CBIS-DDSM ANALYZER

Yuval Shabat ,Liav Levi, Konstantin Margulyan
Advisor: Prof. Gil Ben-artzi

1 Abstract

The Curated Breast Imaging Subset of the Digital Database for Screening Mammography (CBIS-DDSM) is an open-source dataset widely used by researchers for developing algorithms aimed at detecting and segmenting cancerous breast lesions. Despite its importance, accessing and analyzing this dataset presents a challenge, as it requires significant coding effort, limiting its immediate usability for many researchers. To address this issue, we developed an Electron-based graphical user interface (GUI) that provides a streamlined and intuitive method for exploring the CBIS-DDSM dataset. The program enables users to filter data based on various parameters, view detailed patient information, and save frequently used queries for quick access. By simplifying dataset accessibility, the tool lowers the entry barrier for researchers, fostering more efficient exploration and utilization of the dataset for medical imaging research.

2 Introduction

Breast cancer remains one of the leading causes of cancer-related deaths worldwide, and is **the most common cancer in women** in the United States, except for skin cancers. It accounts for about **30% of all new female cancers each year**. Making early detection critical for improving patient outcomes. Mammography is a key imaging technique used to identify cancerous lesions at an early stage. The Curated Breast Imaging Subset of the Digital Database for Screening Mammography (CBIS-DDSM) is an open-source dataset containing mammography scans and corresponding metadata, such as patient demographics and pathology reports. This dataset is invaluable for researchers developing machine learning algorithms for the detection and segmentation of breast lesions, aiding in the fight against breast cancer.

However, despite its importance, accessing and analyzing the CBIS-DDSM dataset presents challenges. The dataset's size and complexity often require researchers to write custom code to load, filter, and visualize the data. This presents a significant barrier, particularly for those without strong programming backgrounds or those who wish to focus on algorithm development rather than data management. The current accessibility of the CBIS-DDSM is not user-friendly, and the initial coding effort needed to interact with the dataset slows down research progress.

To address these issues, this project aims to develop a graphical user interface (GUI) using Electron, a popular framework for building cross-platform desktop applications. The program is designed to provide an intuitive interface for browsing, filtering, and analyzing the CBIS-DDSM dataset without the need for coding. It includes features such as customizable filtering options, patient detail viewing, and the ability to save and load frequently used queries. By lowering the technical threshold required to access the dataset, this tool will make the CBIS-DDSM more

readily accessible to a broader range of researchers, facilitating faster exploration and analysis.

This paper will describe the design and development of the Electron-based program, discussing the dataset structure, key features of the GUI, and the technology stack used to build the application. Additionally, the paper will evaluate the performance and usability of the program, highlighting its potential to accelerate research in the field of breast cancer detection.

3 Methodology

3.1 Dataset Overview

The Curated Breast Imaging Subset of the Digital Database for Screening Mammography (CBIS-DDSM) contains mammography scans from **1,566 patients**, collected across various locations in the United States. This dataset is instrumental in developing and testing algorithms for detecting and segmenting breast lesions. It includes three types of images in DICOM (Digital Imaging and Communications in Medicine) format:

1. **Full Mammography Image:** The complete mammogram showing the entire breast.
2. **Region of Interest (ROI):** A zoomed-in view of a particular area where a suspected lesion is located.
3. **Mask Image:** A black-and-white image that highlights the mask of the suspected lesion, corresponding to the full mammography image.

Each patient entry in the dataset includes comprehensive metadata about the images and findings, which serve as essential parameters for filtering. These include:

- **Breast Side:** Indicating whether the image is of the left or right breast.
- **Image View:** Either CC (craniocaudal) or MLO (mediolateral oblique), which are the standard views taken during a mammography scan.
- **Abnormality ID:** Patients may have more than one suspected lesion, each identified with a unique ID.
- **Abnormality Type:** Categorized as either Calcification or Mass. For Mass, additional details such as Mass Shape and Mass Margins are provided. For Calcification, attributes like Calcification Type and Calcification Distribution are noted.
- **Assessment (BIRADS Category):** An integer value (0–5) representing the Breast Imaging Reporting and Data System (BIRADS) classification, which guides the diagnostic process:
  0 = Incomplete, additional imaging needed
  1 = Negative
  2 = Benign
  3 = Probably benign
  4 = Suspicious
  5 = Highly suggestive of malignancy
- **Subtlety:** A rating from 1 to 5 representing the difficulty in detecting the abnormality, where 1 = subtle and 5 = obvious.
- **Pathology:** This indicates the final diagnosis of the lesion and is categorized as:
- **Benign:** Non-cancerous findings confirmed by pathology or other means.
- **Benign without Callback:** Benign findings that did not require further testing.
- **Malignant:** Lesions confirmed to be cancerous by pathology.
- **Breast Density:** A rating between 1 and 4 that reflects the overall density of breast tissue, which impacts the visibility of lesions on the mammogram:
  1 = Almost entirely fatty
  2 = Scattered fibroglandular density
  3 = Heterogeneously dense
  4 = Extremely dense, reducing the sensitivity of mammography.

The DICOM format is widely used for storing medical images and consists of two main components: a header, which contains metadata such as image dimensions and depth, and the pixel data, which represents the actual image.

3.2 Technology Stack

The project leverages the **Electron** framework for building a cross-platform desktop application. The backend server is written in **Python** using the **Flask** library, which handles data requests and processing. Initially, the application was developed using the **Kivy** library in Python, but several technical limitations led to the transition to Electron.

3.3 Program Architecture

The program is built using Electron, a framework that allows for the creation of desktop applications using web technologies such as **HTML, CSS, and TypeScript**. The application's architecture consists of three main components:

- **User Interface (UI):** The front-end of the application, built using web technologies, allows users to interact with the dataset through an intuitive GUI. The UI displays patient data, mammography images, and filtering options.
- **Backend Logic:** The Python Flask server handles data retrieval and processing, interacting with the CBIS-DDSM dataset to serve relevant data to the Electron app. The backend processes queries, retrieves patient metadata, and returns filtered results.
- **Data Handling:** The app processes DICOM images, converts them into more accessible formats (JPEG) for visualization, and manages the retrieval of patient metadata for quick filtering and viewing.

3.4 Key Features

The Electron program is designed with several core functionalities to enhance the accessibility of the CBIS-DDSM dataset:

- **Filtering Options:** Users can filter patients by parameters based on the metadata unique values. The parameters are not “hard-coded”, and taken directly from the dataset, to allow the most flexibility and up-to-date information. This allows researchers to narrow down the dataset based on their specific needs.

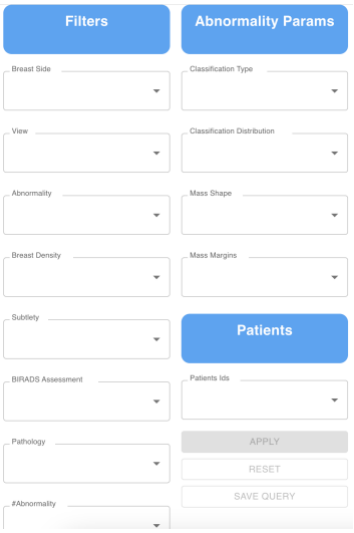

Figure 1: Filter menu

- **Viewing Patient Details and Images:** Users can view patients along with their corresponding mammography images (in reduced resolution for faster loading). For each patient, all abnormalities, along with their Region of Interest (ROI) and mask images, are displayed.

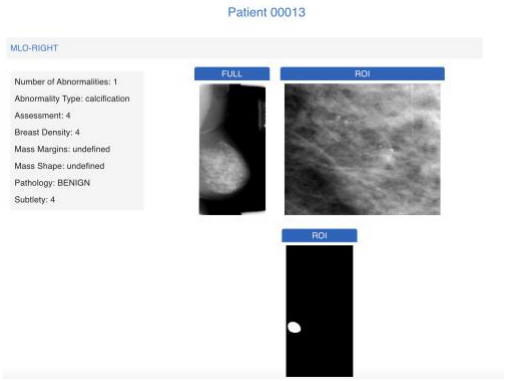

Figure 2: Patient View Page

– **Saving and Loading Queries:** The program allows users to save frequently used queries, enabling quick re-filtering without having to manually input the parameters each time.

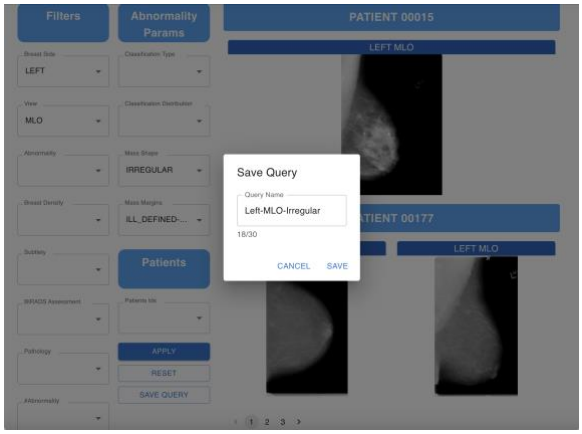

Figure 3: Saving Query in Main Page

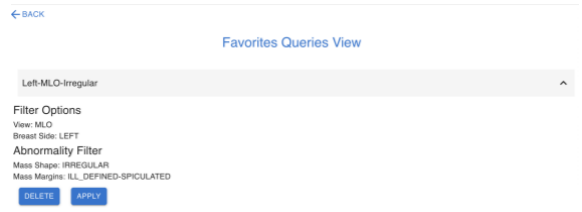

Figure 4: Favorites Queries View Page

3.5 Implementation

The implementation process was divided into three stages, starting with the client-side development using dummy data. By initially focusing on the frontend, we were able to design and test the user interface (UI) and interactions independently from the backend, which helped us gain a clear understanding of the necessary API endpoints and data formats. This incremental approach allowed us to iteratively develop the backend for each stage, ensuring that the data retrieval and processing workflows were optimized. It also gave us valuable insights into how to efficiently structure requests and send data between the frontend and backend.

Stage 1: General View of Patients' Mammography Scans
The first stage of the program displays the general list of patients and their full mammography scans. Filtering options are provided to allow users to filter patients based on specific attributes like breast side, abnormality type, BIRADS category, and patients IDs. All images in this view are converted to JPEG format at a lower resolution to enhance loading times.

Stage 2: Detailed View of Patient Abnormalities
In the second stage, users can select a specific patient to view detailed information about their abnormalities, including all associated full mammography image**,** ROI images and mask images. Those images can be opened into a full page overview, providing accurate visual data for further analysis.

Stage 3: Favorite Queries Page The third and final stage implements a page for saving and managing favorite queries. Users can save frequently used filter configurations, which can be quickly reloaded in future sessions. This feature enhances user efficiency by eliminating the need to reapply filters manually.

4 Results

The Electron-based program developed for interacting with the CBIS-DDSM dataset successfully achieved its main objectives: providing an intuitive user interface for dataset exploration, facilitating advanced filtering options, and reducing the coding threshold required for dataset access.

4.1 User Interface Usability

The program offers a clean, user-friendly interface that allows researchers to interact with the CBIS-DDSM dataset without the need for programming knowledge. Users can easily filter patients based on various parameters available in the dataset. The system displays patient images in a condensed manner, allowing users to get an overview of the dataset at a glance. Additionally, detailed views of the Region of

Interest (ROI) and mask images for individual patients are available.
Feedback from early testers indicated that the interface was easy to navigate and significantly streamlined the dataset exploration process. In particular, the ability to visualize patient-specific abnormalities and their associated metadata, without needing to parse the raw data manually, was highlighted as a key strength.

4.2 Filtering Performance

The filtering functionality performed efficiently, allowing users to quickly narrow down the dataset based on the desired parameters. One of the key aspects of the program is the **server-side filtering**, which is implemented using the **pandas** library to manage and process the CBIS-DDSM dataset. By handling filtering on the server, the program ensures that large datasets are processed efficiently without overloading the client-side application.
This architecture provides several advantages:

- **Improved Performance**: Filtering on the server means that only the relevant data is sent back to the client, reducing the data transfer overhead and speeding up the UI response times. This approach is particularly beneficial when working with large datasets like CBIS-DDSM, where transferring the full dataset to the client would result in high latency.
- **Efficient Data Management**: The pandas library provides powerful tools for data manipulation and filtering. It allows for efficient handling of complex queries (e.g., filtering by multiple parameters such as breast density, pathology type, or BIRADS score) and ensures accurate results.
- **Scalability**: By offloading the data processing to the server, the system can scale better to handle larger datasets or more complex filtering operations in the future.

Tests showed that even complex queries, such as filtering based on multiple criteria, were processed quickly. The server-side approach allowed the backend to efficiently manage the filtering logic while returning only the necessary data to the frontend.

4.3 Image Rendering and Quality

The program successfully manages the DICOM images by converting them to JPEG format on the **server side**. This conversion reduces the image size, allowing for **faster transmission** over the network. Only the JPEG images are sent to the client, minimizing the amount of data transferred and improving performance.

On the client side, the **Buffer library** is used to receive the binary JPEG data efficiently. This approach ensures that the images load quickly in the user interface, especially when browsing through multiple patient scans.

In addition, **pagination** is implemented on the client side, limiting the display to two patients per page. This ensures that images are not all fetched and rendered at once, which helps prevent performance bottlenecks and allows the user to navigate the dataset more smoothly.

Advantages of this approach include:

- **Reduced Data Load:** JPEG images are smaller than DICOM, minimizing the amount of data transferred.
- **Improved Performance:** Server-side image conversion and client-side pagination ensure that only the necessary images are loaded, keeping the client responsive.
- **Efficient Image Handling:** The Buffer library ensures smooth rendering of binary data, while pagination prevents overloading the system by displaying two patients at a time.

4.4 Query Management

The ability to save and load queries for quick filtering proved highly effective. Users were able to save custom filter configurations and reload them at a later time, which reduced the time spent manually re-entering filtering parameters. This feature particularly benefits researchers who focus on specific subsets of data for ongoing experiments, allowing them to easily revisit and refine their previous searches.

4.5 Backend Efficiency

The Flask backend performed well under typical usage conditions, with API requests efficiently retrieving the necessary patient data and images. By adopting an incremental development approach (starting with frontend implementation using dummy data), the final backend was fine-tuned for speed and data retrieval accuracy. Testing revealed that most API requests, even for complex filters, were processed within a few seconds.

4.6 Technical Challenges

During the early stages of development, we initially chose **Kivy** as the framework for the client-side UI. However, we encountered significant limitations that hindered progress:

- **Limited UI Component Flexibility**: Kivy lacked the flexibility required to create complex and modern user interface components, particularly when it came to handling dynamic image displays and advanced filtering options. Customizing UI elements in Kivy required excessive manual effort, slowing down development.
- **Slow Development Speed**: Kivy's structure and development workflow were cumbersome for the needs of the project. Building and testing components, especially when handling large datasets and images, was slow and inefficient.

Due to these limitations, we decided to transition to **Electron** for the frontend. This switch provided greater flexibility in building the UI and significantly accelerated the development process. By utilizing Electron's integration with web technologies (HTML, CSS, TypeScript), we were able to implement a more responsive and user-friendly interface, ultimately improving both the speed and quality of development.

5 Discussion

5.1 Contributions

The development of this Electron program for visualizing the CBIS-DDSM dataset has led to several key contributions in the field of medical imaging research. By integrating a user-friendly interface with a powerful backend, the program provides researchers with an efficient tool to explore and analyze mammography data.

Key contributions include:

- **Enhanced Accessibility:** The application lowers the barrier to entry for researchers by simplifying access to the CBIS-DDSM dataset. The user-friendly interface enables users to filter and review mammography scans without needing extensive programming knowledge.
- **Server-Side Processing:** By implementing server-side filtering using pandas and converting DICOM images to JPEG format, the program improves performance and responsiveness. This design allows users to interact with large datasets without experiencing significant delays, facilitating a smoother analysis workflow.
- **Client-Side Pagination:** The introduction of pagination helps manage the display of patient data efficiently, ensuring that users can navigate through the dataset without being overwhelmed by information.

5.2 Limitations

Despite its contributions, the project has several limitations that need to be addressed.

- **Executable Packaging:** A significant limitation is the inability to create a standalone executable (EXE) file for users. The integration of the Python Flask server with the Electron TypeScript client presents challenges in merging these components into a single deployable application. As a result, users must perform additional steps to set up the program.
- **User Setup Requirements:** While we developed a script to streamline the setup process after downloading the repositories from GitHub, this solution still requires users to have some familiarity with Git and command-line operations. This may deter less technical users from effectively utilizing the program.

5.3 Future Work

To enhance the program's usability and functionality, several areas for future work have been identified:

- **Executable Creation:** A primary focus should be on developing a method to package the application into a single executable format. This enhancement would significantly improve the user experience by allowing easy installation and access to the program.
- **Expanded Features:** Future iterations of the program could benefit from additional features, such as advanced data visualization tools, enhanced filtering options, and the ability to handle more extensive datasets. These improvements would make the application more versatile and useful for a broader range of research applications.

6 References

[1] Flask documentation: https://flask.palletsprojects.com/en/latest/
[2] React documentation: https://react.dev/
[3] Electron documentation: https://www.electronjs.org/docs
[4] MUI documentation: https://mui.com
[5] NumPy documentation: https://numpy.org
[6] Panda's documentation: https://pandas.pydata.org
[7] TCIA API: https://github.com/kirbyju/TCIA_Notebooks/tree/main
[8] CBIS-DDSM dataset: https://www.cancerimagingarchive.net/collection/cbis-ddsm/
[9] Electron-Vite documentation: https://electron-vite.github.io/guide/getting-started.html
[10] NBIA Search REST API Guide: https://wiki.cancerimagingarchive.net/display/Public/NBIA+Search+REST+API+Guide#NBIASearchRESTAPIGuide-getSingleImage
[11] TCIA REST API Guide: https://wiki.cancerimagingarchive.net/display/Public/TCIA+REST+API+Guide