Assist MD - Product Requirements Document

Team name: High Voltage Society
Team lead: Brian Humphreys
Team Scribe: David Roster
Andrew Laux
Ram Malyala
Siddharth Malik

Table of Contents:

Assist - MD: Introduction **2**

System Architecture Overview **5**

User interaction and design **7**

Project Milestones and Objectives Outlined **8**
High-Level User Stories and Acceptance Criteria: 8
Lower-Level Use Cases - User Stories Broken Down: 10

Assist-MD Technical Implementation - Prototype Code **13**
Github: 13
Front End Assist-MD Web App - ReactJS: 13
Image Recognition Component - YOLOv2 Network Structure: 14
Amazon Web Services (AWS) Backend: 14
iOS interfacing with HVStream: 15

Technologies Employed **16**

Appendices **18**
What Assist-MD Does Not Do 18
Technologically: 18
Surgically: 18

Assist MD - Product Requirements Document

Assist - MD: Introduction

"The practice of Medicine, and in particular Surgery, is a continually evolving science. We speak of surgery as "being performed," as though it were a symphony. Technologies and techniques continue to emerge that improve outcomes, health results, and patient experiences.
I discern several critical values in the use of Assist MD algorithms."
- Dr. Brian Humphreys Sr.

Arthrex is a company which provides services, products, and technical innovation for medical professionals. Their enterprise research team has asked our group to implement a machine learning framework that is capable of processing and classifying video from their 4k surgical camera as well as any other cameras present in the operating environment. Specifically, they would like software that can, among other things, recognize and track medical personnel, recognize and track surgical tools.

Towards this end, we will gather test data which will be images of tools and medical personnel that we can use to train a computer neural network. The goal is to develop software that will be able to make real-time inference calls on images from operating room video and relay that information to medical personnel. Accurate processing of this data will allow us to solve a range of problems facing medical personal which he hope to explore. It will also allow us to generate post-surgery reports that will streamline important data collection and booking keeping in the operating environment that will allow personnel to focus on important tasks at hand.

There are many perspectives from which a product such as Assist-MD can be found vital, many of which mentioned in a written review by Dr. Humphreys. Dr. Humphreys believes Assist-MD can be found critical in the medical community in the following ways: teaching surgery, reducing surgical times, tracking key elements of surgery, use in critical surgery, evaluating new surgical techniques, interfacing with robotic surgeries, malpractice legal defense.

Medical Malpractice is a huge issue in the medical community which is the leading cause of death behind heart disease and cancer. To prove malpractice occurred during a surgery, an accuser must prove that a doctor provided inadequate care and an injury occurred with damaging consequences. With the implementation of Assist MD, both malpractice and defense attorneys can use footage to better support their claims. The footage will provide a basis for both party's claims because a doctor's past surgical history can be presented as evidence.

As medical class's size get more substantial, students are unable to get as much one on one time with the professor. Teachers are unable to give enough attention and time to students so

this lack of intimate learning pushes students to study techniques and procedures on their own. In an article titled “Teaching Surgery to Medical Students” by W Brian Sweeney, he states the disadvantage medical students currently have in today’s medical schools: “It is very clear that given the tremendous increase in medical knowledge, surgical technology, and intricate operative procedures, teaching surgery to medical students during this relatively short exposure has become an immense challenge.” Our increasing knowledge of the medical field has made it harder for students to keep up, but that is where Assist MD can make up the difference. Assist MD will provide more quantifiable metrics that students and professionals can judge their skills off and understand where they need to improve in. By providing increased analytics, Assist MD can help up and coming professionals build better technique and procedures for when they operate on a person in their future careers. By improving technical approach, Assist MD will also improve surgical times for students and medical practitioners. This extra time for students will allow teachers to spend other precious allotments reviewing concepts. Current doctors who see improved surgical times due to Assist MD can focus their time on other important matters of theirs.

The innovative features of Assist MD will see transformations in how we view surgery by developing new surgical techniques, advancements when performing crucial surgeries, and interfacing with robotics in the operating room. Currently, medical technology is advancing at a fast rate but proper documentation of new surgical techniques is releasing at a much slower rate. Assist MD’s live feed of a surgery where proper documentation is done at real time will provide enough evidence to release newer techniques to the industry than previously before. These new techniques can then be passed around to medical offices and be in practice quicker for critical surgeries where they could have a tremendous effect. Another positive of improving surgical times and tagging keys elements of a medical procedure is robotics surgeries already implement surgical cameras when conducting surgerys. By adding Assist MD to robotic surgeries, they will have lower risk of human error and produce a much safer outcome for the patient.

Assumptions that will be made are that the hospital has cameras in the surgery room as well as an account for our product. Doctors/medical staff must turn on each of the cameras and the cameras must be functional and positioned correctly to capture each feed. The timing of the cameras should synced. Another assumption is that surgical tools not on the table at the end of the surgery were used for the surgery. If tools are removed for any other reason (dropped by accident, misplaced, moved by a nurse etc.) then there will be an error as our service will assume those tools were used for the surgery.

Assist-MD will utilize video feeds that route to AWS S3 buckets and operate algorithms on said buckets in order to provide a live display of identified surgery equipment. This will be done by routing our S3 buckets to Sagemaker and running our algorithms on the data sent. We will have AI Models that are trained to recognize the instruments, and tag them as such. The output of the application will then be displayed on a web application with a slight delay to account for data transfer and processing times. At the end of a usage run, the program will generate an outline of the operation, showing tools, procedures, and holding the video for later use. Our algorithms can then be expanded to include personnel if time permits us, and potentially even identify people.

High Voltage Society is looking to create a product which will accurately identify medical instruments from a video feed, with the task being done as close to real time as possible. One of our main objectives is to first be able to upload and manipulate the data into our algorithms. Once this is done, we can easily run tests on our AI models and begin training in earnest. Eventually, we expect the models to be able to tell which instrument is which.

Our ultimate objective is to reach the state in which the AI models can differentiate between tools with a high accuracy. At this point, we will have a strong proof of concept and can begin streamlining and adding more features, such as the aforementioned Surgery room personnel identification. Completion of AI identification objective will result in the project itself being technically completed.

However, we wish to go above and beyond the requirements, and as such we have elected to add additional objectives to our project. One of these objectives is adding an end of run information generator. This involves having the application generate a file that contains helpful information about the surgery, such as most frequently used instrument or instruments that were not used at all. This will help surgeons review details about the surgery they performed and help prevent any complications with information that may arise.

We are also looking into the ability of identifying people in the surgery room. We wish to start by simply identifying the surgeon and staff in the operating room. Then perhaps look into being able to identify certain people using pre developed frameworks and libraries. However, this option is dependent on ethical considerations.

System Architecture Overview

The following UML design depicts where the flow of data starts and ends. Data is collected in real time from the three cameras of their respective scenes. For our proof-of-concept project, we have decided to make an iOS app as a substitute for the Web Proxy. The iOS interface will simply stream video to AWS where the data will get parsed into frames and feed into our YOLOv2 network. Once the network makes an inference on the individual pieces of data from incoming from the stream, it will then connect this data with the Web Interface which will provide the UI for Data and Metrics component. From this point, the data that is collected can be compiled into a PowerPoint and then verbally disseminated.

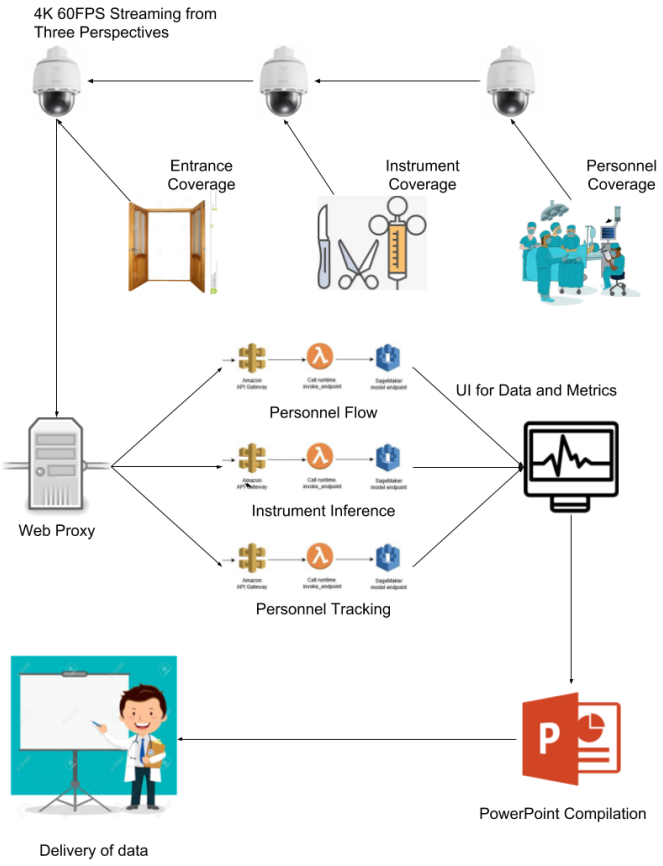


Diagram Legend:

Entrance Coverage: Camera-1 will be oriented towards the operating room entrance to track personnel flow into and out of the surgery

Staff Coverage: Camera-2 will be oriented towards the center of the operating room in order to track medical personnel around the room

Instrument Coverage: Camera-3 will be fixed on the treys where the surgeons place the instruments which not in use. The process of elimination will be used to figure out what instrument the doctor is using

Web Proxy: A router for the data; iOS app in our case

AWS Inferences: The three models (personnel flow, instrument inferences, personnel tracking) will receive stream incoming from the iOS app video stream and will make object inferences on those frames of data.

UI for Data Metrics: Once inferences have been made, the data from all three inferences will then be compiled into a time series and sent to the web interface to medical personnel usage.

PowerPoint Compilation: This is where the user will have the option to compile the main data points (i.e. maximum and minimum number of medical personnel involved at any given time, the

instrument used most frequently, the instrument used least frequently) into a PowerPoint for post-surgical evaluation and teaching.

User interaction and design

The starting point of user interaction in this project is the start of the video feed capture. Once the cameras are all turned on, they will immediately initiate video feed transfer to a web proxy that feeds the streamed data into AWS to be processed. All the clocks of the cameras are synced and data is sent with time stamps for the videos to be synced on a time series automatically.

Upon logging in, the user is prompted and a minimalist, monotonic screen to enter their credentials. When granted access, the user dashboard will appear. The initially opened tab will display aggregated company metrics which include:

1. Aggregated number of company surgery streams that have been processed and stored in their library
2. Aggregated number of total instruments used overall video streams with a break down of how many of each kind of instrument
3. Average number of medical personnel in each surgery
4. Average number of people who exit and enter the Operating Room in each surgery

The user will also be presented with a tab that will route them to a page where they can search video streams by some affiliated attributes. The full video feed (initially ordered chronologically according to upload) will appear as a column of bars that display the video metrics. Once clicked on, the bar expands into almost the full page and displays the three video streams in a time series. The videos can be manipulated by a youtube-esque time bar which moves all three videos' to the same corresponding time frame. Each video also displays the instruments captured by a box around drawn around the space it was detected, times a person leaves or exits, and when people shift positions.

A search bar is available on the column view of the video search page to make queries of videos based on the following attributes of each video stream:

1. The video title
2. Names of Instruments captured in the video
3. Number of Instruments captured in the video or number of personnel involved and captured in the video.

In the expanded video view that appears when a video bar is clicked, the user is also presented with the option to download a compiled version of the triple video stream capture which will be viewable in a PowerPoint.

Project Milestones and Objectives Outlined

High-Level User Stories and Acceptance Criteria:

1. As a user, if I capture video feed on the entrance coverage camera, then that data will be sent to Sagemaker so that it will be available to be trained by the personnel flow algorithm.

   Acceptance: Data from entrance feed is present on Sagemaker.

   Trello Cards:
   https://trello.com/c/CHqPPAMd/7-capture-video-data-and-tag-frames-for-people-leaving-and-entering-a-room

2. As a user, if I capture video feed on the instrument coverage camera, then that data will be sent to Sagemaker so that it will be available to be trained by the instrument inference algorithm.

   Acceptance: Data from instrument coverage feed is present on Sagemaker.

   Trello Card:
   https://trello.com/c/s12EDxfp/8-capture-video-data-and-tag-frames-of-specific-medical-instruments

3. As a user, if I capture video feed on the personnel coverage camera, then that data will be sent to Sagemaker so that it will be available to be trained by the personnel tracking algorithm.

   Acceptance: Data from personnel coverage feed is present on Sagemaker.

   Trello Card:
   https://trello.com/c/s12EDxfp/8-capture-video-data-and-tag-frames-of-specific-medical-instruments

4. As a user, if I train entrance feed data using the personnel flow model, then the model will determine whenever staff enter or exit the room so that the flow of staff in the room can be recorded.

   Acceptance: Model outputs correctly if staff enter or exit the room whenever the event occurs.

Trello Card:
https://trello.com/c/3vbzi1Ot/4-use-and-train-open-cv-algorithm-to-figure-out-when-someone-is-leaving-or-entering-a-room

5. As a user, if I train instrument coverage data using the instrument inference model, then the model will determine which tool was used at a given time so that the use of that tool is recorded.

   Acceptance: Model outputs correctly which tool is used and a confidence level, whenever a tool is used.

   Trello Card:
   https://trello.com/c/3vbzi1Ot/4-use-and-train-open-cv-algorithm-to-figure-out-when-someone-is-leaving-or-entering-a-room

6. As a user, if I train personnel coverage data using the personnel tracking model, then the model will determine which staff members were in the room at a given time so that that information is recorded.

   Acceptance: Model outputs correctly which staff are in the room at any given time.

   Trello Card:
   https://trello.com/c/amu1O5xt/6-train-open-cv-library-to-track-unique-personnel-around-a-room

7. As a user, if the personnel flow model determines that staff entered/exited the room, then I will see that occurrence in the UI so that I am aware of that event.

   Acceptance: Staff entering/exiting the room is displayed in the UI.

   Trello Card:
   https://trello.com/c/8nhMBcWK/10-build-web-interface-to-request-personnel-flow-data-from-sagemaker-backend

8. As a user, if the instrument inference model determines that a tool was used, then I will see that tool in the UI so that I am aware that the tool was used.
   Acceptance: Whenever a tool is used, it is displayed in the UI along with a confidence level.

   Acceptance: Instrument inferences are recorded on a time series and displayed in the UI

   Trello Card:
   https://trello.com/c/8nhMBcWK/10-build-web-interface-to-request-personnel-flow-data-from-sagemaker-backend

9. As a user, if the personnel tracking model determines particular staff members to be in the room, then I will see that information in the UI so that I am aware of the personnel in the room at a given time.

   Acceptance: At any given time, all the staff in the room are displayed in the UI.

   Trello Card:
   https://trello.com/c/pZTIOzbM/12-build-web-interface-to-request-personnel-tracking-of-medical-staff

10. As a user, if data is outputted by any of the models, then I will be able to get a powerpoint containing that data so that I will have that information available for records.

    Acceptance: Whenever requested, a powerpoint can be downloaded from the web service containing all the information output by any of the models.

    Trello Card:
    https://trello.com/c/js3OD4Cp/13-programmatically-compile-most-relevant-information-into-a-power-point

11. As a user or developer, I want to be able to directly upload a video of selected instruments or the surgeon room, and be able to send this video to sagemaker for quick results out of the AI system.
    Acceptance: Whenever activated, the data will be available on amazon web services and executes the selected model.

    Trello Card:
    https://trello.com/c/5jzjQNjP/14-upload-video-to-aws-from-a-mobile-device

Lower-Level Use Cases - User Stories Broken Down:

1. Use Case: Send Data from HVStream to Amazon S3
   Actors: HVStream App, User, AWS S3 servers
   Precondition: User has opened the app
   Postcondition: Data is uploaded to S3 server and is available for manipulation
   Flow of Events:
   1. User selects their server. Application updates all variables
   2. User selects video or picture
   3. Application sends data to the selected server.

   Alternative Paths:
   1. Specific server is offline. Data will fail to send.

   Trello:
   1. https://trello.com/c/AydpirK6/15-send-video-feed-data-in-bulk-to-aws

   GitHub:

1) https://github.com/brianhumphreys/Assist-MD/commit/821606ff3f570148aaccc60ee06175462faf4d05

2. Use Case: Send live video data from HVStream to Amazon S3
   Actors: HVStream App, User, AWS S3 Servers
   Precondition: User has opened the app
   Postcondition: Data is live streamed and uploaded to the S3 servers, ready for real time manipulation by the Assist-MD algorithms.
   Flow of Events:
   1. User selects their server and clicks live stream button
   2. User begins live streaming.
   3. Stream is uploaded directly to AWS servers.

   Alternative Paths:
   1. Specific Server is offline. Data will fail to send and app will stop the stream.

   Trello:
   1. https://trello.com/c/9HYVxrRh/26-get-video-feed-to-be-a-continuous-real-time-stream-sent-from-ios-app-to-aws-to-be-inferred-frame-by-frame

3. Use Case: Receive GET response from API Gateway with video data
   Actors: Assist-MD Web App, User, AWS S3 Servers, AWS API Gateway, Axios library, AWS Lambda
   Precondition: User has logged in with AWS credentials
   Postcondition: Page is loaded with relevant video feed
   Flow of Events:
   1. Upon user logging in, app state is updated with user AWS credentials
   2. In ComponentDidMount() function, use Axios.get() to hit AWS gateway with path /home
   3. /home API call triggers video in relevant S3 buckets to be formatted in Lambda function and sent in request body
   4. Assist-MD app state is updated with array of videos

   Alternative Paths:
   1. There are no videos in the S3 bucket
      a. A null value is presented to the lambda function
      b. Lambda sends back null array in request body

   Trello:
   1. https://trello.com/c/5YvGQCZj/30-configure-api-gateway-to-handle-root-get-requests-and-to-return-either-the-video-or-a-link-to-the-video
   2. https://trello.com/c/qqrsCuRS/29-using-axios-library-send-a-get-request-upon-entering-the-page-to-get-and-videos-from-an-s3-bucket-and-store-on-site

4. Use Case: If videos are available in S3 buckets, add the video thumbnail to a surgery card
   Actors: Assist-MD Web App, User, React
   Precondition: Request from AWS has been received
   Postcondition: Surgery Card components are loaded with video thumbnails
   Flow of Event:
   1. If the Assist-MD app video state updates, create a surgery card component for each video that is included in the app state

2. If there are too many surgery cards to fit on the page, app will incorporate scroll component

Alternative Paths:

1. If there are no videos in the app state, no surgery cards will appear on the page

Trello:

1. https://trello.com/c/1euZluef/31-once-a-response-is-returned-with-a-video-add-the-thumbnail-to-a-surgery-card

5. Use Case: Create expandable surgery cards
   Actors: Assist-MD Web App, User, React
   Precondition: There are live surgery cards existent on the Assist-MD site and user is logged in
   Postcondition: Surgery card expands to full view and video thumbnail extends to a full video window
   Flow of Event:
   1. Upon clicking a Surgery card, surgery metrics are accessed and passed into the Full View component

   Alternative Paths:
   1. If no metrics or only faulty metrics are available, display nothing

   Trello:
   1. https://trello.com/c/WEvKdTPQ/32-make-videos-expandable-upon-clicking
6. Use Case: Configure AWS Cloudfront to stream on-demand videos in our S3 bucket through HTTP requests
   Actors: Assist-MD Web App, User, AWS MediaConvert, AWS MediaLive, AWS MediaLive
   Precondition: There is either a video file uploaded into the S3 bucket or a surgery is currently being recorded and is connected to an AWS account. Also the Full View window must be active for the video to play
   Postcondition: The video of the surgery which has either happened or is currently happening will be streamable from the Assist-MD Web App
   Flow of Event:
   1. Once the play button is clicked, an embedded URL will become active that will hit an endpoint set up by AWS CloudFront and video in the format of HLS will be delivered to Assist-MD

   Alternative Paths:
   1. If data is not sent properly, an error will display on screen

   Trello:
   1. https://trello.com/c/aBp6yrwy/35-configure-aws-cloudfront-to-stream-hls-video-from-video-files-stored-in-s3-buckets
7.
8.
9.
10.
11.
12.
13.
14.
15.
16.
17.

Assist-MD Technical Implementation - Prototype Code

Github:

Our Github currently features the prototype of the Assist-MD Web App and the iOS App that records and streams video data to an AWS S3 bucket. More to be added to this soon:

https://github.com/brianhumphreys/Assist-MD/tree/feature_HVStream

Front End Assist-MD Web App - ReactJS:

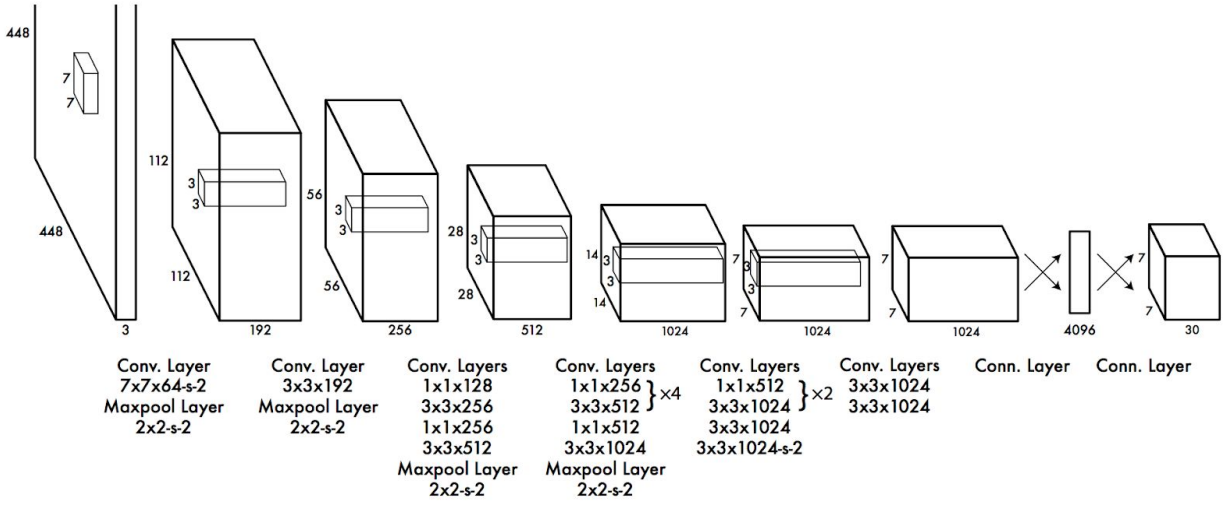
Figure 1: Folder Structure

React-JS Web Interface - The app prototype has been built using standard ReactJS formatting with component and container folder structure. The three individual page components shown in the image are instantiated in the container class and rendered in the App.js file. The Post and Full Post will be remodeled into Surgery Cards and Surgery Full View, respectively and the New Post components will deleted. The blog container will be converted to the Assist-MD container.

```
import React, { Component } from 'react';
import axios from 'axios';

import Post from '../../components/Post/Post';
import FullPost from '../../components/FullPost/FullPost';
import NewPost from '../../components/NewPost/NewPost';
import './Blog.css';

class Blog extends Component {
    state = {
        posts: [],
        selectedPostId: null
    }

    componentDidMount() {
        axios.get('https://jsonplaceholder.typicode.com/posts')
            .then(response => {
                const posts = response.data.slice(0, 4);
                const updatedPosts = posts.map(post => {
                    return {
                        ...post,
                        author: 'Max'
                    }
                })
                this.setState({posts: updatedPosts})
                //console.log(response);
            })
    }
```

Image Recognition Component - YOLOv2 Network Structure:

For the image recognition portion of Assist-MD that will make inferences of the medical equipment and personnel, the High Voltage Society has decided to implement an instance of Joseph Redmon's YOLOv2 network to make multiple object inferences in a single picture. The below figure describes the structure of the YOLO network which implements some layers of convolution.



Amazon Web Services (AWS) Backend:

Lambda Function - invokes SageMaker's Endpoint API. Extracts data from a web request, feeds raw sample data and invokes the SageMaker model on the data. In this example, we took in as input, data from a SageMaker image recognition inference and formatted the data to a desired format before sending the data back through an HTTP response. He will use Lambda functions for many things in this project, such as watch S3 folders for change and for publishing to S3 folders when videos are being uploaded.

lambda_function

```
import os
import io
import boto3
import json
import csv

# grab environment variables
ENDPOINT_NAME = os.environ['ENDPOINT_NAME']
runtime= boto3.client('runtime.sagemaker')

def lambda_handler(event, context):
    print("Received event: " + json.dumps(event, indent=2))
    # print("this is the event", event)
    data = json.loads(json.dumps(event))
    payload = data['data']
    print(payload)

    response = runtime.invoke_endpoint(EndpointName=ENDPOINT_NAME,
                                       ContentType='text/csv',
                                       Body=payload)
    print(response)
    result = json.loads(response['Body'].read().decode())
    print("result", result)
    # pred = int(result['predictions'][0]['predicted_label'])
    pred = int(result['predictions'][0]['score'])
    predicted_label = 'Malignant' if pred == 1 else 'Benign'

    return_message = "Tumor is " + predicted_label
```

iOS interfacing with HVStream:

Direct and quick video upload system to aws for use with iphone. Three different cameras can be at play at a time on the same AWS account for surgical scene inferences. The data captured by the iPhone cameras will be streamed into an S3 bucket, fed into our YOLOv2 network to be processed and inferred upon, sent back to S3 and then streamed from Assist-MD Web App.

```
    func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [String : Any]) {
        print("Image picker finished")

        selectedVideo = info[UIImagePickerControllerMediaURL] as! NSURL
        print(selectedVideo)
        do{
            try self.videoData = NSData(contentsOf: self.selectedVideo as URL)
            //let myReader = AVAsset(url: self.selectedVideo as URL)
            //print(videoData)
            let credentialsProvider = AWSStaticCredentialsProvider(accessKey: accessKey, secretKey: secretKey)
            let configuration = AWSServiceConfiguration(region: AWSRegionType.USWest1, credentialsProvider: credentialsProvider)
            AWSServiceManager.default().defaultServiceConfiguration = configuration
            let url = self.selectedVideo as URL
            let remoteName = "test.mov"
            let S3BucketName = "hvstream1"
            let uploadRequest = AWSS3TransferManagerUploadRequest()!
            uploadRequest.body = url
            uploadRequest.key = remoteName
            uploadRequest.bucket = S3BucketName
            uploadRequest.contentType = "video/movie"
            uploadRequest.acl = .publicRead

            let transferManager = AWSS3TransferManager.default()
            transferManager.upload(uploadRequest).continueWith(block: {(task: AWSTask<AnyObject>) -> Any? in
                if let error = task.error{
                    print("Upload failed with error: (\(error.localizedDescription))")
                }
                if(task.result != nil){
                    let url = AWSS3.default().configuration.endpoint.url
                    let publicURL = url?.appendingPathComponent(uploadRequest.bucket!).appendingPathComponent(uploadRequest.key!)
                    print("Uploaded to:\(publicURL)")
                }
                return nil
            })



        } catch {
            print("Error")
        }

        self.dismiss(animated: true, completion: nil)
    }
    @IBAction func KeyChange(_ sender: Any) {
        switch UserSelection.selectedSegmentIndex {
```

Technologies Employed

AWS API Gateway - A console that can easily configure and deploy API endpoints and give your website's functionality unlimited scalability with little work or maintenance

AWS Lambda - A snippet of code that is spun up through AWS by an event trigger. The code takes event data as input and returns the desired output before the function instance is deleted from the servers and then saved as an image.

AWS Sagermaker - Amazon's service for training and deploying custom machine learning code. Includes pre-built machine learning code to solve specific problems as well as support for frameworks like PyTorch, Tensorflow, Apache Mxnet. Currently, we are planning to use Tensorflow. Training as deploying models is resource intensive process, SageMaker provides access to high-performance cloud computing

AWS S3 - Amazon Simple Storage Service is storage for the Internet. It is designed to make web-scale computing easier for developers. Amazon S3 has a simple web services interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web.

AWS Cloudfront - Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users. CloudFront delivers your content through a worldwide network of data centers called edge locations. When a user requests content that you're serving with CloudFront, the user is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance.

AWS MediaConvert - AWS Elemental MediaConvert is a file-based video transcoding service with broadcast-grade features. It allows you to easily create video-on-demand (VOD) content for broadcast and multiscreen delivery at scale. The service combines advanced video and audio capabilities with a simple web services interface and pay-as-you-go pricing. With AWS Elemental MediaConvert, you can focus on delivering compelling media experiences without having to worry about the complexity of building and operating your own video processing infrastructure.

AWS MediaLive - AWS Elemental MediaLive is a broadcast-grade live video processing service. It lets you create high-quality video streams for delivery to broadcast televisions and internet-connected multiscreen devices, like connected TVs, tablets, smart phones, and set-top boxes. The service works by encoding your live video streams in real-time, taking a larger-sized live video source and compressing it into smaller versions for distribution to your viewers. With AWS Elemental MediaLive, you can easily set up streams for both live events and 24x7 channels with advanced broadcasting features, high availability, and pay-as-you-go pricing. AWS Elemental MediaLive lets you focus

on creating compelling live video experiences for your viewers without the complexity of building and operating broadcast-grade video processing infrastructure.

Axios - Axios is a very popular JavaScript library you can use to perform HTTP requests, that works in both Browser and Node.js platforms. It supports all modern browsers, including support for IE8 and higher. It is promise-based, and this lets us write async/await code to perform XHR requests very easily.

Docker - Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.

OpenCV - OpenCV (Open Source Computer Vision Library) is a powerful open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

ReactJS - In computing, React (also known as React.js or ReactJS) is a JavaScript library[2] for building user interfaces. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. Complex React applications usually require the use of additional libraries for state management, routing, and interaction with an API.

Sony SNC ep580 - Wall mounted camera for use in operating rooms. Provides 4k resolution. Arthrex has provided our team with one in the hopes that we can use it to generate test data that accurately simulate real-world use.

Swift - Swift is a programming language that was created to be extremely safe and fast. It is utilized in various mobile and cloud platforms, including all of apple's products and software, which is where we will be utilizing it.

Tensorflow - TensorFlow™ is an open source software library for high-performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

UHD4 - Arthrex's 4k imaging platform. Includes image processing unit and light source unit for the 4k camera. Controlled remotely via android tablet. This is the unit which supplies video from the surgical scene that will be processed with our trained models.

Appendices

What Assist-MD Does Not Do

Technologically:

Assist-MD (AMD) has many implications attached to it upon reading the Requirements sheet. One might think upon learning that AMD tracks medical personnel, that we will be building profiles of individuals. All data received about a medical personnels facial structure, eye color, build, etc. will be discarded. In a sense, every surgery will build its own environment of personnel characteristics and instrument usage.

The algorithm will not be able to directly make instrument inferences of instruments while they are in use. It is assumed that that the camera making instrument inferences will not be oriented in the surgeons perspective but at the instruments not in use. From this data, process of elimination will be used to figure out the instrument in use.

Surgically:

Although AMD will provide metrics of each surgery, no high-level analysis will be made on the surgery. As an example, AMD will be able to figure out the instruments that were used the most and the least and will also be able to know how many people were in the room at any given time, but it will not be able to give the surgeon advice on what instruments to use in the future or how to improve his performance. These high-level analyses will be left up to human intelligence for now.