Group 1: Medical Segmentation System Documentation

Ignatius Boadi
AIMS AMMI
Mbour, Sénégal
iboadi@aimsammi.org

Millicent Omondi
AIMS AMMI
Mbour, Sénégal
momondi@aimsammi.org

Leonard Sanya
AIMS AMMI
Mbour, Sénégal
lsanya@aimsammi.org

1 Introduction

Brain tumors are abnormal growths of cells in the brain or nearby areas, varying widely in size. Treatment options depend on the tumor type (cancerous or non-cancerous), size, and location. We have categorized brain images into two classes: those with tumors and those without. Find the github here.

- **Goal:** To develop a convolutional neural network (CNN) model from scratch that accurately segments tumor regions in brain images.
- **Pains:** Medical practitioners often face difficulties in identifying tumor regions and assessing tumor size within brain images.

1.1 Value Proposition

i. **Offer:** A machine learning model that accurately segments tumor-affected areas of the brain, highlighting both the size and location of the tumor.

ii. **Benefits:** By clearly identifying the affected regions and tumor size, medical professionals can make informed decisions regarding treatment options more quickly and efficiently.

1.2 Objectives

i. Develop a system that utilizes a CNN model for precise segmentation of brain tumor regions.

ii. Achieve high accuracy and reliability in segmenting brain images.

iii. Create a system adaptable to changes in appearance and environmental conditions.

1.3 Solutions

- **Technology:** Employ advanced techniques such as the ResNet-50 model for accurate identification of tumor regions within medical images.
- **Features:** The system will provide real-time analysis as CT or MRI scans are taken, quickly identifying tumors.
- **Integration:** Ensure compatibility with existing medical systems.

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

1.4 Feasibility

To implement this we need;

- Adequate computational resources for image processing and a database for image storage.

1.5 Potential Barriers

Challenges may include:

- Poor lighting conditions and the angle the image was taken may cause poor segmentation.
- Challenges include a lack of expertise in interpreting brain images, privacy concerns regarding medical data usage, and integration difficulties with the current infrastructure.

1.6 Data Sources

- **Training Data Sources:** publicly available datasets for initial training and validation.
- **Production Data Sources:** Images from the head CT scan and brain MRI from medical institutes.
- **Labeling Process:** Images will be labeled by trusted medical practitioners to ensure high accuracy in dataset preparation

1.7 Metrics

- **Primary Metrics:** Brain segmentation accuracy, false acceptance rate (FAR), false rejection rate (FRR), and system response time.
- **Tracking** Continuous monitoring of these metrics during testing and live operation to ensure performance standards are met.

1.8 Evaluation

- **Offline Evaluation:** Regular testing against a validation dataset to maintain consistent performance and adaptability to new data.
- **Online Evaluation:** Real-time assessment using data captured from operational environments to quickly resolve any segmentation issues.

1.9 Modelling

- **Mode:** Real-time processing on local edge devices to minimize latency and maximize responsiveness.
- **Deployment Strategy:** Integrate the brain tumor segmentation software with hardware at medical institutions, ensuring sufficient processing power for accurate segmentation.

1.10 Inference

- **Mode:** Real-time processing on local edge devices to minimize latency and maximize responsiveness.
- **Deployment Strategy:** Integrate the segmentation software into web applications for instant segmentation on mobile phones and laptops or desktops.

1.11 Inference

- **Mode:** Real-time processing on local edge devices to minimize latency and maximize responsiveness.
- **Deployment Strategy:** Integrate the brain tumor segmentation software with hardware at medical institutions, ensuring sufficient processing power for accurate segmentation.

1.12 Feedback

- **Sources:** Direct feedback from medical practitioners and patients, along with system logs and incident reports.
- **Utilization:** Utilize feedback to fine-tune image segmentation algorithms and enhance system usability.

1.13 Technological Considerations

We will be using the following tools in implementing the brain tumor segmentation system:

- **GitHub:** For version control and collaboration.
- **VS Code:** As the integrated development environment.
- **Jupyter Notebook:** Data exploration and code testing.
- **DVC:** For data versioning and management.
- **Docker:** For containerization, ensuring consistent deployment.
- **MLflow:** Tracking, versioning, and management of the model used.
- **MySQL:** Database to store user data and metadata.
- **FasAPI:** Web framework for building APIs.
- **GCP:** Cloud platform for model deployment and to ensure scalability.
- **Streamlit:** Frontend interface for users.

1.14 Project

- **Team:** comprises AI and machine learning engineers, medical practitioners, system integrators, and project managers.
- **Deliverables:** A fully functional brain tumor segmentation system, user manuals, and training sessions for medical practitioners.
- **Timeline:** Estimated project completion within 4-6 months from initiation, including testing and adjustments.

2 System Architecture

The brain tumor segmentation system has been designed to provide a seamless user experience from logging in, uploading images, and getting the segmentation. It ensures robust data management and security through the authorization procedure put in place using the Oath authenticator. The architecture 1 is divided into different key components, each serving its purpose for a smooth brain tumor segmentation process.

In figure 1, we observe the following components of the system:

- **User Clients:** In the system, both web and mobile clients can interact with the system. They will be able to log in and upload the scanned brain images.
- **Data Management:** The system uses DVC for data versioning, ensuring that all datasets are tracked. Uploaded images are processed through the data transformation pipeline before being stored in the pipeline.
- **Data Storage:** The images will be stored in the cloud on the GCP cloud bucket and queried using MySQL.
- **Model Registry:** It is utilized to manage the life-cycle of the machine learning model. The pre-trained model is registered and can be updated as needed, allowing easy retrieval and deployment.
- **brain tumor segmentation Service:** It processes the uploaded images for the users, creates a bounding box on the detected tumor, and segments it. The segmented images are then stored in the database.
- **Performance monitoring:** For tracking prediction accuracy and data pipeline performance. It ensures that any degradation in service quality can be addressed.

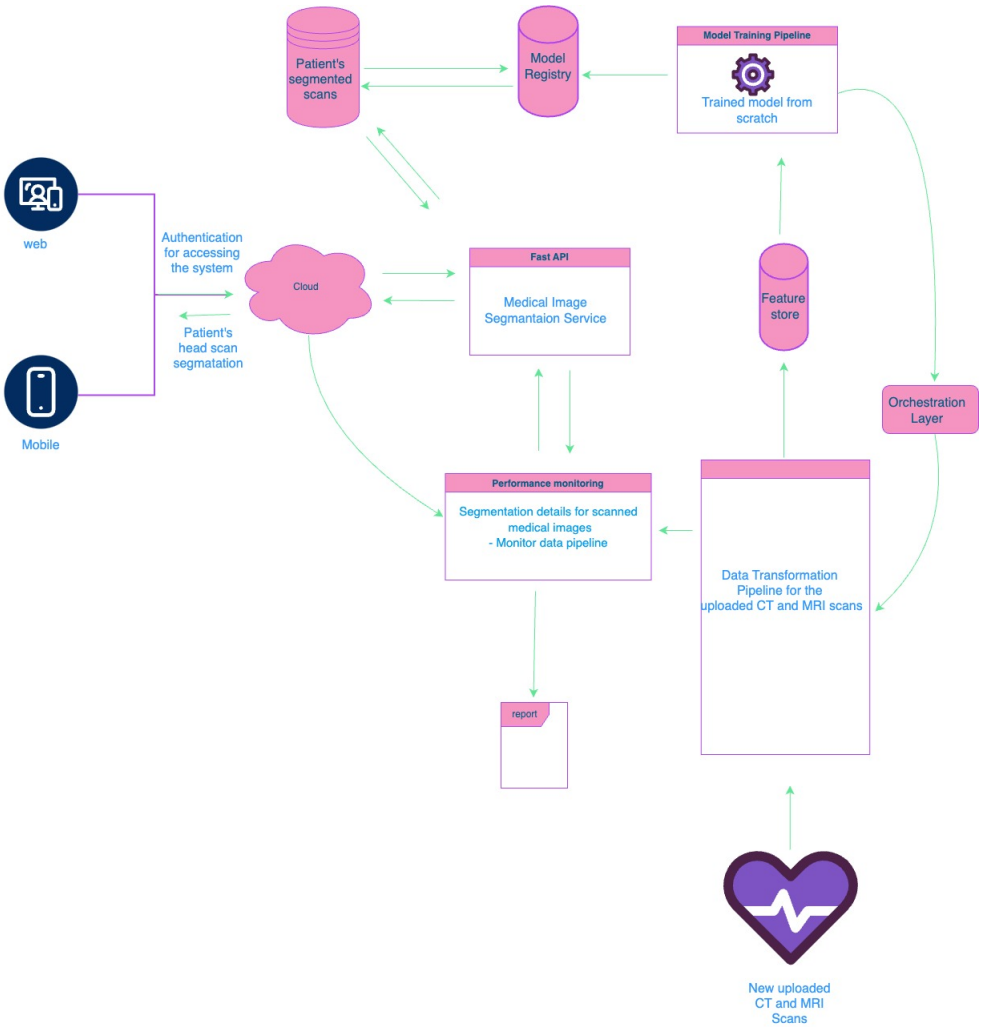

Figure 1: System Architecture Overview [source:neptune.ai]

2.1 Workflow

The workflow as shown in figure 2, starts with users uploading the brain scans. Then the set input image requirements are considered. If they are met, the system proceeds to pre-process the image, segment the part with a brain tumor, and output the segmented brain tumor scanned image. On the other hand, if the brain scan image did not meet the set standards, the process would end there.

3 Setup and installation

We will need to install the following tools and libraries before setting up our brain tumor segmentation system:

Tools: Python (version 3.7 or higher), docker, git, MySQL, DVC, and Anaconda or virtual environment.

3.1 Environment Setup

3.1.1 Local development

1. Clone our repository

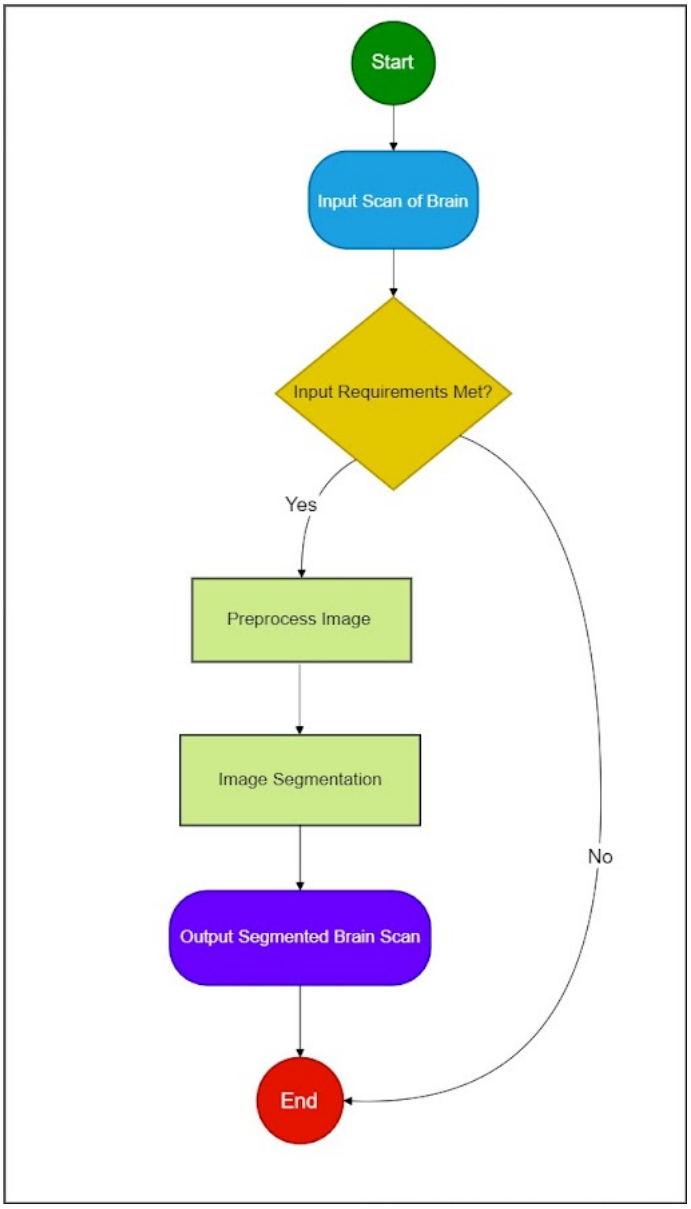

Figure 2: System design workflow [source: Authors]

```
git clone <git repo>
cd face_segmentation_system
```

2. create a virtual environment

```
python -m venv venv
source venv/bin/activate  # On Windows use ‘venv\Scripts\activate‘
```

3. Install required packages. Create a requirements.txt file with the following dependencies:

```
torch==2.4.1 #+cu118
opencv-python~=4.10.0.84
numpy~=2.1.1
scikit-image~=0.24.0
tifffile~=2024.9.20
fastapi
```

```
uvicorn
PyJWT
passlib
bcrypt==3.2.0
pandas
evidently
python-multipart
pdfkit
wkhtmltopdf
mlflow~=2.16.2
dagshub~=0.3.37
pillow~=10.3.0
torchvision
tqdm~=4.66.5
segmentation_models_pytorch
```

4. Install the required packages:

```
pip install -r requirements.txt
```

3.1.2 Using Docker

Docker allows seamless deployment of the application across different environments. Steps to set up the project using Docker:

1. Install Docker here.
2. Build the Docker image. Create a Docker file using "touch Dockerfile" in your terminal.

```
FROM python:3.11.5
# Set the working directory
WORKDIR /code

# Copy the requirements to the working directory
COPY ./requirements.txt /code/requirements.txt

# Install dependencies
RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt

# Copy the app directory contents to the working directory
COPY ./main.py /code/main.py

EXPOSE 8000

# Run the FastAPI application using Uvicorn
CMD ["sh", "-c", "uvicorn main:app --host 0.0.0.0 --port 8000"]
```

3. Build the docker image and run the docker container.

```
docker build -t image-segmentation-system .
docker run -d -p 8000:8000 image-segmentation-system
```

4. Open `http://localhost:8000` on your browser to access the FastAPI application.

3.2 Data Management

3.2.1 Data Versioning with DVC:

Follow these steps to set up DVC for the project:

1. Install DVC, if not yet installed.

```
pip install dvc
```

2. Initialize DVC

```
dvc init
```

3. Track data files assuming your dataset is in a folder, named data.

```
dvc add data/
```

4. Push data to a remote storage

```
dvc remote add -d myremote <remote_storage_url>
dvc push
```

4 Model Development

1. We will build a ResNet-50 from scratch and train the model. Run the main.py file:

```
python main.py
```

2. Log Metrics with MLflow

5 Deployment

5.1 API Development with FastAPI

In the main.py file, we have API endpoints for user login, drift detection, and image segmentation. Run the API using Uvicorn:

```
uvicorn main:app --reload
```

5.2 Frontend with Plotly Dash

It provides an easy way to create a frontend interface for users to interact with the brain tumor segmentation service. We chose Plotly Dash as it is a powerful framework that is well-suited for data visualizations given that we will be using image data.

1. Create a plotly dash script, e.g app.py
2. Run the plotly dash application

```
pip install dash
python app.py
```

3. After running the command, you will see an output (http://127.0.0.1:8050/ or http://localhost:8050/).
4. Open your browser and paste the URL to access the dash application.

5.3 Database Integration:

The FastAPI application interacts with the MySQL database to manage patient data. It ensures that the database connection settings are configured correctly in the application.

6 Continuous Integration and Deployment (CI/CD)

We used Github actions to set up the workflow to ensure code quality and smooth deployment.

1. Set up GitHub Actions: Create a .github/workflows/ci.yml file with steps for building and testing the application.
2. Deploy to GCP

7 Monitoring and Maintenance

After a given period, we will use the images that were earlier collected and compare them to the recently uploaded ones for monitoring model performance. We will use "evidently" to detect data drift.

References

`https://neptune.ai/blog/mlops-architecture-guide`

`Class notes: https://github.com/AMMI-2024/mlops-course.git`