions of interest in breast scans

Key Components:

SequentialRoiPredictor: Processes slices one at a time

Loads the model once and reuses it for all slices

Simple error handling

Suitable for low-end devices

ParallelRoiPredictor: Processes slices concurrently

Uses coroutines with limited parallelism (MAX_PARALLEL_REQUESTS = 4)

Implements interpreter pooling

Includes memory-aware bitmap processing

Automatic GPU delegate fallback to CPU

Input/Output:

Input: 512x512 RGB normalized bitmap (CHW format)

Output: ROI coordinates (xMin, xMax, yMin, yMax) with confidence score

2. **Seed Predictor Implementation**
Purpose: Locates precise seed points within the ROIs identified by the first model

Key Components:

SequentialSeedPredictor:

Processes Seed points sequentially

Basic error handling with default fallback values

Simple model loading strategy

ParallelSeedPredictor:

Processes slices in parallel with resource constraints

Validates input dimensions

Implements bitmap recycling

Uses efficient coordinate rescaling

Includes comprehensive error handling

Input/Output:

Input: Cropped ROI region resized to 512x512

Output: (x,y) coordinates of seed point in original image space

Performance Considerations
Resource Management
Memory Efficiency:

- Bitmaps are recycled after processing
- Configurations are standardized to ARGB_8888
- Large buffers are allocated directly

Compute Optimization:

- GPU delegate used when available
- Thread count constrained (1-4 threads)
- XNNPACK disabled for stability

Parallel Processing

- Semaphore limits concurrent operations (4 max)
- Coroutine dispatchers separate IO from computation
- Model interpreters are pooled to reduce loading overhead

Usage Recommendations

For low-end devices:

Use sequential implementations

Set numThreads = 1

Disable GPU delegate if unstable

For high-end devices:

Use parallel implementations

Enable GPU delegate

Set numThreads = 4

Monitor memory usage with large sequences

Debugging:

Check "ModelDetails" and "[Predictor]" log tags

Validate intermediate coordinate transformations

Profile memory usage during batch processing

Interpreter Initialization and Configuration

To optimize performance and reduce load times, both the ROI and Seed Predictor models are initialized once and reused throughout the application's lifecycle. Interpreters are configured with adjustable thread counts and can optionally leverage the GPU delegate for acceleration on supported devices.

Key configuration parameters include:

- Number of threads allocated for inference
- Use of hardware acceleration (GPU delegate)
- Stability-focused disabling of unnecessary delegates like XNNPACK

These settings allow dynamic adaptation to a device’s capabilities, enhancing reliability on low-end devices while maximizing throughput on high-end devices.

Interpreter Pooling (Parallel Execution)

In the parallel predictor implementations, a dedicated interpreter pooling mechanism ensures safe and efficient concurrent access to the TensorFlow Lite model. A fixed pool size is maintained based on a defined maximum parallel request threshold. This strategy avoids reloading models for each inference, significantly improving throughput and lowering memory churn.

The pooling system guarantees:

- Thread-safe access to multiple interpreter instances
- Load balancing across coroutines

- Controlled resource allocation to prevent overconsumption

Delegate Fallback Strategy

In scenarios where the GPU delegate is unavailable or unstable, the system gracefully falls back to CPU execution. This ensures the application remains functional across a broad range of Android hardware, prioritizing stability and compatibility.

Log messages clearly indicate whether GPU or CPU execution is used, aiding in debugging and performance profiling.

Coordinate Mapping and Image Transformation

ROI Scaling

The ROI Predictor operates on normalized, resized images (typically 512x512 pixels). After inference, the predicted bounding boxes must be translated back into the coordinate space of the original MRI slice. This process involves scaling the model's output using ratios derived from the original and resized image dimensions.

Maintaining spatial accuracy in this transformation is critical, particularly for medical applications where pixel-level precision may influence diagnostic outcomes.

Seed Point Mapping

Once an ROI is cropped and resized, the Seed Predictor identifies a specific point within that region. The predicted seed coordinates, initially relative to the cropped 512x512 image, must be accurately mapped back to the original MRI image.

This is achieved through a two-step transformation:

1. Rescaling the seed coordinates to the ROI’s actual size
2. Offsetting by the position of the ROI within the full image

These calculations ensure that the seed location aligns with the anatomical region identified by the original scan, preserving clinical relevance.

Conclusion

This integration provides flexible implementations suitable for various Android devices while maintaining consistent model accuracy. The parallel implementations offer significant performance benefits on capable hardware, while the sequential versions ensure reliable operation on resource-constrained devices. The architecture allows for easy swapping of implementations based on runtime device capabilities.

- ## Fuzzy System

- ## Origin and Initial Implementation

The fuzzy connectedness system in this project was initially based on the methodology described in Dr. Ahmed Shawqy's research paper (A fuzzy tumor volume estimation approach based on fuzzy segmentation of MR images). The core idea is to segment images by evaluating the "fuzzy connectedness" between pixels, which quantifies how strongly pixels are connected based on intensity and spatial relationships.

The first implementation was inspired by the open-source Python code available at FuzzyConnectedness.py, line 78. This code provided a reference for the main algorithm, which used a breadth-first search (BFS) approach to propagate fuzzy affinity values from seed points throughout the image.

- ## Transition to Dijkstra's Algorithm

To improve efficiency and accuracy, the BFS approach was replaced with Dijkstra's algorithm. While BFS treats all paths equally, Dijkstra's algorithm prioritizes paths with higher affinity, ensuring that the strongest connections are always considered first. This change allows the fuzzy connectedness computation to more accurately reflect the underlying image structure, especially in cases with varying intensity gradients.

- ## The main modifications included:
    - Replacing the queue-based BFS with a priority queue (min-heap) for Dijkstra's traversal.
    - Updating the propagation logic to always expand the most strongly connected pixel next.
    - Ensuring that affinity values are updated only if a stronger connection is found.

- **Strategy Pattern: Sequential and Parallel Execution**

  To make the system flexible and scalable, a wrapper was developed using the strategy pattern. This design allows the fuzzy connectedness algorithm to be executed in different modes:

  - **Sequential Strategy:** Processes the image in a single thread, suitable for smaller images or environments without parallel processing capabilities.
  - **Parallel Strategy:** Divides the image into regions and processes them concurrently, leveraging multi-core CPUs for faster computation on large images.

  The strategy pattern enables easy switching between these execution modes without changing the core algorithm logic. The wrapper exposes a unified interface, and the desired strategy can be selected at runtime based on the context or user preference.

- **Summary**

  - **Initial version:** Direct translation of the research paper and reference Python code, using BFS.
  - **Algorithmic improvement:** Switched to Dijkstra's algorithm for more accurate and efficient propagation.
  - **Extensibility:** Introduced the strategy pattern to support both sequential and parallel execution, improving performance and maintainability.

  This evolution demonstrates a careful balance between fidelity to the original research, practical performance considerations, and modern software design principles.

- Implementation of UI in Kotlin

1. Introduction

The application follows a step-by-step workflow, guiding users from uploading DICOM images to visualizing tumor regions and calculating tumor volume. The process involves multiple screens, each serving a specific purpose in the tumor analysis pipeline.

The application consists of the following screens:

1. **Splash Screen** – Displays a loading animation while the app initializes.

2. **Onboarding Screens** – Introduces users to the app's features.
3. **Upload Screen** – Allows users to select a directory containing DICOM files.
4. **ROI (Region of Interest) Screen** – Detects potential tumor regions in MRI slices.
5. **Seed Screen** – Predicts seed points inside the detected regions of interest for tumor segmentation.
6. **Fuzzy & Result Screen** – Computes tumor volume and displays results.

---

2. Splash Screen

The SplashActivity serves as the entry point of the application. It displays a brief loading animation (for 4 seconds) before redirecting users to the OnboardingActivity. This screen ensures that all necessary resources are loaded before the user interacts with the app.

Key Features:

- A full-screen logo or loading indicator.
- Automatically transitions to onboarding after a short delay.

---

3. Onboarding Screens

The OnboardingActivity introduces users to the app’s functionality through a series of slides. Each slide highlights a key feature:

1. **Welcome Screen**
    - **Title**: "Welcome to Tumor Track"
    - **Description**: Introduces the app as an AI-powered MRI analysis tool.
    - **Visual**: An illustrative image (e.g., a doctor using a tablet).
2. **Feature Explanation Screen**
    - **Title**: "Fast & Accurate Tumor Analysis"
    - **Description**: Explains how the app detects tumors and calculates volume.

    - **Visual**: An MRI scan with highlighted tumor regions.

3. **Offline Capability Screen**
    - **Title**: "Works Anywhere, Anytime!"
    - **Description**: Emphasizes that the app works offline, making it suitable for hospitals with limited internet.
    - **Visual**: A doctor using the app in a remote location.

User Interaction:

- **Next Button**: Moves to the next slide. On the last slide, it redirects to the UploadScreen.
- **Skip Button**: Immediately takes the user to the UploadScreen, bypassing the remaining slides.

---

4. Upload Screen

The UploadScreen is where users select a directory containing DICOM files for analysis.

Key Features:

- A clickable container that triggers a directory picker.
- A loading overlay that appears while processing files.
- Automatic transition to the Region of interest screen “Roi Screen” upon successful DICOM loading.

Workflow:

1. User Action: Taps the upload area.
2. System Action: Opens a file explorer (via `ACTION_OPEN_DOCUMENT_TREE`).
3. Validation:
    - Only directories with valid DICOM files are accepted.

- If no DICOM files are found, an error message appears.

4. Processing:
    - The app reads DICOM metadata and converts images to bitmaps.
    - Upon success, it proceeds to the region of interest screen “Roi Screen”.

Error Handling:

- Displays a toast message if storage permissions are denied.
- Shows an error dialog if the selected directory contains invalid files.

5. ROI (Region of Interest) Screen

The RoiScreen predicts and displays tumor regions (ROIs) in MRI slices, along with performance metrics.

Key Features:

- **Image Navigation:** Navigate MRI slices using “Prev” and “Next” buttons.
- **Mode Selection:**
    - Parallel Mode: Multi-threaded processing (faster).
    - Serial Mode: Sequential processing (simpler).
    - GPU Toggle: Optional GPU acceleration (if available).
- **Prediction Button:**
    - "Predict ROI" → Runs detection and updates the report table.
    - Changes to "Re-predict ROI" after the first run.
- **Report Table:** Displays the time taken for ROI prediction in both modes.
- **Navigation Buttons:**
    - **Skip (→):** Always enabled to skip directly to FuzzyAndResultScreen.
    - **Next:** Becomes active only after the ROI prediction to continue the workflow.

Workflow:

- The user selects the mode (Parallel/Serial).
- Clicks "Predict ROI".
- The system processes all slices for ROI prediction.
- Filters low-confidence results (score > 0.3).
- Displays red rectangles for detected regions.
- Update the timing data in the report table.
- The "Next" button becomes active after prediction.
- The "Skip" button is always available to go to SeedScreen.

---

6. Seed Screen

The SeedScreen predicts seed points (tumor centers) within the ROIs and measures performance.

Key Features:

- **ROI Continuity:** Inherits filtered ROIs from the previous screen.
- **Mode Selection:** Parallel/Serial modes with optional GPU toggle.
- **Seed Prediction:**
    - "Predict Seed" → Detects seed points (yellow circles).
    - Update the timing in the report table.
- **Navigation Buttons:**
    - **Skip (→):** Always enabled to skip directly to FuzzyAndResultScreen.
    - **Next:** Enabled after seed prediction.

Workflow:

- The user selects the processing mode.
- Clicks "Predict Seed".
- The system predicts seed points within detected ROIs.
- Displays yellow dots on the images.
- Update timing info in the report.
- The "Next" button becomes active after prediction.
- The "Skip" button is always available to go to the FuzzyAndResultScreen.

---

7. Fuzzy & Result Screen

The FuzzyAndResultScreen calculates tumor volume based on fuzzy logic and provides full-process recalculation support.

Key Features:

- **Two Layouts:**
    - Fuzzy Layout:
        - Alpha-cut slider (0–100%, default: 50%).
        - "Calculate Volume" triggers volume computation.
        - "Show Results" appears after computation.
    - Results Layout:
        - Displays tumor volume and highlighted regions.
        - "Recalculate" to re-run all steps.
- **Mode Selection:** Parallel, Serial, or GRPC mode for offloaded server processing.
- **Report Table:** Tracks ROI, Seed, Fuzzy timing, and whole-process duration.

Workflow:

- Loads MRI slices with ROIs and seeds.
- User adjusts alpha-cut value (optional).
- Clicks "Calculate Volume".
- The system runs fuzzy segmentation and volume computation.
- Enables "Show Results" for visualization.
- "Recalculate" re-executes ROI → Seed → Fuzzy flow in the selected mode.

---

8. Conclusion

Tumor Track delivers streamlined, performance-aware tumor analysis with:

- **Structured Flow:** Upload → ROI → Seed → Volume.
- **Performance Metrics:** Time tracking at each step.
- **Flexibility:**
    - Re-predict individual steps.
    - Re-run the full pipeline using "Recalculate".
- **User Navigation:**
    - The "Skip" button on the ROI and Seed screens allows fast-forwarding.
    - The "Next" button enables controlled step-by-step progression.
- **Visual Feedback:** Red rectangles (ROIs), yellow dots (seeds), and final volume overlays.
- **Processing Control:** Supports Parallel, Serial, and GRPC modes with GPU acceleration where available.

- ## gRPC for real-time distributed computation

  Overview

Our application leverages gRPC (Google Remote Procedure Call) to enable real-time, distributed computation across multiple devices in a local network. This architecture allows computationally intensive tasks—such as medical image analysis—to be split and processed in parallel by several worker devices, significantly reducing total computation time and enabling scalable, collaborative processing.

System Architecture

- **Coordinator Node:**

  Acts as the central controller. It receives computation requests, splits the workload, assigns tasks to available worker nodes, collects results, and aggregates them for the final output.

- **Worker Nodes:**

  Devices that register themselves with the coordinator and execute assigned computation tasks. Each worker processes a subset of the data and returns results to the coordinator.

- **gRPC Communication:**

  All communication between coordinator and workers is handled via gRPC, using protocol buffers for efficient, strongly-typed message serialization.

Key Implementation Details

1. Service Definition (Protobuf)

- The computation tasks and results are defined in **.proto** files.
- Services include methods such as **AssignTask** (for distributing work) and **RegisterWorker** (for worker registration).

2. Coordinator Logic

- **Task Distribution:**

  The coordinator receives a computation request (e.g., MRI image slices for tumor analysis), splits the data into chunks, and assigns each chunk to a worker using gRPC calls.

- **Parallel Execution:**

Tasks are dispatched to workers in parallel using Kotlin coroutines, allowing all workers to process their assigned data simultaneously.

- **Result Aggregation:**

Once all workers complete their tasks, the coordinator collects and aggregates the results (e.g., combining partial tumor volume estimates).

3. Worker Logic

- **Registration:**

Each worker registers itself with the coordinator using a gRPC call, providing its address and capabilities.

- **Task Execution:**

Upon receiving a task via gRPC, the worker processes the data (e.g., runs a machine learning model on image slices) and returns the result to the coordinator.

4. Real-Time Feedback

- **Event Bus:**

The system uses an event bus to post real-time status updates (e.g., task assigned, task completed, errors) from both coordinator and workers.

- **UI Integration:**

The application UI observes these events and displays real-time logs and progress indicators, providing transparency and immediate feedback to the user.

5. Fault Tolerance

- If a worker fails or becomes unresponsive, the coordinator can reassign the task to another available worker, ensuring robustness.

- **Testing**

1. Functional Testing

Test Case ID	Test Description	Input	Expected Output	Status
TC-F01	ROI detection for each MRI slice	MRI Sequence	Slices with Bounded ROI	Passed
TC-F02	Seed detection for each MRI slice	Slices with bounded ROI	Seed point coordinates (x, y)	Passed
TC-F03	Fuzzy logic-based volume estimation	Alpha cut, Roi list, seed list and MRI Sequence	Affinity Matrix and estimated Volume	Passed
TC-F04	Display results on UI	Estimated volume, image	Tumor overlay and volume shown in interface	Passed
TC-F05	Export results as PDF	Completed scan	Downloadable or shareable PDF report	Passed

2. Performance Testing

Test Case ID	Test Description	Input	Expected Output	Status
TC-P01	ROI detection model inference time	MRI Scan	Inference time < **45** seconds	Passed
TC-P02	Seed detection model inference time	Slices with tumor with marked bounds	Inference time < **7** seconds	Passed
TC-P03	Fuzzy logic-based volume estimation		Inference time < **1** second	Passed
TC-P04	Full process detection and volume estimation	MRI Scan	Total processing < **45** seconds	Passed
TC-P05	Distributed processing task completion	4x worker devices for full process	Parallel execution < **15** seconds	Passed

3. Offline Mode Testing

Test Case ID	Test Description	Input	Expected Output	Status
TC-O01	Run complete pipeline offline	MRI Scan	Detection, segmentation, and report	Passed
TC-O02	gRPC discovery and task sharing with no internet	Local Wi-Fi only	Worker devices connect and process tasks with consistent results	Passed
TC-O03	Export reports offline	Completed scan	Save PDF to local storage	Passed

4. Security & Privacy Testing

Test Case ID	Test Description	Input	Expected Output	Status
TC-S01	Ensure data remains on device	Full workflow	No external/cloud access attempted	Passed
TC-S02	Unauthorized access blocked	Restricted mode user	No access to patient scans or reports	Passed

5. Usability & Compatibility Testing

Test Case ID	Test Description	Input	Expected Output	Status
TC-U01	UI scales across screen sizes	Phone + tablet	Responsive layout and readable UI	Passed
TC-U02	Dark mode rendering	System in dark mode	Proper theme applied to all screens	Passed
TC-U03	Touch interaction responsiveness	User gestures	Smooth interaction (swipe, tap)	Passed

References

[1] S. A. Yones and A. S. Moussa, "A Fuzzy Tumor Volume Estimation Approach Based on Fuzzy Segmentation of MR Images," World Academy of Science, Engineering and Technology, vol. 71, pp. 720-725, 2012.

[2] R. Zhang, Z. Zhou, W. Wu, C.-C. Lin, P.-H. Tsui, and S. Wu, "An improved fuzzy connectedness method for automatic three-dimensional liver vessel segmentation in CT images," Journal of Healthcare Engineering, vol. 2018, Article ID 2376317, 18 pages, Oct. 2018. DOI: 10.1155/2018/2376317.

[3] B. Reig, L. Heacock, K. J. Geras, and L. Moy, "Machine learning in breast MRI," Journal of Magnetic Resonance Imaging, vol. 52, no. 4, pp. 998–1018, Jul. 2019. DOI: 10.1002/jmri.26852.

[4] B. S. Anami and P. H. Unki, "A Fuzzy-C-Means Approach for Tissue Volume Estimation in Brain MRI Images," in Proceedings of the National Conference on Recent Advances in Information Technology (NCRAIT), no. 3, pp. 21–24, February 2014.

[5] VSCO Engineering Team, "Suggesting Presets for Images: Building 'For This Photo' at VSCO," *TensorFlow Blog*. [Online]. Available: https://blog.tensorflow.org/2019/06/vsco-suggesting-presets-for-images.html. Accessed: Feb. 9, 2025.

[6] "What is the OCR feature in WPS Office," *WPS Office Academy*. [Online]. Available: https://www.wps.com/academy/what-is-the-ocr-feature-in-wps-office/about-wps/1863224/. Accessed: Feb. 9, 2025.