Requirements and Installation

***Basic knowledge of command line is required.**

Software:

- Docker

Host System:

- 15GB storage
- 8GB RAM
- Modern CPU

Installation Instructions:

1. Download and install the Docker command line environment:

   ```
   sudo apt-get install docker-ce docker-ce-cli containerd.io
   ```

   a. Alternatively, you download Docker Desktop if you prefer a desktop GUI.

2. Download ark_mirai:

   ```
   docker pull harrivle/ark_mirai
   ```

   a. If needed, you can add a suffix with a colon to download a specific version, i.e. "harrivle/ark_mirai:0.5.1"

3. Verify that the image has been successfully loaded:

   ```
   docker image ls
   ```

   You should see "harrivle/ark_mirai" under the "REPOSITORY" column.

Local Command Line Demo

The “ark/mirai” container creates a server for an API. The API implements the endpoint “/dicom/files” to receive DICOM files through a HTTP POST request. The API will then return predictions with the server’s response.

Running the Server/Model:

1. Download and extract demo data.

2. Choose a [PORT] to expose, i.e., 5000. Run the docker image:

   ```
   docker run -p [PORT]:5000 harrivle/ark_mirai
   ```

   You should see logging messages indicated that a server has been started.

3. With the extracted folder “mirai_demo_data/” in the same working directory, you can send a request to the server to run the model using the “curl” command like so:

   ```
   curl -X POST -F 'data={}' -F 'dicom=@mirai_demo_data/mlor2.dcm' -F
   'dicom=@mirai_demo_data/mlol2.dcm' -F 'dicom=@mirai_demo_data/ccr1.dcm' -F
   'dicom=@mirai_demo_data/ccl1.dcm' http://localhost:[PORT]/dicom/files
   ```

   The curl command may need to be installed with your package manager depending on your OS. Be sure to replace [PORT] with your chosen port number.

   a. **Important Note**: Each request to this endpoint should be the equivalent of a single exam. The model requires **four files** from the **same** exam corresponding to different mammography views.

4. After a half a minute or so, you will receive a JSON response like so:

```
{
  "data": {
    "predictions": {
      "Year 1": 0.0298,
      "Year 2": 0.0483,
      "Year 3": 0.0684,
      "Year 4": 0.09,
      "Year 5": 0.1016
    }
  },
  "message": null,
  "metadata": {
    "patientID": "001"
  },
  "runtime": "24.59s",
  "statusCode": 200
}
```

   a. On the hospital side, some program/script would need to be written to send files to the API and handle the JSON return object. This program ideally will be what interfaces with a database to grab the DICOM images and store the JSON predictions.

API Documentation

POST /dicom/files – Takes four .dcm files belonging to a single breast exam as input and returns probability diagnosis across five years.

Request JSON:

Field	Type	Description
data	JSON	Required freeform JSON object. Any data contained in the object will be returned in the HTTP response JSON. Can be empty.

Response JSON:

Field	Subfield	Type	Description
data		JSON	Data containing the \`predictions\` object i.e. {"predictions": {...}}
metadata		JSON	Freeform JSON object taken from the \`data\` field in the request JSON.
message		string	A message detailing any important information (usually regarding error messages).
statusCode		int	The HTTP status code of the response.
runtime		string	Runtime string in seconds, formatted as "0.00s".