AI for Chest X-ray

Development Plan

Group 18

Supervisor

Name: Dr. Mehdi Morad
Email: moradm4@mcmaster.ca
Organization: McMaster University, CAS

Team Members

Name: Abhishek Sharma
Email: shara109@mcmaster.ca
Student Number: 400322503

Name: Anthony Vu
Email: vua11@mcmaster.ca
Student Number: 400306059

Name: Hussein Saad
Email: saadh@mcmaster.ca
Student Number: 400307995

Name: Nathan Starr
Email: starrn@mcmaster.ca
Student Number: 400323095

Name: Yuvraj Jain
Email: jainy3@mcmaster.ca
Student Number: 400259484

Table of Contents

1.0 Revision History

Version Number	Authors	Description	Date
0	● Abhishek Sharma ● Anthony Vu ● Hussein Saad ● Nathan Starr ● Yuvraj Jain	Started Initial Draft	October 20th, 2023

2.0 Team Meeting & Communication Plan

- Group communication will be through a group chat on Microsoft Teams.
    - The response time should be as fast as possible. (≤ 12 h)
- Group meetings will either take place on Microsoft Teams or in-person.
    - The time and location of meetings can be determined by the program manager by communicating in the group chat.
    - Once a meeting time is decided a formal meeting invite can be sent to all required participants from Outlook to ensure that the meeting time is blocked off on everyone's calendar.
- There will be a weekly team meeting taking place on Microsoft Teams every Monday at 7:30pm.
    - All team members will be required to join this meeting.
    - These meetings will be an opportunity for team members to provide updates on assigned tasks/features, discuss issues, and discuss future plans.
    - These meetings will be 30 minutes in length.
    - Additional follow up meetings will be scheduled as needed.
    - Minutes of the meetings: these are notes taken during the meeting by a note taker. The note taker role will be circulated among all team members and the note taker for the meeting will be decided by the project manager at the beginning of the meeting.
        - These notes should provide information on the current status of each task and the future tasks for each team member with the tentative deadlines.
- All team documents will be stored in a shared Google Drive.
- All documents will be shared/collaborated on with either Google Docs or Google Colab.
- GitLab Issues will be used for the program management aspect of the project.
- Communication with the supervisor will be through email and scheduled meetings.
    - Meetings with the supervisor can be scheduled through email communication or by booking a time in the shared spreadsheet on the general channel of the capstone team. (Once a meeting time is decided a formal meeting invite will need to be sent to all required participants from Outlook to ensure that the meeting time is blocked off on everyone's calendar).
- There will be separate meetings/working sessions taking place for each main feature being worked on. These meetings will only require the associated team members working on the feature and will be scheduled as needed.
- Coding/working meetings will be scheduled as needed.

- Communication with the supporting TA will either be through email or through messages in the assigned group channel of the capstone team.
- Meetings with the supporting TA can be scheduled through the assigned group channel in the capstone team. (The scheduled meetings will take place in this channel).

3.0 Team Member Roles

Roles	Team Member(s)
● Training/Testing the model. ○ Cross training pre-trained model. ○ Training our own model. ○ Using the TorchMetrics library to test the F1 score, precision score, IoU, ROC AuC). ● Deployment ○ Create a simulated PACS server. ○ Implement our model in the backend. ● Working on presentations and group documents.	Abhishek Sharma
● Working on frontend website and middleware. ○ Creating a hosted site that will accept both JPG and DICOM images. ○ Create a website that displays results from the model. ○ Implementing the backend will convert all uploaded DICOM images to JPG format. ○ Work on middleware to connect backend and frontend. ● Working on presentations and group documents.	Anthony Vu
● Training/Testing the model. ○ Cross training pre-trained model ○ Training our own model ○ Using the TorchMetrics library to test the F1 score, precision score, IoU, ROC AuC). ● Working on presentations and group documents.	Hussein Saad
● Training/Testing the model. ○ Cross training pre-trained model. ○ Training our own model. ○ Using the TorchMetrics library to test the F1 score, precision score, IoU, ROC AuC). ● Working on presentations and group documents.	Nathan Starr
● Working on frontend website and middleware. ● Working on presentations and group documents. ○ Creating a hosted site that will accept both JPG and DICOM images. ○ Implementing the backend will convert all uploaded DICOM images to JPG format.	Yuvraj Jain (Coordinator/ Program Manager)

○ Create a website that displays results from the model. ○ Work on middleware to connect backend and frontend. ● Project Manager Responsibilities: ○ Is responsible for booking meetings, keeping track of the issue board, keeping track of timelines, and booking follow-ups as needed.	

4.0 Workflow Plan

A. How will you be using GitLab or GitHub, including branches, pull requests, and issue management?

Version Control:

- We will be using Git for version control and GitLab for managing our repository and storing code.
- GitLab's Issue board will be used to facilitate and streamline the management of issues and tasks through a visualized workflow plan.
- All the code will be stored under the *AI For Chest X-ray* repository on GitLab.
- All five group members are added as *Owners*. While the Professor and TAs are added as *Developers*.

Branches:

- Sample branch name: 1-implemeting_user_auth
- The first part indicates the issue with which the branch is associated. We do this by mentioning the issue ID.
- The second part would briefly describe the purpose of the branch.

Commits:

- Sample commit message: Adding user authentication, Issue: #1
- The first part of the commit statement will provide the rationale behind the commit.
- The second part will indicate the issue with which the commit is associated. We do this by mentioning the issue ID after the '#' symbol.

Information Tracking:

- We will create an issue for every task associated with the project, these can include implementation of a new feature, bug fixes, enhancement tickets, etc. A summary of how we do this is as follows:
    - Create a new issue
        - Briefly describe what the purpose of the issue is, what needs to be done/fixed/implemented.
    - Add a label that suits the goal of the issue:
        - New Feature
        - Improvement
        - Bug
        - Documentation
    - Assign the issue to Developers
    - Add a due date to signify the time sensitivity of the issue

- For better visualizing the workflow plan, each issue will be designated in the below four labels to keep track of and check progress on the Issue board:
    - *To do* - for an issue that has not been started
    - *In Progress* - for an issue that is actively being worked on
    - *In Review* - for an issue that is done and awaiting reviewer's approval
    - *Done* - for an issue that is completed and merged
- After completion of an issue, a merge request (MR) must be opened to merge the changes into the dev branch. The MR should:
    - Describe what the issue did, a summary of how it was implemented, and what is fixed/implemented now
    - *Assignee* - the developers who worked on the issue
    - *Reviewers* - at least 1, ideally 2 reviewers should be assigned based on the complexity of the issue
- GitLab automatically closes an issue once the merge request is successful. (Only requirement is to mention the issue number which we are in the commit statement)

Tracking Changes to Supporting Documentation

- All development documents contain a revision log that will be used whenever a finished document is updated. The authors of these revisions should identify who was involved with the change and when the change occurred, what revisions were made including the purpose for said revisions.

Process Workflow

- We have three branches: dev, qat and prod.
    - *dev* will act as our main branch where the active code resides
    - *qat* will act as an intermediate branch where the testing will be done
    - *prod* will act as the final deliverable of our project which is fully implemented and tested.
- All members are expected to follow the following when doing a task:
    - Do a git pull from the dev branch to keep the code up-to-date
    - Create a new branch from the dev branch (dev is acting as our base branch)
    - Commit code with descriptive messages
        - Add unit tests/integration tests (if necessary)
    - Open merge request (MR)
        - Reviewers ensure that code meets acceptance criteria
    - Merge the branch into the dev branch
- The code from dev branch is pushed to qat branch to perform testing of the system as a whole ensuring all implemented functional works well
- After successful testing and verification, the code from qat branch is pushed to prod branch which is the final deliverable. (No developer is allowed to make direct changes to prod branch)
- Such a three-tier approach ensures structure and verifiability to our code while also ensuring fewer conflicts and more reliability.

B. Using agile methods is encouraged. You can use scrum and sprint planning in Jira. State if you will do this. We will not monitor your sprints.

We are planning to use the agile development methodology to work on our project since it is commonly and effectively used in the industry. We will be specifically using the scrum method within the agile methodology. We will have regular sprint meetings through Teams and use GitLab's Issue board as our Kanban board (Open, To do, In Progress, In Review, Done, Closed).

C. Where do you store your data (especially if you are doing machine learning)?

We plan to store our training data and testing data on our shared Google Drive. To accommodate the large amounts of data we plan to buy additional storage for the shared Google Drive. We are choosing to store our data on a shared Google Drive since it will allow us to access the data on any device and it will allow us to easily collaborate with the data. (We also have a combined capacity of ~5TB over all personal computers that can be used if needed due to load speed issues).

D. Where do you run/compute heavy tasks like training models?

We plan to run/compute heavy training models on Google Colab. We are planning to use Google Colab since it can be directly linked with our shared Google Drive allowing team members to easily collaborate on tasks and access data. We also like that we can run computing tasks directly in Google Colab by using the GPU provided by Google. Google Colab provides a relatively powerful GPU for free and it will help us resolve the GPU constraints of our personal devices. In the free version Google Colab notebooks can run for up to 12 hours. If additional computing power or computing time is needed we plan to upgrade to Google Colab Plus to get additional computing power and time. For any local work we will use a personal computer with a GTX 1070 GPU that can be run all-day.

E. What tool/method is used to achieve each of the requirements and achieve the performance metrics that were proposed in your SRS?

- Within **Pytorch,** we can use the **TorchMetrics** library, which will give us access to all performance metrics mentioned in section 5.8.3 of the SRS document (F1 score, precision score, IoU, ROC AuC).
- **Sklearn** will give us access to testing methods such as the k-cross validation mentioned in section 5.9 of the SRS document.
- **TorchXrayVision** provides popular pre-trained models, which we can modify to achieve our metrics within resource constraints. These pre-trained models will serve as benchmarks for our future custom models due to their extensive usage and reliability in the industry.
- **Flask** will be our medium of communication between our ML pipeline, the PAC server, and our client side product. Optimizing Flask code will reduce input-output time, bringing us closer to our reporting time requirement specified in section 5.1 of the SRS.
- **JINJA** will allow us to integrate HTML, CSS and Python which will serve as the tools we use to create our frontend.
- **Google Drive** will be used to store training and testing data. If the training process is too slow, we will revert to **Physical SSD Storages**.
- **Google Colab** will be used for collaboration as well as model training.

5.0 Proof of Concept Demonstration Plan

- Demonstration slideshow of project purpose, objective and planned features.
- Have a prototype frontend with a rough **intractable website**.
- **Backend** of our model will run a **modified pre-trained model** that will return tags of diseases identified.
- The Backend will be run locally on a laptop.
- Website will accept both **JPG and DICOM images.**
    - The backend will convert all uploaded DICOM images to JPG format.
- Demonstration
    - Upload a local DICOM image to the frontend.
    - The uploaded Image is sent to the back end.
    - Back end runs a pre-trained model and produces tags.
    - These tags are sent back to the frontend.
    - Frontend displays what tags the model identified on the image.

6.0 Technology

A. Specific programming language (frontend and backend), coding environment. State if will use unit testing framework, why or why not. If your project is primarily software development (as opposed to research), we expect you to follow software engineering best practices including unit testing.

- **ML:** Python will be used since it is the most supported programming language for AI chest x-ray libraries.
- **Frontend:** Python, HTML, and CSS.
- **Backend:** The AI model in the backend will be made using a pre-trained model found on TorchXRayVision that will run on images from a simulated version of a PACS server.
- **Coding Environment:** We will use Google Colab for easy collaboration on the ML portion and VS Code will be used for any necessary local development for the frontend and back-end.
- We will use GitLab CI/CD tools for automated testing on new merges.
- PEP 8 naming standards will also be followed for best practices.

B. ML libraries (if relevant)

- We will use TorchXRayVision for their pre-built models and for training our own models.
- PyTorch, the library that TorchXRayVision was built on, may be used for specific optimizations and finetuning.
- OpenCV can be used for computer vision tasks, such as mapping affected areas.
- Scikit-learn also includes a lot of useful classification features.
- The Keras library might be used if needed.

C. Will you use a GPU? Any other relevant technology aspects.

- **GPU:** Since we will be using Google Colab to work on the training of the model we will use the GPU provided by Google for Google Colab. In the

free version Google Colab notebooks can run for up to 12 hours. If more powerful hardware or more computing time is required, the team is open to paying for Google Colab Pro or AWS. For any local work we will use a personal computer with a GTX 1070 GPU that can be run all day.

- **CPU and RAM:** Since we will be using Google Colab to work on the training of the model we will use the CPU and RAM provided by Google for Google Colab. For local tasks we have consumer-grade processors and a RAM capacity ranging from 8GB to 32GB. If more powerful hardware is required, the team is open to paying for Google Colab Pro or AWS.
- **Storage:** We will be using a shared Google Drive to store our training and testing data to make it easier to collaborate. Google Drive comes with 15GB of storage by default and additional storage up to 2TB can be purchased if needed. We also have a combined capacity of ~5TB over all personal computers that can be used if needed.
- **Operating Systems:** Windows, MacOS, Linux.

7.0 Project Scheduling

Task	Start Date	Duration (Days)	Actual Due Date
Form team and create GitLab repository	September 8, 2023	14	September 22nd, 2023
Select Project Idea and work on project title and description	September 8, 2023	21	September 29th, 2023
Work on potential revision of project description	September 29, 2023	7	October 6th, 2023
Work on SRS Document	October 8, 2023	12	October 20th 2023
Work on Project Plan	October 20, 2023	7	October 27th, 2023
Download/deploy pre-trained model found on TorchXrayVision	October 29, 2023	14	November 17th, 2023
Create a rough version of frontend.	October 29, 2023	14	November 17th, 2023
Work on Proof of Concept Demonstration	November 5, 2023	12	November 17th, 2023
Work on Design Document	January 8, 2024	18	January 26th, 2024
Work on V & V Plan	January 8, 2024	25	February 2nd, 2024
Make updates for the Final Submission	November 17, 2023	112	March 8th, 2024
Work on Final Demonstration Presentation	March 8, 2024	14	March 22nd, 2024
Create EXPO Demonstration and Poster	March 22, 2024	8	April 1st, 2024