CS CAPSTONE PROJECT ARCHIVE

JUNE 4, 2020

GESTURE RECOGNITION USING NEW INTEL REAL SENSE LIGHT CODED CAMERA

PREPARED FOR

INTEL

EDUARDO X. ALBAN _____ *Signature* _____ *Date*

SATOSHI SUZUKI _____ *Signature* _____ *Date*

PO-CHENG CHEN _____ *Signature* _____ *Date*

PREPARED BY

GROUP 33

GESTURE RECOGNITION

JONATHAN HULL _____ *Signature* _____ *Date*

NICHOLAS DAVIES _____ *Signature* _____ *Date*

SHANE CLANCY _____ *Signature* _____ *Date*

SHIHAO SONG _____ *Signature* _____ *Date*

ULISES ZARAGOZA _____ *Signature* _____ *Date*

ZHIDONG ZHANG _____ *Signature* _____ *Date*

Abstract

This is the project archive document which will contain all of documentation and instructions we have worked on as a team over the past year. This document contains an overview of our project, the requirements document, the design document, tech review from each member of the team, weekly blog posts, our poster for the virtual showcase, project documentation for code setup and running the project, recommended technical resources, and project reflections.

CONTENTS

1 Introduction to Project

1.1 Project Background

The project was requested to us by Intel employees, Eduardo Alban, Satoshi Suzuki, and Po-Cheng Chen. It was requested for the purpose of creating a project centered around the utilization of their employer's depth sensing camera, called the Intel RealSense SR305 camera. The importance of this project was that our group sought to create a cost-effective solution in the domain of gesture translation. Specifically, we looked to utilize the RealSense camera to perform gesture translations within the domain of the American Sign Language (ASL) alphabet letters. The goal is was to create a software capable of using the RealSense cameras to capture user input in the form of ASL gestures, and then utilize machine learning methods to classify the user's input by the specific letter that was gestured from the ASL alphabet. We aim to tackle accessibility issues by creating a user friendly software potentially capable of breaking down communication barriers by creating the capability to turn ASL gestures into readable text that is more readily understood by a wider variety of people.

1.2 Project Development Information

Our team was comprised of the following individuals: Ulises Zaragoza, Shane Clancy, Jonathan Hull, Nic Davies, Zhidong Zhang, and Song Shihao. In order to tackle this project we split up into three major teams according to separate project components, the Machine Learning (ML) team, the Software (SW) team, and the User Interface (UI) team. The disbursal of members across development teams was as follows: ML team - Ulises Zaragoza & Zhidong Zhang, SW team - Shane Clancy & Nic Davies, UI team - Jonathan Hull & Song Shihao. The role that the clients played in our project was crucial to the overall planning and development of our project. Although they did not directly contribute to any developmental efforts, they helped us create the team division for tackling the project, and also constantly provided constructive feedback during each iteration of the developmental cycle. We had weekly reports where each development team reported the progress we had made, any areas we got stuck, and plans moving forward. These weekly meetings helped our team stay on track towards end of term goals, as well as ensured the product we were developing adhered to the expectations of our clients.

1.3 Spring Term Changes

The changes during spring term only slightly affected the deliverables of our group. By the time spring term came along, our project was nearly finished in completion with only minor UI variations and code optimizations left to be tackled by our group. Throughout the duration of winter and fall term, our team had already began implementation in remote settings, and the split of our team into separate feature teams really helped aid in remote development for our group. Each team had their own responsibilities to accomplish, and the majority of project integration had already been completed. The main effect that the changes this term caused our group is the inability to spread cameras across the team. Our clients provided our team each with a total of 3 cameras, one for each feature team. The spring term changes did not allow us to share the cameras with the entire team, and also greatly limit our abilities to have other individuals test our project without the RealSense Camera. This was the main effect encountered by our group, but thankfully we were able to complete all planned development.

1.4 Future Work

This document may be used to pick up our remaining work by first getting users up to speed on the development made so far, and the decisions backing up those developmental ideas/plans. Users may review the technical resources for learning more about background work if they feel the need to do so. Users may also refer to the project documentation section for further detailed assessments of how to interact with and run our project. The main steps moving forward for our group, mainly consists of the developmental feat in implementing a model capable of a multiple frame recognition. The current implementation only supports the classification of single frame images, however the next step would be to move into the domain of multiple frame (video) recognition of ASL gestures. This expands the domain of the project significantly, going from a restriction of only being able to accomplish ASL alphabet recognition, to a broader range of words/phrases from the ASL domain. Unfortunately time was not on our side and we were unable to implement these advanced features, but they would indeed be nice additions to the project.

2 REQUIREMENTS DOCUMENT

2.1 Introduction

2.1.1 System Purpose

Our team will collaboratively work to create and train a machine learning (ML) algorithm with the task of gesture recognition using the Intel RealSense Depth Camera SR305. In addition our goal is to build a Graphical User Interface (GUI) application to house the model and utilize it. The machine learning model will be expected to have the ability to classify American Sign Language (ASL) gestures from a single image frame that is provided by the RealSense Camera.

2.1.2 System Scope

The system scope of our project has an end goal of having a ML be able to classify sign-language gestures from the ASL domain, and present this classification via a text message to the user through the housing GUI application. This GUI application will allow interface capabilities with the RealSense camera, and will be a direct source of input to the ML model. Our project will take on a much smaller initial scope, as we seek to first build and train a foundation ML model that can perform basic gesture recognition tasks from just a few letters of the domain. Once this has been established, we will work to fine-tune the model and prepare it to handle our end-goal task of sign-language recognition of all letters, excluding the letters J and Z because of their need for multiple frame recognition.

2.2 System Overview

2.2.1 System Context

The camera we will be using is the Intel RealSense Depth Camera SR305. This indoor camera has an incredible quality of depth, specifically when used at the ideal one meter distance. The code written for this device is easily transferred to utilize on other Intel RealSense gadgets, making it user friendly. This camera is also very cost effective, making the product affordable for a wider range of customers.

ML Algorithms are a branch of artificial intelligence that allows computers to learn for themselves. ML Algorithms can be supervised or not, allowing them to learn for themselves or have strict guidance depending on the application. In this context, our ML Algorithm will be supervised, and is going to learn to recognize patterns through visual recognition and coded light in the training frames that we provide. This will then allow the system to classify new image frames of letters from which it has been trained.

Coded light systems, a tool utilized by RealSense cameras, interpret the depth and distance by using infrared light. The infrared light is a single pixel that is cast onto the object to see if the pixel lands on a dark or light stripe. Then, a binary code is created for each pixel. This data is then recorded and memorized on the device to help it interpret certain gestures. The Intel RealSense Depth Camera SR305 has a 640x480 depth resolution as well as 60 frames per second, making for quality video/image capturing capabilities[5].

2.2.2 System Functions

Our team will be utilizing the SDK provided by Intel on GitHub which will allow us to interface with the camera and use the camera's features. The way we will be implementing our ML Algorithm is through transfer learning approach

which is using a pre-trained ML Algorithm as a baseline classifier, and will then train the task of gesture recognition from this foundation.

2.2.3 User Characteristics

The individuals we are targeting with this software are those with language barriers. Our goal is to make a software that allows language-disabled people to communicate smoothly with the movement of their hands. We hope to tackle the barrier of interpreting sign language for those not familiar with the gestures, and furthermore to achieve the goal of helping people with language disabilities communicate with the outside world.

2.3 System Requirements

2.3.1 Functional Requirements

The following sections outline the functional requirements that this project needs to meet. We will also evaluate how we will measure these functional requirements in order to meet the needs and expectations of the clients. We will be measuring the functionality of the Intel RealSense camera, machine learning model, and the Graphical User Interface.

2.3.1.1 Intel RealSense Camera: The Intel RealSense camera is a light coded camera that we will be using the capture data for our project. The software for this component has already been developed in an open source environment by Intel. The camera needs to be attached to the computer through USB 3.1, then the Intel RealSense SDK 2.0 has the capability to find and read input of the Intel RealSense camera itself. Our project repository will always include the proper SDK that is compatible with the Intel RealSense SR305 Camera. We will be measuring the functionality by making sure that the depth of certain objects is measured correctly through manual measurements.

2.3.1.2 Machine Learning Model: The machine learning model is the brain of the project, and will need to interact with the Intel RealSense camera, as well as the GUI application. We will be doing our development for the machine learning model using Python3 and the PyTorch library, which need to be installed in order for the model to be able to train and classify information that is provided to it through the data storage that we will setup to store our information. Measuring the functionality of the machine learning model will be done after the model is developed and trained. We hope to achieve high accuracy with our model, and we will be measuring its accuracy throughout the development on a set of data that we set aside for testing. The metrics by which we will test our model's performance is through a confusion matrix, which measure the per-class accuracy of the classifications made by the model.

2.3.1.3 Graphical User Interface: The graphical user interface will be in the form of an executable that should be deployable on all platforms. We will need to make sure that this application can be deployed on each platform before we release the project. The user interface needs to be accessible to people who lack verbal communication, and customizable to their needs. Consistent manual testing on the interface needs to be done to ensure the usability meets these standards.

2.3.2 Performance Requirements

The following sections outline the speculative performance requirements for our project's chosen ML Model and the GUI application our team will create to house and utilize the model. See the Verification section for information on how our group will ensure performance criteria is met.

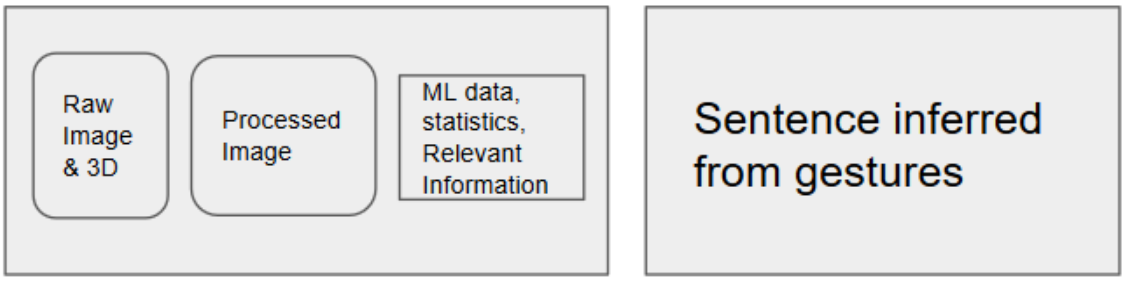
2.3.2.1 Machine Learning Model: Our ML model will be expected to classify single frame gestures from the American Sign Language alphabet, excluding the letters J and Z because of their need for multiple frame recognition due to the movement needed to sign the two letters. No quantitative metrics have been established on the amount of

accuracy in which the model will be able to perform these recognitions, however it is expected to have sufficient training data across all planned classification letters in order to appropriately train the model. Our group is expected to utilize the output from the confusion matrix to determine how well our model is at classifying the letters, and will help dictate if more data or training is needed for our model.

2.3.2.2 Graphical User Interface: The performance requirements expected of our GUI include the ability to securely transfer RealSense image feed output as input to the ML model of our project. Our encasing GUI must be able to securely and dependably read input from the RealSense camera, without communication issues between the software and hardware. Furthermore, it is expected that our model's classification is output in a reasonably quick manner. Exact time specifications may be detailed upon deeper experimentation with RealSense cameras and related SDK's.

2.3.3 System Interface

The design requested by the client is to provide user with two different screens. The first screen is used to capture/present data from the camera. Within the first screen there will be three components displayed to the user: an RGB and 3D image, the filter-processed image, and relevant ML interpretation information information (e.g. accuracy, statistics, etc.). The second screen is used to display the ML gesture recognition in its string literal form. A representation of this visual using the two screens can be seen directly below:



This layout will display to the user all relevant information that is calculated from our project. Beginning with the raw input they give to the camera, and ending with the final recognition made from our ML algorithm. This UI layout captures all parameters of our project, and will effectively allow a user to input gestures to the camera, and receive the ML classification.

2.3.4 Environmental Conditions

The GUI Environment should be on the development machine the client chooses. If no machine is given, then the development location is subject to change. The GUI should be able to run on Windows 10, IOS, and Linux. The environment on which it runs should have internet access and enough processing power in order to properly run all system components and program requirements.

2.3.5 Information Management

The information most pertinent to our group's project is the training data we will use as input for our chosen ML model. The management of this data will involve identifying a candidate storage method followed by beginning the process of collecting images to use as training/test/validation input for our ML model. Due to the high importance of obtaining plentiful and quality forms of training data, the process of collecting data itself will also take on a form of

management. This form will involve a strict set of guidelines detailing which types of gestures we are obtaining, image quality, number of different gesture iterations, involving diverse participants, and an equal number of training images across the separate labels. Our group has decided upon 500 training images for each letter we will be classifying. The letters to capture will be spread across all members of the team to ensure diversity in training examples.

2.3.6 Policies and Regulations

For work completion, all group members are in accordance with the given work assigned. If the work completed is code-based, then a peer-review has to be completed prior to submission. If the work completed is text-based, then at least two other group members have reviewed the document. Each process should establish an outline at the beginning and the task should meet all requirements specified by the outline. Work should sufficiently accomplish task at hand. Work should be well thought out with clearly fleshed out ideas. Meeting attendance, is required for every group member unless properly communicated, or unforeseen circumstances occur. If we encounter problems, it is best to solve them internally. If we can't solve them, then we will escalate as needed.

2.3.7 Packaging, handling, shipping, and transportation

The cameras were initially sent to one group member's home and have been secured. We have divided the cameras amongst the team members, and plan to give equal access throughout the first term so we may all familiarize ourselves with the equipment. Equal equity among camera access is essential for proper understanding of project completion and components.

2.4 Verification

2.4.1 Functional Requirements

The following sections outline the verification techniques our group will act on in order to make sure all components of the project are functional.

2.4.1.1 Intel RealSense Camera: The camera needs to cover many edge cases that could be encountered based on the depth calculations and bad IR sensor data. Although the product has been thoroughly tested by Intel, we will need to make sure that the camera can obtain edge case data for certain gestures that we want to classify. While our group is gathering data, we hope to find new edge cases to explore with the camera. This will help the accuracy and training of the machine learning model directly.

2.4.1.2 Machine Learning Model: The machine learning model will need to be regularly tested in different ways. Once a model is built, we will verify feature importance and relevance, feature thresholds, and feature relationship with output. Some features might not be relevant to classifier, and can be pruned accordingly to increase speed of training and advancement of the model. Feature thresholds are bounds for specific features, and if features can specifically fall within a range, this can be analyzed and a multi-layer model can be developed in order to account for these ranges relating to specific output that is wanted. As a group we will need to continually verify that features are being given different priorities in order to create an accurate model.

2.4.1.3 Graphical User Interface: The graphical user interface needs to go through manual testing, by many different people. This will make sure that the user interface is intuitive and usable by different types of people with different goals for the program. The user interface needs to be customizable to different needs, and any specific need that is not met, should be addressed during verification steps of new feature introduction.

2.4.2 Performance Requirements

The following sections outline the speculative verification techniques/guidelines our group will implement in order to ensure the validity of the performance requirements established above.

2.4.2.1 Machine Learning Model: An mentioned prior, the metrics by which we will measure the performance of our ML model is by testing the performance on a set of data that is completely independent of the set of data used to train it. The specific metric will be done via confusion matrix, which measure the per-class accuracy of the classifications made by our model. This will give our group a solid idea of what classifications are being properly done by our model, and which labels will require more training data increase overall accuracy.

2.4.2.2 Graphical User Interface: The main form of verification for performance of the GUI application, which houses the ML model, will come through various forms of testing of the application. Usability testing, unit testing, regression testing, and other forms, will be utilized on our GUI application to ensure its quality in performance and functionality. Furthermore, this will help our group ensure that proper communication is occurring through our entire project's pipeline. Starting from the input received by the camera, followed by assurance that the RealSense video feed is being properly fed into our ML algorithm. Lastly, tests will help ensure that the output being displayed by the GUI is true to the output produced by the ML model. The various forms of testing conducted by our group on the housing GUI application will aid in ensuring the performance quality of the entire project, not just solely the GUI application.

2.4.3 System Interface

Users will be able to consistently open the GUI application to interact with the RealSense camera, and the housed ML model. Apart from the rigorous manual testing our group will perform for verification of performance and functional requirements, specific manual tests will be performed to ensure the quality of the usability for our GUI application. We will be looking to establish metrics such as ease of use, and specifically we hope to address issues that could arise for users with specific accommodation needs.

2.4.4 Environmental Conditions

The system interface should be able to run on popular operating systems. Anything from Windows 10, IOS, and Linux. The focus will be on whatever development machine we are tasked with building this on. The machine will need Internet connection, ports to connect to the camera, processing power to run the machine learning model, and access the database. The interface can also be made to be portable such as exporting it to a Raspberry Pi or Arduino to run if the group decides that this is its end state. The database will need to have an adequate amount of space to store training videos. It is important that the computer running this interface is not hindered by its processing power to avoid unnecessary delays.

2.4.5 Information Management

Ensuring quality and an adequate quantity of training data is vital to the success of our group's project. Therefore, a formal team meeting needs to take place in order to detail the guideline set and timeline for our project's data goals. Furthermore, a form of data storage needs to be decided amongst our group members and clients, with the idea in mind that we are going to potentially need a large storage room to store videos as our training/test/validation sets. Once this meeting has taken place, along with the specification of guidelines, timelines, and data storage methods, this document may be signed off.

2.4.6 Policies and Regulations

The policies and regulations of this system interface should reflect those of our client and Oregon State University College of Engineering. This project should aim to keep anonymity, as well as privacy of any individuals caught in the camera's lens. Gesture collection should only be used for maintaining the accuracy of the machine learning model and keeping and up to date database. If this project is used commercially, then this section is subject to change for its correct purpose.

2.4.7 Packaging, handling, shipping, and transportation

The only package-able items for this interface would be the GUI. The machine learning model and the database can be remotely connected to. The could should either be able to be zipped up or tarred to another machine that can access the machine learning model and the database. Transportation may vary from email, cloud drives, or any sort of transferring service. Shipping is not applicable to software unless the interface becomes too large to send over the Internet.

2.5 Appendices

2.5.1 Assumptions

- This document is a fluid requirements list created by the group members in collaboration with the project clients. Future meetings and work will help cement details within this document that reflect the needs of both the group and the client. This document is subject to change.

2.5.2 Dependencies

- The Intel RealSense SDK will be used by our group to interface with the camera and receive video feed [6].

2.5.3 Acronyms and Abbreviations

- SDK: Software Development Kit
- ML: Machine Learning
- GUI: Graphical User Interface
- CNN: Convolutional Neural Network

2.5.4 Gantt Chart Timeline

Figure 1 below details the tentative timeline for group's projected milestones. This timeline currently displays only high level goals and is anticipated to change once finer project details have been firmly established.

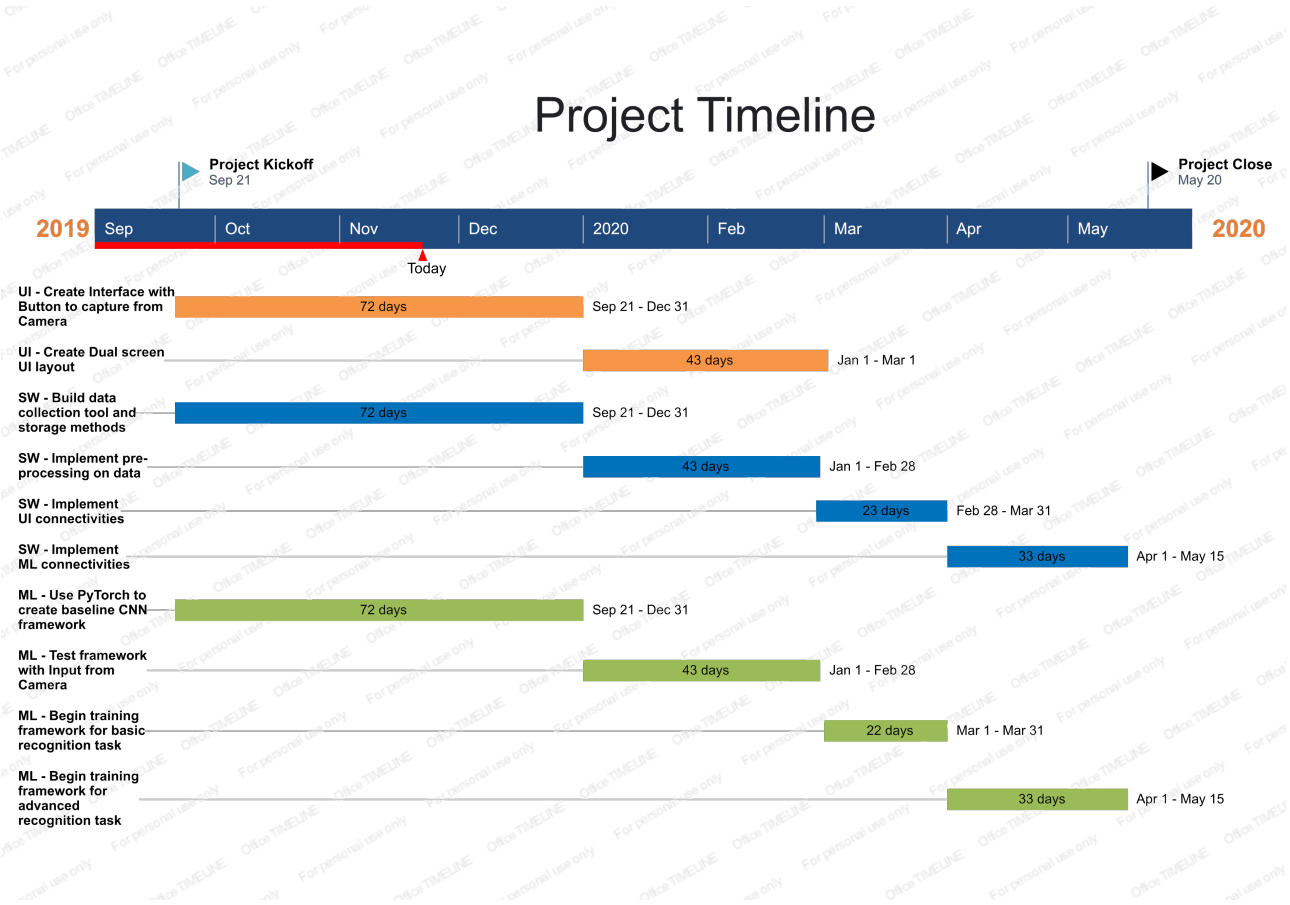


Fig. 1: Gantt Chart

2.5.5 Table of Changes

Below is a table describing the changes that took place from the original planning in Fall and Winter term.

Change Table	
Project Component	Changes made
Machine Learning	The main changes that have occurred on the Machine Learning portions of the project include the fact that our project scope has changed from video recognition of ASL gestures, to single frame recognition of gestures from the ASL alphabet. Furthermore, the type of data being processed by the ML algorithm has changed from matrices of raw pixel values, to depth images that have been applied a color-map that represent different ranges of depth within the image. This type of data suits the Resnet-18 expected input more naturally than our original implementation of the matrices of pixel values.
Software	After our first implementation, we decided to go with a different approach to capturing images for the machine learning model. Instead of using the raw depth images, we decided to convert eat depth image to a color-map, and then process these images with the machine learning model. We were able to capture more training examples this way as each image took less time to capture. This gives us a more accurate training set for our machine learning model to use to classify gestures.
UI	The main change that has occurred to the UI is addition of extra instructions upon the startup of the interface, as well as connecting the information display of classification confidence from ML portions, and depth values from SW portions.

2.5.6 *Updated Gantt Chart Timeline*

The Gantt Chart seen below is a final record of the developmental timeline accomplished by our group.

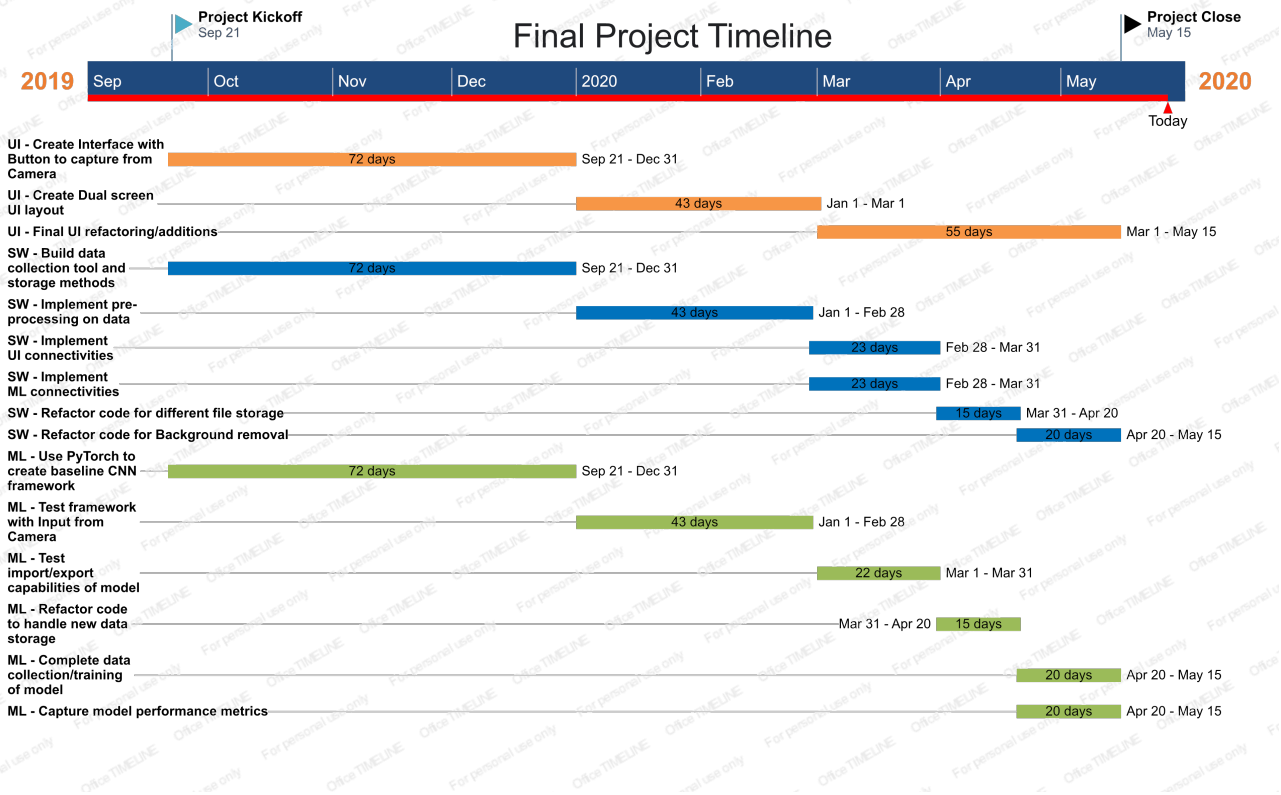


Fig. 2: Final Gantt Chart

3 DESIGN DOCUMENT

3.1 Introduction

3.1.1 Purpose

Our group is expected to train a machine learning (ML) classification model, with the task of having the model accurately identify human gestures from the Intel RealSense camera input feed. We are also expected to build a program with a GUI interface to present the text representation of classified gestures. Furthermore, the motivation behind this project includes the desire to create a cost effective and accessible system that aids people in communicating in more effective ways. Our end goal is to be able to identify American Sign Language gestures using our ML model wrapped in the GUI application. This opens up the possibility of aiding individuals with difficulty using speech as a primary communication medium.

3.1.2 Scope

The system scope of our project has an end goal of having a ML model be able to classify sign-language gestures, and present this classification via a text message to the user through the housing GUI application. Our project will take on a small initial scope, as we seek to first build and train a foundation ML model that can perform basic gesture recognition tasks such as identifying a hand waving. Once this has been established, we will work to fine-tune the model and prepare it to handle our end-goal task of American Sign-Language recognition in real-time from video camera feeds.

3.1.3 Glossary

- UI - User Interface
- SW - Software
- ML - Machine Learning
- GUI - Graphical User Interface
- CNN - Convolutional Neural Network

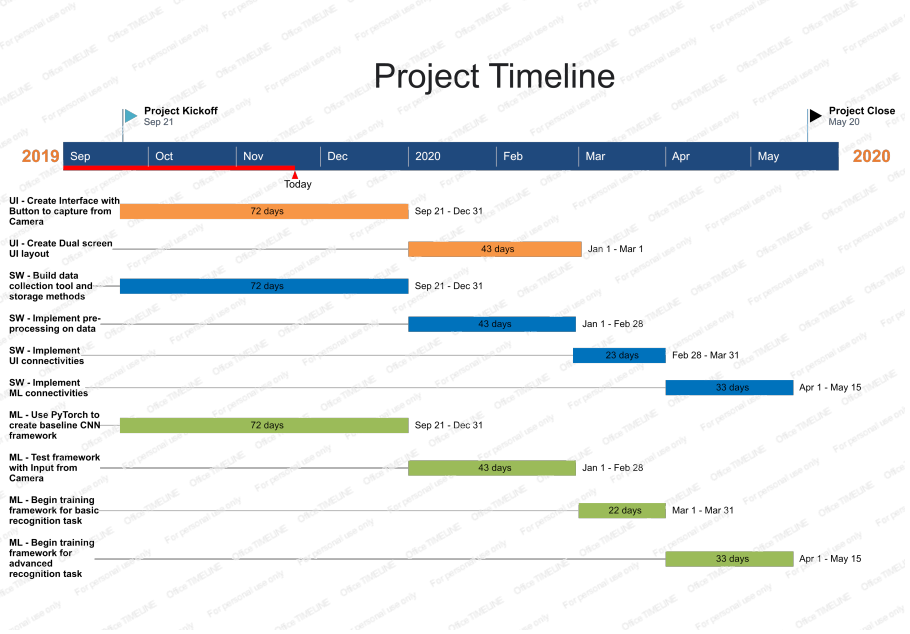
3.1.4 Intended Audience

Our client and direct audience for this document is primarily the Intel stakeholders identified on the cover page, and associated group members. For the users of this project, the intended audience would be those in the deaf community or any in need of a gesture recognition software, bundled with the RealSense Camera. The end goal, as identified by stakeholders, is to create a software tailored to those who may require aid with speech communication. We will keep this specific audience in mind as we begin implementation.

3.1.5 Design Timeline

The figure below displays the timeline for our group's separate key project components, the User Interface (UI), Soft-Ware Interface (SW) and Machine Learning (ML) project portions. As noted from the Gantt Chart timeline below, the key milestones for the UI team include two separate goals of producing an initial basic UI that can perform a simple capture, followed by an advanced UI version that will ultimately be our final interface production model. The SW team has four major goals of creating a Data collection tool and implementing a storage method, performing pre-processing data manipulations, and implementing UI & SW interfacing capabilities. The ML team has four major goals which include creating a baseline CNN architecture, preparing the algorithm to receive camera input, train for basic gesture

recognition, and then training for advanced gesture recognition. These tasks are planned such that each team may work semi-independently from one another, until the interfacing connections by the SW team must be implemented.

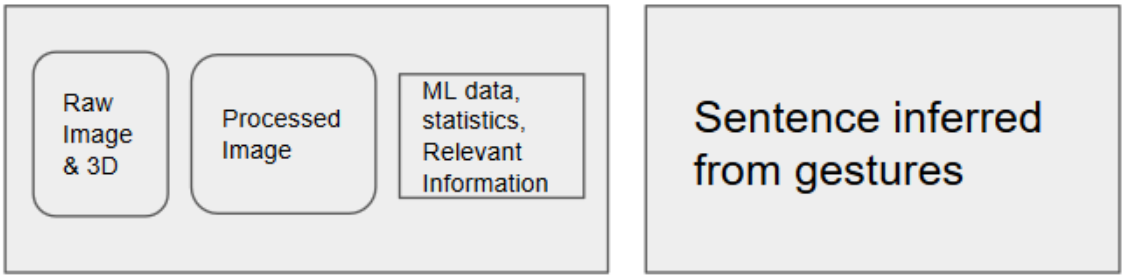


3.2 Design

3.2.1 Viewpoint : User Interface Design

3.2.1.1 Introduction: A major component of this project is creating the UI. The UI within the pipeline serves as bookends to the entire process. Specifically, the user will click a button within the UI to begin the capture process. Next, the camera will collect data, which will processed through SW project portions and then sent to the ML algorithm for a classification result. Finally, that interpretation is sent back to the UI to be presented to the user.

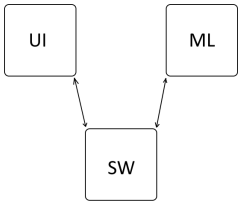
3.2.1.2 Design Elements: The design requested by the client is to provide user with two different screens. The first screen is used to capture/present data from the camera. Within the first screen there will be three components displayed to the user: an RGB and 3D image, the filter-processed image, and relevant ML interpretation information information (e.g. accuracy, statistics, etc.). The second screen is used to display the ML gesture recognition in its string literal form. A representation of this visual using the two screens can be seen directly below:



This layout will display to the user all relevant information that is calculated from our project. Beginning with the raw input they give to the camera, and ending with the final recognition made from our ML algorithm. This UI layout captures all parameters of our project, and will effectively allow a user to input gestures to the camera, and receive the ML classification.

3.2.1.3 Design Concerns: A few elements of the UI that could potentially go wrong are the connections between the camera and the UI and the connection between the ML and the UI. Both of those connections are made through the SW team, which is imperative to the success of our project as a whole. Any incorrect connections between the camera and the UI or the ML and the UI would result in loss of data or data misrepresentation.

3.2.1.4 Relationship: The UI is one of the three parts of the application: the UI, the SW, and the ML. The UI is what the user will see and how the user will interact with our project. Bridging the UI, the ML algorithm, and the camera is the SW project portions. The SW acts as a data exchange point between the UI and ML project portion. Lastly, the ML classifies the user input data (images) and uses the SW as a transfer pipeline to the UI to present the classification. Each individual component relies on the other to relay information through the project pipeline, resulting in a cohesive and interactive project. Below is a basic workflow pipeline of the interaction between the main project components:



3.2.2 Viewpoint : Software Connectivity

3.2.2.1 Introduction: This dependency aims to outline the interconnection between components that the SW project components should bring to our project. Figure 2 below outlines the connection points that the software needs to create.

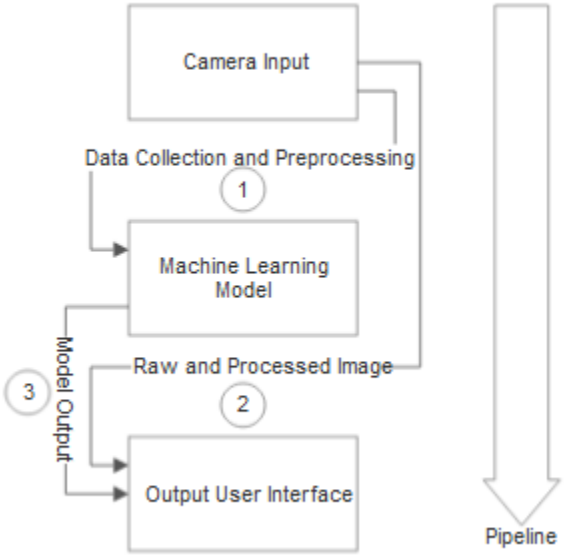


Fig. 3: Block Diagram Displaying Inter-connectivity of Software

The software aims to ease the sharing of data between different components of the project, and connect the input user interface to the machine learning model and output user interface.

3.2.2.2 Data Collection and Preprocessing: During data collection, the user interface will have a button to start capturing information. This data needs to be preprocessed for the machine learning model. The preprocessing design will start by extracting the depth per pixel image from the Intel RealSense Camera pipeline through the Intel RealSense SDK. For training the machine learning model, the data will be added to a current HDF5 dataset after being normalized, and then saved for later training of the machine learning model. For the real time testing and implementation of the machine learning model, this information will be directly sent after being normalized to the machine learning model, where it will be classified. An example of normalization would go as follows:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 1 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} & \frac{3}{4} & 1 \\ \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} \end{bmatrix}$$

This technique will help our machine learning model be able to more easily recognize the same feature repeatedly, as all gestures will be scaled over the same interval $\mathbb{R} \rightarrow [0, 1]$.

3.2.2.3 Raw and Processed Image: The raw and processed images need to be sent to the Output User Interface through a pipeline. This can be done by building a function that can retrieve the current camera information from within the Output UI. We will want two images to display within the user interface, the raw image, and the depth pixel image, that way the user can see what their gesture will be represented as. There will be a separate function for retrieving the raw image and depth image.

3.2.2.4 Model Output: The model output needs to appear in the UI. Once the data is put into the machine learning model, the data will be classified as a string and this result is what needs to appear on the user interface. It is imperative that connecting SW portions are ready to properly receive expected ML algorithm output format, and properly parse this for presentation to user via the UI.

3.2.2.5 Design Concerns: The main concerns for this section of the project is not over-promising on functionality. Not all of the team members are extremely familiar with processing the images and normalization of data. The main problem would be able to get this section completed quickly enough, so that we can properly train the machine learning model with real data that has been normalized.

3.2.2.6 Relationship: This software section is hand-in-hand with the ML project portions. This needs to be done in order to build the database of training data that has been properly normalized. Furthermore, the UI team needs the initial GUI built so that we can have a housing application our group can use to actually begin the data collection process using the RealSense Cameras.

3.2.3 Viewpoint : Continuous Data Input

3.2.3.1 Introduction: This information section will show how the software will deal with persistent data received from the camera. Specifically how the process will be able to take the camera input and classify the gesture real-time using the information that had been previously collected to train it. The gestures that it will be processing are ones that we are assuming the model has been previously trained for.

3.2.3.2 Data Storage: Before this project can classify real-time, it needs to be trained to recognize them. After that, it will need a way to store the data it has collected to recognize performed gestures real-time. The original plan was to have a database connected to this project but at this point, the goal is just having accessible data without the hassle of retrieving it from a database. We will utilize an HDF5 file format that we can simply read for the model and will contain all of the input data. This will allow us to not waste time getting a database set up with the proper amount of data needed for ML model training.

3.2.3.3 Design Concerns: There are a few concerns for the data storage, namely that our group needs to ensure that the location where we store the HDF5 files is accessible for the machine learning model. Furthermore, there needs to be substantial space available on the drive for the files to be added and read from. Any issues with storage must be fleshed out as it can block our progress with the project overall, and impede our overall timeline.

3.2.3.4 Relationship: This section again is directly related to the ML team. The data we record must persist since the model will not be able to have accurate training without it. The data should be recorded via the first GUI interaction with the camera. It is imperative that the GUI application is ready to begin receiving camera input in order to begin implementation for this SW viewpoint.

3.2.4 Viewpoint: Machine Learning Dependencies

3.2.4.1 Introduction: The dependencies required by our Machine Learning model include the utilization of the Intel RealSense camera for model input, specified by our client, and the utilization of the PyTorch Deep Learning Framework to build the ML model. The camera input will ultimately feed the model, which will be created using PyTorch.

3.2.4.2 Intel RealSense Camera Input: The camera we will be using for our project is the Intel RealSense Depth Camera SR305. This indoor camera has an incredible quality of depth capturing capabilities, specifically when used under ideal conditions. Our group will utilize the existing Software Development Kit offered by Intel to utilize the camera's components to capture data. Specifically, in the context of our project's pipeline, the ML algorithm will receive camera input delivered from the UI portion of our project and processed by the SW pre-processing manipulations. Once the camera data reaches the ML algorithm, it will be ready for direct input into the model in order to yield a classification output.

3.2.4.3 PyTorch Deep Learning Framework: Our group has chosen to implement a Deep Learning Convolutional Neural Network in order to accomplish the task of gesture recognition. Our group will be using the PyTorch Machine Learning Library Framework in order to begin building our Machine Learning Model. The PyTorch framework is geared to be more user-friendly than other frameworks in existence, and because the rest of our project is being built in Python it will allow for easier overall integration to utilize the PyTorch library built on top of the Python programming language. With this framework, our group will aim to build a Convolutional Neural Network algorithm to tackle the task of gesture recognition.

3.2.4.4 Design Concerns: The main concerns surrounding our group for both of these dependencies in the ML domain of our project involve the fact that this our group's first exposure to both of these tools. We will have to perform thorough research and development into both of these tools and their interactions, in order to be able to fully utilize these components for the ML project portions.

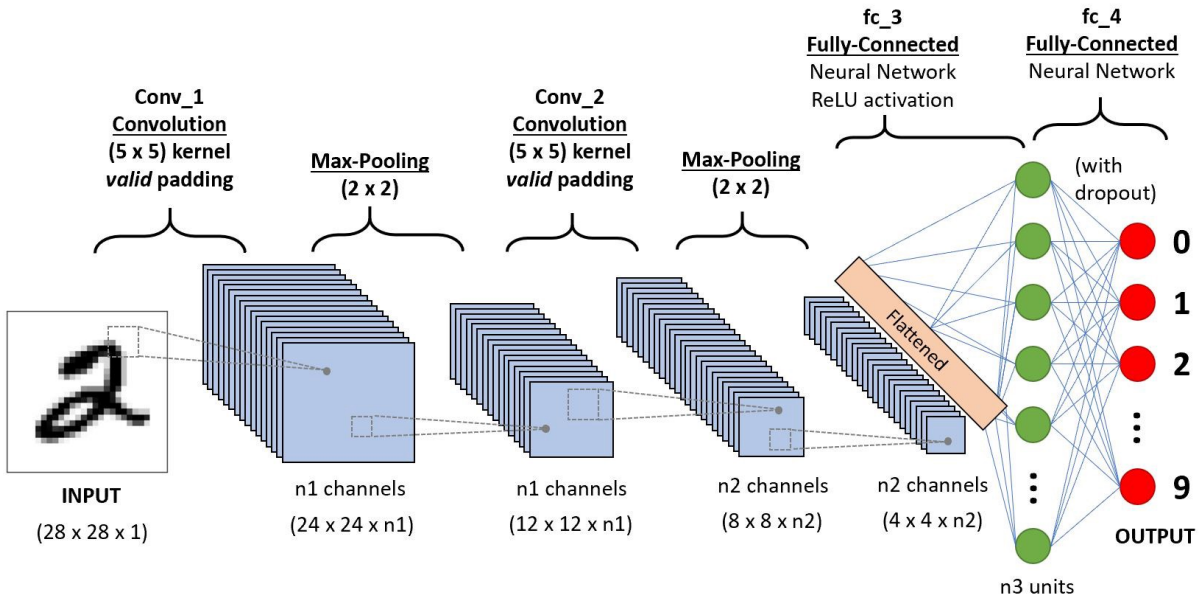
3.2.4.5 Relationship: The relationship involvement with the RealSense Camera dependency comes from both the UI viewpoint and the Software Connectivity viewpoint. User will need to interact with the UI in order to give input

to the ML model, and the raw data will need to be manipulated prior to being fed to the model.

3.2.5 Viewpoint: ML Algorithm

3.2.5.1 Introduction: The chosen algorithm for our Machine Learning model is a Convolutional Neural Network (CNN). This algorithm will be the driving force behind our project's capability of recognizing human hand gestures. The algorithm will be built by using the PyTorch Deep Learning Framework, and will make the ultimate gesture recognition predicition for our project.

3.2.5.2 Convolutional Neural Network: The nature of our project ultimately boils down to a computer vision problem, where our group is tasked with creating a computer program capable of recognizing human hand gestures coming from the Intel RealSense Camera. While there are many ways our group may choose to approach the implementation for this task, our group went with likely the most promising avenue at our disposal, which is a Machine Learning algorithm. Within the domain of Machine Learning, there exists multiple different types of algorithms each with their own strengths and weaknesses. As of recent, one particular ML algorithm known as Convolutional Neural Networks has shown astounding capabilities in it's ability to perform ML tasks[7]. For this reason, this is the ML algorithm that our group has chosen to implement for our gesture recognition task.



A pictorial representation of the Convolutional Neural Network pipeline can be seen above, with a single image input example [7]. A high level look at this algorithm reveals a sophisticated computational graph that is comprised of several layers working together to accomplish a machine learning task. A deeper dive into the algorithm reveals three major layer types that compose the CNN architecture; the Convolutional Layer, the Pooling Layer, and the Fully Connected Layer. These three layers are sequentially connected and result in sophisticated computation graphs that performs very well in machine learning tasks. The Convolutional layer performs image dimension reduction, and extract the high level features of the input image such as edges and colors. The Pooling layer takes the output from the convolutional layer and performs spatial reduction on the newly represented dataset. In this process dominant features are extracted and noisy features are ignored which results in higher accuracy models. Lastly, the Fully Connected layer takes the pooling layer output and performs calculations to make the algorithm's ultimate prediction[7].

3.2.5.3 Design Concerns: One of the main design concerns involved with this chosen algorithm is simply the complexity of such an implementation. The area of machine learning is an active area of research, and specifically CNN research due to its recent success. This being said, our group will likely have a steep learning curve with this chosen implementation. Furthermore, this algorithm requires a large training set to properly learn its given task. Our group will have to build this training database from scratch, and this leads to concerns of time constraints. Our group will not only have to properly build this algorithm, but also create the training sets needed for a thorough implementation.

3.2.5.4 Relationship: This viewpoint directly relations with the Machine Learning dependency viewpoint, as the PyTorch framework will be used to build this algorithm. Furthermore, the Machine Learning Connectivity Viewpoint is a direct relationship to this algorithm, as each connection that is highlighted through that viewpoint will have direct interaction with the algorithm highlighted in this viewpoint. These two interactions are vital for the smooth implementation of this design viewpoint.

3.2.6 Viewpoint: Machine Learning Connectivity

3.2.6.1 Introduction: The ML portion of our project must interact with the two other major components of our project, being the UI and SW portions. Our project will initially require a user to interact with the UI to provide input data to SW project portions, which will perform pre-processing manipulations to provide direct input the ML algorithm. The ML algorithm will then provide it's gesture classification output back to the SW portion which will parse the information, and ultimately present the classification in the UI to the user.

3.2.6.2 UI Interaction: The UI interactions with the ML project portions are indirect, with the SW portion mediating the data exchange between the two. Although the relationship is indirect, it is dire that the information exchange between the two is accurate, as they are the raw truths as to what the direct camera inputs and direct classification outputs are. It is important to establish a firm pipeline connection between these entities in order to ensure the creation of best possible application we can feasibly produce. The main way this can be accomplished is through a solid mediator, being the SW portion of our project.

3.2.6.3 SW Interaction: The SW interactions with the ML project components are direct, with the output from the SW portion directly feeding the ML algorithm, and the ML algorithm's output directly being fed back to SW portions for parsing and presentation. The interactions between these two components are essential to our project's success, and we must ensure that the correct pre-processing occurs to the data, in the context of our selected ML algorithm. Furthermore, we must ensure that the SW is ready to receive expected ML output formats so that it may correctly parse this information, perform any required performance metrics, and securely present this information to the user via the UI. These connection components must be thoroughly tested to ensure the quality of our project's information retrieval and presentation.

3.2.6.4 Design Concerns: The design concerns for this viewpoint mainly arise for the direct interactions taken between the ML and SW project portions. Because the SW project portion will act as a data exchange between the UI and ML algorithm, it is imperative that a proper data hand-off occurs between these components. The pre-processing manipulations (explained in the Software Connectivity Viewpoint) must be properly formed for input to the ML algorithm, and these same SW portions must be ready to handle the output format of the ML algorithm. These connections must be thoroughly tested to ensure project integrity.

3.2.6.5 Relationship: One main relationship for this viewpoint is the ML algorithm viewpoint. The main reason for this relationship is again, the fact that a proper data exchange must occur between SW and ML project portions

in order to ensure the accuracy of information displayed to the user. Furthermore, this relationship cascades to the UI viewpoint, as the user will ultimately input directly to ML portions, and see the output of ML calculations.

3.2.7 Viewpoint: Machine Learning Required Resources

3.2.7.1 Introduction: The process by which our group will train and test the validation of our ML algorithm's gesture recognition capabilities is through a collection of training sets created by our group, using the Intel RealSense cameras. Our group will utilize on-campus computing resources to tackle the large computational cost required to thoroughly process these training sets through our algorithm.

3.2.7.2 Machine Learning Training Data: The training data we will use to train our ML algorithm for the gesture recognition task will be the most vital data aspect of our project. There is a high importance in obtaining plentiful and quality forms of training data, portraying a wide range of possible motions for a single given gesture. Our group has established base guidelines when capturing data, such as requiring that gestures be captured under the ideal condition of a one meter distance from the camera. Quantitative measures such a total number of videos per gesture, and video length per gesture will be determined immediately upon foundational building of our ML algorithm.

3.2.7.3 Computational Resources: The computational cost required by our project to sufficiently train the ML model for gesture recognition is going to be more than just a single group member's machine can handle. The complexity of the algorithms used to build ML models will require high computational power, and therefore require our group to utilize methods for multiplying computational powers. Our group's chosen option for this need is to utilize the in-house computing cluster hosted on OSU's server. This parallelized computing resource will help mitigate wait times caused by complex computations, and be a highly beneficial component to the ML portion of our project. Although this will mainly affect the training/testing phases, it will help ensure a higher quality ML algorithm as a result.

3.2.7.4 Design Concerns: The design concerns associated with this viewpoint mainly revolve around the required machine learning data for this project. The concern is the amount of data that is required to properly train the algorithm for the recognition task at hand, and the fact that our group must build this training set from scratch. The main concern ultimately boils down to a worry of lack of time for the sheer amount of work involved in this project. Our group will aim to follow the timeline described above, and hope that this is enough to time implement all portions of our project.

3.2.7.5 Relationship: The relationship this viewpoint holds to other viewpoints is mainly concerned with the Software Connectivity viewpoint, and specifically the data collection domain. The form of data collection outlined in the Software Connectivity viewpoint will dictate the data form that collected training sets will take. This will also dictate the form of input taken by the ML algorithm, and will help guide the building of this algorithm.

3.3 Design Rationale

3.3.1 Pipeline

The project structure is a sequential pipeline. The goal of this structure is to be able to divide the project into parts that can be developed, tested, and function on their own with simulated inputs and outputs. Eventually each part of the pipeline will be connected to create the complete product. The following figure explains the connections within the pipeline.

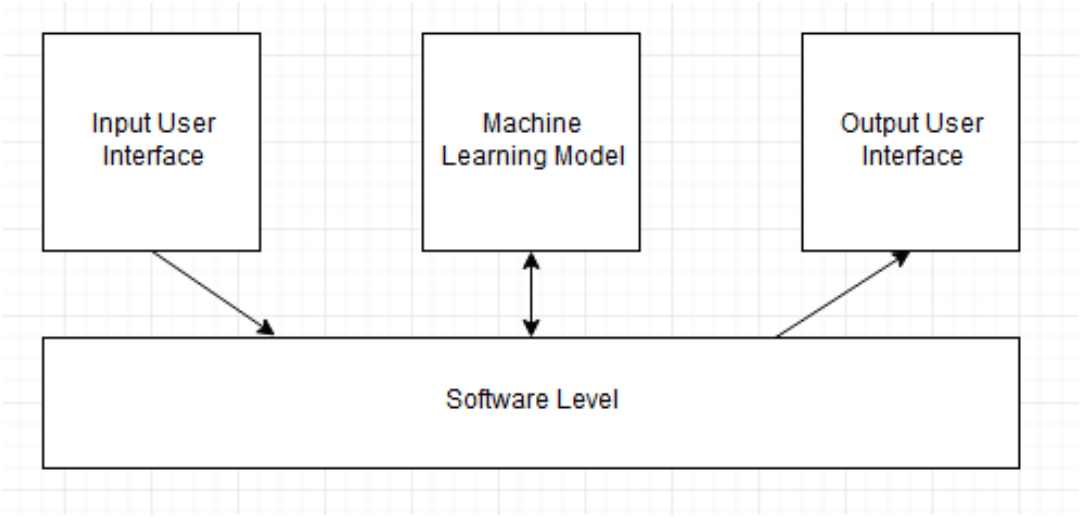


Fig. 4: Pipeline and Connectivity Diagram

The rationale behind this structure is the Dependency Inversion Principle. We would like all of the parts to be simulated and function on their own, but once all of the modules are built, they need to depend on each other. This design helps us be able to build our product in parallel so that there is more time for bug fixing, testing, and training of our machine learning model. The project is data driven, and thousands of image data needs to be captured and stored into our data storage method. This design allows us to develop certain parts of the project while gathering our data set, which will be time consuming and done manually. This pipeline model outlines the interactions for our project's separate components.

3.4 Changes

3.4.1 Change Table

Below is a table summarizing the changes that occurred across the separate development teams, over the course of Fall term and early Winter term.

Change Table	
Project Component	Changes made
Machine Learning	The main changes that have occurred on the Machine Learning portions of the project include the fact that our project scope has changed from video recognition of ASL gestures, to single frame recognition of gestures from the ASL alphabet. This significantly changes the type of work needed to be done on the ML project components, and simplifies the recognition task at hand to better suit our group's knowledge within this domain.
Software	We have added more preprocessing techniques for the machine learning model to better classify gestures. Normalization was too simple of an approach, especially after weak results of verifying our machine learning model. We have inverted the image data, as well as re centered the data before normalization, that way we could focus on maximizing the values that closely relate to the gesture, instead of the image as a whole. This solved the issue of the same gestures at different depths being treated differently by the model.

4 TECH REVIEW

4.1 Tech Review: Shane Clancy

4.1.1 Introduction

The way in which our group handles data is extremely important for the optimization and functionality of our project. We would like the components of the project to be able to work with each other fluidly and want to result to the least amount of file input and output that we can for speed purposes. There are many advantages and disadvantages to the tools that are going to be discussed for these three main topics:

- Data Processing and Collection
- Machine Learning Input/Output
- Data Manipulation and Preprocessing

4.1.2 Data Processing and Collection

4.1.2.1 Overview: Data processing and collection is fairly straightforward because of the software that we have been provided by the Intel RealSense SDK [8]. Many people have contributed and helped develop the software for the Intel RealSense Camera [9] that we are using within our project. There are detailed examples for most of the work that can be done with the information provided by the camera, as well as everything that we will need in order to collect the correct information from the Camera. We need to collect an abundant amount of information to be able to create an accurate machine learning model, since sign language has a broad spectrum of different gestures. Although we are aware it will be almost impossible to represent all of sign language in such a little amount of time, we are hoping to achieve a very accurate representation of the language that will be usable by those with communication challenges.

4.1.2.2 Python: The first option that we have for data processing and collection is Python, which is one of the easiest tools to setup for a new environment. Python behaves similar on all operating systems and is easy to install on the operating systems that we are using to build our code base. Python is also efficient because it is fast to write, and will reduce the amount of time it will take to program a task, letting us focus on the details of the project instead of debugging complex code. The SDK that we have been provided has a python library that exists as a wrapper that contains plenty of examples to work from. Python is also fairly easy to setup for databases as well as reading and writing to storage devices, which we will have to do frequently for the data we are collecting. Since our machine learning models will most likely be written in Python, the integration between the data collection and processing will be very simple, and can even be connected by another process without much work. However, Python is generally slow compared to some of the other languages that we could use, and if we are trying to do tasks in real time, it might become difficult.

4.1.2.3 C++: C++ is one of the fastest languages at run-time, which makes them a great candidate for collecting our data since we can do this as fast as possible with this language. All of our group members should have in depth knowledge of C++ due to studying it a fair amount at Oregon State. There is a detailed and well documented wrapper for OpenCV, which is a popular C++ based tool and will be easy to work with [11]. Although C++ is a very fast tool, it might take a fairly long amount of time write the code for basic operations. C++ is also very volatile and we might run into memory management issues which would lead to lots of debugging. C++ might be very complicated to setup a database with, as none of our group members have experience with working with a database in C++. Even though we all know this language, we would have to do much more learning to be able to properly incorporate this into our project.

4.1.2.4 C#: C# is an easier high level language to work in, in which many members of our group have experience with already. The SDK wrappers for C# are some of the most detailed [12], and provide lots of references for different data collection methods that we will use. Our group members who are planning the GUI, also like the Unity integration that comes with the SDK, and writing some of the data collection in C# would greatly benefit the GUI development. C# however, is not too friendly to develop on different operating systems, but would not be too hard to get set up for everyone to use. We would most likely be able to interact with our machine learning model directly using this approach because of how we could run other processes within C# using the ProcessStartInfo Class [12]. Database integration might be complex for C#, since it is more meant for desktop applications that present the GUI, and not interacting as a backend server. C# would be a good choice for integration with the GUI but some of the group members might have to do some learning to get their parts to coordinate with this service.

4.1.2.5 Discussion and Conclusion: We believe that our data collection would best be done in Python, since the wrappers provided by the SDK require the least setup, and can easily be run on any machine with any operating system. The documentation for the example programs makes the data collection simple and straightforward. Our group members are working with different operating systems, and some languages are very friendly to install and run code. We would like each member to experiment with data collection so we can have variance in our data. Since we have picked Python for a large portion of our project, we would like to continue to use Python as much as possible throughout the rest of the components. Machine learning is almost solely done in Python, with libraries existing in the language that provide many different functionalities, such as PyTorch, Tensorflow, and Keras, to name a few.

4.1.3 Machine Learning Input/Output

4.1.3.1 Overview: Machine learning input and output will be done differently depending on the language that we will be using for the machine learning aspect. However we as a group have decided we are going to be using Python for the machine learning aspect of the project. Some of our group members have machine learning experience in Python already, and think that it is the best option to get the job done. We will need to be able to get data from our database or file format, train our machine learning model on that data, and continuously save our model in a format that we can backup and store. Due to the fact that we are set on using Python for the machine learning it will most likely be best fitted to use Python for input output. However, we can format the data in differently for input and output due to where the information will be present. The following sections will talk about different data formats for input and output for our model.

4.1.3.2 Comma Separated Value: This format is fairly popular for existing data sets, and is fairly straightforward on how to use. Each value is separated by a comma, even if values are easily compressible, the format stays consistent and is easy to read into a machine learning model. Comma Separated Value (CSV) is a great format if we were dealing with only a small amount of features, but since we are likely going to have at least one feature per pixel, depending on the type of image we store, which creates a very large dataset very quickly. All of us are already familiar with this format, and it would be simple to incorporate this format with all of our components.

4.1.3.3 Hierarchical Data Format 5: Hierarchical Data Format 5 (HDF5), is a popular compressed data format for machine learning. This data format makes storing more information easy, with quick storage and read times. This type of format is really useful for storing images and videos [14], as it is good at storing lists of items. HDF5 is becoming increasingly popular, and is supported by many different languages, including Python, which would make it easily usable within our project. Since our project is computer vision based, we can choose to store each image in a format

such as HDF5 for easy representation and be able to create more entries into our database than we would with an uncompressed format. The only downside of this format is that we have not interacted with the format before, and would need to understand the format when writing our other components.

4.1.3.4 Gzip: Gzip is one of the fastest and most efficient compression types for Unix systems and would be a great fit for storing large quantities of data which we will be doing in our project. Gzip is free to use under GNU and would be easy to incorporate into our projects if we are running our machine learning model on any type of Unix based server. The algorithm that Gzip uses will work well with our data, since sometimes we will encounter lots of depth that is zeroed out, and not useful. We would not like to store all of this zeroed out data individually and would like to compress it for easier storage. Gzip algorithm works as such "If the program encounters these recurring sequences, it replaces them with a link to the string that first appears" [15]. One of the disadvantages to using Gzip is that our group does not have much experience with this tool, however we think that it will not take long to understand its benefits.

4.1.3.5 Discussion and Conclusion: Overall we believe that HDF5 and Gzip are both great candidates, but both depend on type of image we are choosing to store into our database. Since we could store the r,g,b values for each pixel, or we could choose to store the depth at each pixel. If we were to use depth, we might want to explore Gzip on our data to see how well it can compress a long sequence of zeroed data from the camera cutting out pixel depth that is too far. However, HDF5 has been proven to work well with image processing and deep learning, so if we don't want to experiment, this would be to go to file format.

4.1.4 Data Manipulation and Preprocessing

4.1.4.1 Overview: There are many popular preprocessing methods for machine learning. The methods that we are going to find the most useful on our set of data are normalization, scalar normalization, and principal component analysis. These methods all help make the data less variant and more consistent on the model.

4.1.4.2 Normalization: If we chose to use a depth per pixel image, normalization can be a great technique to utilize. The closer a pixel is to the camera, the larger the depth value is. We can find the minimum and maximum nonzero values of depth and scales these values onto a range of 0 to 1. This will make gestures that have been captured at different depths, to be more similar in raw data to the machine learning model. The way that normalization works is it takes the largest value in the data set, and divides every other entry by that value, creating a scale of 0 to 1 for each of the data entries [16]. This process seems almost necessary for our model, and we should definitely use this technique in order to train on our data set as it will drastically improve gesture recognition and variation reduction between data of the same gesture.

4.1.4.3 Robust Scaler Normalization: In some cases outliers can play a large factor within normalization, and this is the purpose of Robust Scaler Normalization. This technique tries to get rid of outliers before performing a normalization technique, which can be important if there are some depth pixels that extrude more than the rest of a gesture. This is done because "outliers can often influence the sample mean / variance in a negative way. In such cases, the median and the interquartile range often give better results" [17]. However, robust scalar normalization could accidentally prune features from a gesture that might define it differently from another closely related gesture, which would become problematic for the model. We are going to explore the performance of the model with this normalization compared to base normalization on different similar gesture and test our findings.

4.1.4.4 Principal Component Analysis: Principal Component Analysis (PCA) breaks down vectors into a smaller space, that way the most important features can be extracted for training [**?**]. While this seems like a good approach for

such a large data set we will be working on, we would be worried that the position of certain gestures within the image will make the model worse than with no preprocessing at all. However, if we were to determine if user is in a position to where their gesture is located in a single spot, we might be able to build a separate model that trains off of the smaller vector space. This could be an interesting approach, but also negatively impact the training of the model.

4.1.4.5 Discussion and Conclusion: Most of the preprocessing methods mention would greatly improve some of the feature recognition of the machine learning model and should be taken into consideration. Normalization seems to be a technique that we must use when preprocessing the data for our model. However the type the preprocessing that needs to be done needs to be experimented with, as certain gestures could lead to problematic outcomes due to types of normalization. If our model proves to be less accurate on gestures that don't use the whole dimension space, we might be able to do some calculations and make a separate model to perform PCA on.

4.1.5 Appendix

- SDK - Software Development Kit
- GUI - Graphical User Interface
- CSV - Comma Separated Value
- HDF5 - Hierarchical Data Format 5
- GNU - GNU Not Unix
- Gzip - GNU zip
- PCA - Principal Component Analysis

4.2 Tech Review: Ulises Zaragoza

4.2.1 Introduction

Our project for the CS 461 Senior Software Engineering Capstone class is the '*Gesture Recognition using new Intel RealSense coded light camera*' project. My team consists of the following individuals: Shane Clancy, Nicholas Davies, Jonathan Hull, Shihao Song, Zhidong Zhang and myself. A high level summary of our project looks at utilizing the RealSense cameras offered by Intel to automatically recognize human gestures and produce translations of these gestures into their readable text representations. The steps needed to accomplish this task will include building a database of gesture videos for which to train a Machine Learning (ML) algorithm to correctly identify different gestures seen from camera video feeds. The classifications made will be presented to users via a Graphical User Interface (GUI). This document aims to detail the specific technical elements of our project, that have been broken up by our group and divided out amongst team members.

Our group has broken up our project's technical components into 18 fields, to focus our group's research efforts and be able to confidently begin project implementation. The following fields will be researched by our group: data normalization techniques, potential data storage methods, potential computational resources, GUI creation as internal tool or on web page, programming languages used to build GUI applications, existing Intel Software Development Kit (SDK) GUI capabilities, Intel RealSense Cameras, existing Intel SDK interfacing capabilities, data collection for ML training, data parsing for ML training, ML input and output, pre-processing data manipulation techniques, gesture translation into languages other than English, ML Dimensionality Reduction, general research into American Sign Language, Natural Language Processing, Deep Learning/Neural Networks, and reinforcement learning. The research efforts conducted on behalf of myself will include a dive into the following fields: Potential Data Storage Methods, Potential Computational

Resources, and Data Normalization Techniques. This paper will look at potential implementation options for each of the fields, with specific context to our project, with the goal of identifying a candidate implementation based on research conducted.

4.2.2 Data Storage Methods

The main goal of our project is to build a ML model that is able to classify human gestures coming from the video feed of an Intel RealSense camera. The process by which our group will build and train a ML model for the task of gesture classification will require a large amount of data. This data will take the form of an array/matrix of pixel/color information for a given captured image from the RealSense cameras. Because our project's aim is to capture gesture recognition, this will require a larger memory allocation than normal as each gesture training example will be a series of frames captured from the RealSense camera, representing a video. Our group will need multiple different training examples for each of the different gesture we are aiming to eventually classify. This means that our data storage method must support scalability, grouping mechanisms, and portability as we will be looking to upload/access these examples at any given moment, from any location.

With the needs of our group's storage method in mind, one potential option that our group may consider for data storage would be to simply store our array/matrix of meta-data into .CSV files and upload these to a shared folder on the OSU FLIP servers. This option is appealing to our group because of the ease of access to all of our group members, and the security measures offered by the school to ensure the integrity of FLIP servers. Furthermore this option is cost-effective for our group, as we do not have to pay any additional fees to utilize the FLIP servers for storage. Unfortunately, this option does have its downsides, with the main fault being the lack of scalability for this option. Each student has a fixed amount of storage on the FLIP servers, and there is no telling if our group will require an amount of storage beyond what is offered to us via the FLIP servers. Additionally, storage on the FLIP servers would not allow our clients to have access to this data, due to the security measures set in place by the University. Although this option has it's clear downsides, it would provide a quick and easy option for our group during infant stages of project implementation.

Another potential data storage method our group could consider, would be the utilization of a cloud storage service. To begin with, this option mediates the main issue associated with lack of scalability through storage on OSU FLIP servers. Nearly all cloud storage services are meant to tackle the issue of scalability in memory spaces, and this option would alleviate any worries of lack of storage for our training examples. Furthermore, utilization of a cloud storage service would allow our clients to also have access to our project's data, as security measures that do now allow foreigners to enter University servers would no longer be an issue under this implementation strategy. Unfortunately, the down-side with this storage method ultimately boils down to the cost associated with utilizing cloud storage services. While some storage options, such as Google Drive, will allow free storage up until a certain capacity, our group will ultimately need to pay a fine once our example data grows past a certain point. Despite the costs associated with this option, I believe that the security of scalability with this option justifies the cost needed to implement a cloud storage service.

Yet another potential data storage method our group could consider would be utilization of a database through a common service such as MySQL or MariaDB. This option again alleviates the issue of scalability encountered through storage on OSU FLIP servers, and because we have free access to MySQL services as OSU students, this option would also be cost-effective for our group. While this seems like the perfect combination for our group's data storage method, there are potential issues that could occur with this option. Namely, this option may be over-complicating our storage

needs. Our project's training data will take the form of an array/matrix of pixel/color information for a given captured frame, and this option the best method for storing such a primitive type of data. Implementation for our group, would look like a single massive table, representing the meta-data information captured from the frame. This may not be the best approach for our group, considering the type of data we are working with.

Considering the information provided above, I believe that the candidate data storage implementation for our group's project would be the utilization of cloud storage service. Despite the potential costs associated with this method, the benefits offered through scalability, ease of use, ease of access, and portability justify having to pay for a cloud storage service. This will allow our group to confidently begin the process of data collection, and will alleviate any worries of lack of memory allocation. This option will be the best for our group, given our project's context and the potential for the need of a large storage capacity.

4.2.3 Computational Resources

The computational cost required by our project to sufficiently train our ML model with the task of gesture recognition is going to be more than just a single group member's machine can handle. The complexity of the algorithms used to build ML models will require high computational power to thoroughly execute and train the model. Therefore, our group will need to consider different methods for multiplying computational powers in order to accomplish our project in a timely manner. We do not want to be held back by the wait time required to compute the ML algorithms, and are looking to consider different techniques for accomplishing different forms of computational resources. Again, our group must consider this computational resource to be cost-effective, and allow for the ease of use when partitioning computing tasks with the goal of optimizing computation times.

One primitive, but effective option that our group can consider to utilize when partitioning computational tasks is to simply divide these tasks physically amongst our group members. Under this implementation, each group member would take on the task of running our algorithm through a set number of examples. This way, the total number of training examples needed to be analyzed by our model will be accomplished by dividing this total number amongst group members. This is a simple approach to multiplying computational powers, but if coordinated well, this could prove to be an effective strategy for our group to minimize waiting times due to long computations. The downside to this strategy, is that each group member is still subject to potentially long computational times within their specified training domain. This strategy still directly exposes our group to needing to wait for computation times, and could potentially mitigate process in other areas because of the need to wait after dividing tasks amongst members.

Another cost-effective option that our group can consider is the use of OSU's Computing Cluster. This group of computer's hosted through OSU's network, are enabled to parallelize a task with the goal of computing this task in very high speeds. Unfortunately, only a few individuals have access to the cluster, so this would be the initial obstacle that our group would need to deal with in order to utilize this option. Once our group has been granted access to run scripts on the cluster, we may then freely use this computing power to efficiently train our ML model in a timely manner. Another great, time-saving component, of utilizing the cluster is that multiple jobs may be given to the cluster, and they will simply run until they are completed. This is a highly beneficial component to our group, as not only will this minimize computation time through a parallelized system, but we may also just give the cluster a series of jobs to complete, and may spend time focusing on other tasks while we wait for the cluster to iterate through all jobs and complete them. The main obstacle with this implementation will be gaining proper access and training to utilize OSU's cluster system, however once this has been established our group will see great benefits through implementation.

Yet another computation resource our group may consider for our project's task of training the ML model for gesture recognition, would be the utilization of a cloud computing resource. The utilization of this technique would look similar to the utilization of OSU's cluster, where our group would simply use this service to submit multiple jobs to the chosen service, and the service would complete this jobs at a pace higher than any of our group member's machine may accomplish the computational task. We see the same benefits from this option, as seen when utilizing OSU's computing cluster, however the main downfall from this method arises from the costs associated with using these cloud computing resources. Although the benefits associated with using these services may outweigh the potential cost, the fact that our group has option to utilize the same services through a free service on campus, make this option slightly less appealing in comparison.

Considering the information provided above, the candidate implementation for our group's potential computation resource will be the OSU computing cluster. This option is unrivaled to simply splitting computation tasks amongst team members, and because this option is free to us as OSU students, this beats having to pay for a cloud computing service. The need for relatively quick computation times for our project is high, as our group must accomplish a lot for this project, and we cannot afford to be backtracked due to waiting on computational times. Having access to the cluster to submit jobs for training ML model is imperative to our group's success, and we will employ this option as our computation resource, once granted access.

4.2.4 Data Normalization Techniques

One common approach taken when working with the training data of ML algorithms, is to normalize the data that spreads across the different features that are represented. The idea of normalization is explained in Shay Geller's article, Normalization vs Standardization — Quantitative analysis, where readers learn that data is 'normalized' whenever the data across different features is represented on different scales. When this is the case, all of the corresponding data belonging to different features is normalized so that these features data is represented across the same scale. The goal is to allow the model to view all features in the same scale, with the hope that they are all treated equally and then properly weighted in the corresponding algorithm [27].

One common and primitive approach to the normalization of differently scaled data is known as Decimal Scaling. This approach is extremely simple and to accomplish this scaling method, the essence is that decimal point is shifted for all these values, so that the new set of values ranges from [0-1]. The shift that occurs is a division of all values, by the power of ten that is greater than all values in the vector. An example of this with a set of values containing: [1,50, 200], would divide all values by 1000, and the new corresponding set would be: [.001, .05, .2]. The result is a scaled representation of the original corresponding set of values.

Another common approach taken to normalize training data is known as Min-Max Normalization. In this approach, described in CodeAcademy's article on Normalization, the goal is again scale the feature data to allow for equal scaling. In this approach, the minimum value of a given feature receives the value of 0, and the maximum value receives a value of 1. The values that lie in between are scaled accordingly, using the new min/max values. The equation to accomplish this for any given value is: (value-min)/(max-min). The new corresponding values are now scaled accordingly between the range of [0-1], and this allows the model to view features under the same scale [29].

Yet another approach to data normalization is known as feature clipping. This is a different take on normalization, that is described in detail on Google's Machine Learning Courses page on Normalization. This technique requires a deeper dive into the meta-data, as features are examined for obvious outliers, and then a data capacity is imposed on

the data set. With this capacity in set, any piece of data that falls beyond the clipping bound will be ignored by the model. This approach aims to reduce noise in the model, and allow the model to focus on only the informative features and their corresponding values, allowing for proper model learning [30].

The field of ML is still an active area of research, and the effects of different normalization techniques is something that researchers are still looking to learn about. This being said, our group may attempt different normalization techniques, with the hopes of identifying which method works best with our project's context. However, we are still looking to track on the steps that researchers have already taken. As noted from Ashikin Ali and Norhalina Senan's article, The Effect of Normalization in Violence Video Classification Performance, these researchers found that the normalization technique of Min-Max with a range of [0-1] yielded the best results in their model [28]. Our group will likely attempt this normalization technique at first and then explore different techniques later on.

4.2.5 Conclusion

Our group has broken up our project's technical components into 18 different aspects, to be divided amongst the group members for research and identifying candidate implementation for each of the fields. This paper has researched the topics of potential data storage methods, potential computational resources, and potential normalization techniques for our project's training data. For potential data storage methods, research conducted has yielded that a cloud storage service will be our group's best option for data storage due to scalability and ease of access. For potential computational resources, research conducted has found that utilization of OSU's computational cluster will be our group's best option for utilizing external computing power for our project. Lastly, for normalization techniques, our group will likely have to explore which technique works best with our given project's context, however research conducted elsewhere has found that Min-Max Normalization with a range of [0-1] has showed promising results. From the research conducted above, these are the decision our group will choose to implement for the given topics.

4.3 Tech Review: Jonathan Hull

4.3.1 Introduction

As a team of six, we are working together closely with our client at Intel to create a machine learning algorithm using a RealSense camera that recognizes gestures and converts them to text. The end goal is to ultimately create a way for the RealSense Camera to recognize American Sign Language through the ML algorithms we create as a team.

4.3.1.1 Team Goal/Project Process: Firstly, our team will start researching all aspects of the assignment by delegating different elements to each person. We will work on an individual and group level as well as meeting with our client and teacher's assistant on a weekly basis. This will help our team stay focused on the end goal as well as creating a collaborative work environment to produce high quality results. Next, our team will build the ML algorithm. Up to this point, our team expects to be using Python as the language, however this might change. Our group also plans to look at previously established ML algorithms with the potential of adopting them for our project.

After we create or obtain the ML algorithm, our next mission will be to train the algorithm to use the Intel RealSense Depth Camera SR305 to recognize simple gestures. To begin, we will most likely use gestures such as "rock, paper, scissors" and then evolve to more complicated gestures such as American Sign Language. In the meantime, while we are creating the gesture recognition training data, we will work on creating our GUI. With this we hope to accomplish being able to see what the camera interprets the gesture to be. Our end project goal is to showcase our created ML algorithm to our client.

Since this is a major project with several different layers and opportunities for research, we broke down all of the elements of the project and delegated them out for us to research on an individual level. My three main responsibilities delegated to me as the group are researching the following:

-Intel RealSense Camera

-researching the SDK

-researching different methods of data collections.

4.3.2 Camera

The camera our client provided us with is the Intel RealSense Depth Camera SR305. However, there are a few different cameras on the market that also allow for depth calculations. The two different cameras I will be discussing are Qualcomm Spectra Module Program and the camera for our project chosen given by our clients, the Intel RealSense Depth Camera SR305.

4.3.2.1 Qualcomm Spectra Module Program: This camera is unique because it is used on an Android device and was created exclusively for mobile users to experience extended reality (XR). The benefits to the Qualcomm Camera is that it can interpret depth on the go from a mobile device and can operate at an impressive rate of 60 frames per second. The cons of this camera is that it is on a mobile device and is not Windows, Linux, or Mac compatible. This results in a more difficult and not universal process of creating the applications [31].

4.3.2.2 Intel RealSense Depth Camera SR305: This entry level camera is not only affordable, but is made for indoors and has incredible quality at a 1.5 meter range. Once there is coding created for this device, the code can be transferred onto other Intel RealSense equipment. This makes the camera a perfect device to learn on. The Intel RealSense Depth Camera SR305 uses binary coded light to interpret depth. In order to do that, the camera repeatedly sends unique patterned infrared light to the focused object. By seeing whether or not each pixel is illuminated, the camera able to give a depth value for each pixel in that frame. The Intel RealSense Depth Camera SR305 has many other technological features that help the camera work effectively for sensing depth. Specifically, it has a rolling shutter and coded light technology[5]. Overall, this camera provides a number of qualities our team is looking for when it comes to using a depth camera.

4.3.2.3 Conclusion: Since Intel has provided our group with the Intel RealSense Depth Camera SR305, we will be using it. However, even if we had all of the depth cameras available for use on the market, we would most likely still use this particular RealSense camera due to its user and budget friendliness as well as being compatible on all of our groups' computers.

4.3.3 SDK

The next area that I am researching is the SDK for the camera. The SDK is a cross-platform and multi-language. It is well documented for use on Windows, Linux, and Mac. The SDK can have a wide variety of wrappers: C/C++, ROS, Python, OpenCV, Node.js, UnrealEngine, PCL, OpenNI, Unity, LabVIEW, Matlab, and C# [5]. The following information is for some of the wrappers we have considered using.

4.3.3.1 C++: There are many known advantages to programming in C++. From my personal experience, this language is well established, however, it is not being used as frequently as it once used to be. Our group is interested in obtaining previously coded databases and C++ is a slightly better documented system than Python. Although our

group as a whole feels comfortable using C++ as a programming language in general through our personal practice, we do not feel as confident using it for this database.

4.3.3.2 Python: Python is a well used and known language amongst our group members. Since Python is a more traditional language used in ML, we have an abundance of experience using this language. There are many known benefits with Python such as a decent among of documentation and its strong ability to interface better with the ML algorithm.

4.3.3.3 Conclusion: Both C++ and Python are known for their widely available resources such as tutorials and sample programs. Our group has come to the agreement that Python is the simplest and best language to use when programming the depth camera. C++ simply takes a longer amount of time to write which is not ideal for the timeline of this project Since Python is easier to interface with ML, our team will most likely be using Python as our choice language.

4.3.4 Data Collection

The third and final portion of the project that I am responsible for is data collection. For data collection, the technologies being used are the camera, our computers, and a database. Our team threw around the idea of using an external hard drive as a database, however, we quickly noted that hard drives have limited access (one person can access at a time) and have limited data storage. This is a straight forward time consuming process using mainly a database to store our findings. For data collection, our group will be using the camera to collect our data and storing it on a server. One of our group members created a program to take a snapshot with the depth camera. At the moment, those snapshots are being stored locally on our machines. However, we will need to store it on a database.
We can use that same program on a larger scale for the data collection. Some of the data that we are looking at learning is pixel color information and depth information.

4.3.4.1 Pixel Color Information: The Intel RealSense Camera uses a coded light system that projects a pixel to an object to determine its depth. The camera then identifies whether or not the pixel landed on a stripe of color or not [1]. With this information, it assigns a value to the pixel and then repeats the process over and over. Eventually, there are a series of values assigned to the pixel color, creating a data collection to interpret the depth of the object.

4.3.4.2 Depth Information: Through the pixel color information stated above, the camera will then continue to, ”uses multiple different coded light patterns that change over time to increase the accuracy of the depth image. The camera also features a 2MP RGB sensor to allow for fully textured and colored depth points” according to the Intel Website. Our group will be collecting this data from our 3D American Sign Language gestures done on the RealSense Camera.

4.3.5 Conclusion

Overall, our group will be using all three technologies: the Intel RealSense Camera SR305, Python on the SDK, and collecting a wide variety of data in regards to the depth information collected from the American Sign Language Gestures.

4.4 Tech Review: Nic Davies

4.4.1 Introduction

The goal of the Intel RealSense Gesture Recognition project is to use this camera in order to correctly translate gestures. We are planning to train a machine learning module with the help of database to classify gestures in real-time. We have

been able to split up this project into 3 main parts: User Interface, Software, and Machine Learning. This paper will focus on the User Interface and specifically the GUI that will need to be built. I will review the type of interface that can be reasonably built, language that should be used, and how to integrate the Intel software that is at our disposal.

4.4.2 User Interface

The user interface will be the first thing that anybody sees on the display. This display will show the camera feed real-time albeit with slight delay. This interface should show the classification of a gesture and the raw data that the camera is creating in the background. This section which type of user interface will best suite the project's needs. There are many options for the User Interface that needs to be built but I will only explore the ones we can reasonably write within the allotted time. Most machine learning projects that I saw at Engineering Expos from years prior, seemed to be tools that simply can be launched and did not a have a web based front-ends. I personally have written a few GUIs this way but I learned through my internship, that if someone is going to use this tool frequently, be export it to other systems, or play an integral role in some fashion; then it should be written in a web based format.

4.4.2.1 Window-based: Now writing a GUI this way seems to be the easiest route. For the work that I have done during my latest internship and for school have all been in Java. The best I believe for a window-based GUI is to use Java Swing or JavaFX.

These development tools have been claimed to be easy to use[20]. I can personally attest to this since I have experience with using both. I would not need to learn a new language or tool to use in order to create this application just how to specifically integrate live feed.

There are some limitations to using Java Swing or JavaFX. Since our group has not hard requirements for hardware or OS, there could be a possibility of choosing a something that does not have Java installed somewhere[3]. This really depends of the longevity of our work and whether Intel would like to take what we have and recycle it into something that they can use. Its a very niche problem that might not occur but still a drawback as we move to lower level problems. Security also has been a bit of a concern for Java applications[21]. I am sure there are some that remember the days of installing some sort of Java application or update to play games on the computer. The finished product can also standout in its environment as the layout can cause GUIs to look a little unpolished [21]. All of the GUIs I have created were for personal or internal use and don't need to be prepared for commercialization. It is my belief that it would be easier to create a web-based GUI that looks more polished than some Java Swing GUI.

4.4.2.2 Web-based: Hosting this interface on a web platform is also another viable option that could fit our needs. If we also have to deal with monitoring clusters and tracking our database, we can group this all together depending on how the project progresses.

Web-based GUIs have there uses and can be especially helpful when dealing with clusters, databases, etc. My work experience has led me to believe that making a GUI through this route is best looking, best for longevity, and widely used in the industry (really depends on the industry and the the preference of the project manager but these are my conclusions drawn on my knowledge). After reaching out to a coworker, I would probably use some sort of MEAN stack such as Node.js. Creating a web-based GUI is not tethered to any OS and can be easily ported to other platforms [22]. There are also apparently easier to deploy and can be written at a paced not matched by a window-based GUI [22].

Creating a interface like this will be difficult with my lack of experience programming this way. Using Node.js would require writing JavaScript which is something I have not touched since Sophomore year and could mean a great deal of

learning JavaScript in order to get this working. Different browsers can render images and views differently which could become a problem if the user interface needs to be relocated to a different browser and messes up all the rendering[22].

4.4.2.3 Discussion: There are certainly many pros and cons associated with each platform that this interface can be created as. I worry more about having issues right away with getting a web interface set up and working rather than building this as a window inside the Intel SDK. Though I can also fear that possibly down the line, that we reach a point where a window-based app won't be supported.

4.4.2.4 Conclusion: Both options seem to be perfectly acceptable for the requirements. Depending on who is on the User Interface portion of the project, they can use whatever framework they have the most experience with but I would recommend a window-based since it will most likely be the easiest to use.

4.4.3 Language

There are many languages that this can be written in. The most obvious choice should be Java as it is the language I have the most experience in it. I would be interested in writing this in Python as I am comfortable coding in it as many classes have allowed us to use it for homework. Although Java is the usually the go to for GUI creation, Python could be a good alternative if such developing tools exist like Java and Java Swing.

4.4.3.1 Java: Java has a myriad of tools to develop GUIs and has so for quite a few years. It has been the only language that anyone has recommended for me to use and the only language I have used besides JavaScript that interacts with the front-end dynamically.

As previously described, this language has many tools to use in order to create this GUI. Discussed previously was Java Swing and JavaFX as ideal. I am very familiar with this language and the tools mentioned and would allow me to spend more time trying to perfect the GUI rather than learning another tool and try to get it barely working.

Other than the aforementioned cons of using the Java tool-kits of Swing and JavaFX, Java programming could be difficult integrating with the existing user interface that Intel gave us. I personally have not had a chance to tinker with RealSense Camera and see what can be salvaged and integrated.

4.4.3.2 Python: After doing some research, it seems as if there are plenty of SDKs available to Python and this could be a avenue to explore[23]. I will look specifically at TkInter for the sake of time and word count.

As quoted from the article, "TkInter is the grand old man of the Python GUI world, having been distributed with Python almost since the language was founded. It is cross-platform and omnipresent, stable and reliable, and easy to learn, it has an object-oriented, brilliantly Pythonic API, and it works with all Python versions, but it has some drawbacks as well [23]". There would be no question if this worked cross-platform as it seems this tool-kit was not long after Python itself came to be. The source code for the demo of the kit seems pretty straight forward and similar to Swing. I managed to find a an article that used TkInter and OpenCv that were able to get a photo-booth app running with live feed which could be the ground work to get a bare-bones version working[25]. If this can be modified to properly work with the RealSense camera, then this can be very doable[24].

The article itself describes the tool-kit as a bit basic and lacks a few features and options not available in other tool-kits[5]. It also explains that the end product theme is straight out of 1985[24].

4.4.3.3 C sharp: C sharp is essentially Java but by Microsoft. I only am including it as it might be scattered throughout the Intel SDK and might prove to more useful than I would assume.

I want to state the obvious that this is a Windows language [26]. I have found a lot of pros for this language for back-end programming but I doubt that will prove beneficial purely for the front-end [26]. It apparently is quoted in

this article that it is a common language that programmers learn yet I have never had a course here that has forced us to use C# [26].

I really don't believe this language might be much of a contender against the 2 giants it faces. It is good for .NET programming but that might only be useful if we go that direction for our project [26]. What also worries me is the lack of coding articles and help available for it. I managed to find someone's implementation of a calculator but I don't want to code in a new language that doesn't have much support from the programming communities via online forums.

The research for this language has

4.4.3.4 Discussion: Java and Python are solid languages to code in for not only the back-end but the front-end as well. It really comes down to the experience of those who are going to implement this. I don't believe that C# can give either of those a run for their money since there are so many tools and support for both of them.

4.4.3.5 Conclusion: It seems that there are quite a few options that exist for a good language to use. It really depends on what the teammates that implement the user interface decide. I would not recommend C# unless the team members inside this are well-versed in it. I would recommend Java in order to use Swing but Python doesn't seem to have that many trade-offs.

4.4.4 Intel RealSense Library

The Intel SDK given to our group has revealed that there a some basic features and user interface that show the data flowing in and out. This code is supposed to be written in C#, Java, and mixture of C++. It basically just shows us the camera feed and is very bare-bones. There might be a possibility that we can develop directly inside this however in the time I spent working in it, I can barely get anything to work.

4.4.4.1 Develop within: This will require a lot of work but could be our best option. Within this project, we might have access to far more resources in order to get this user interface hooked with at least the camera for now, the database will be another story. We might be either coding in Java, C++, or C sharp so there's a good possibility we could continue to use the tools I would use to complete this task. There is a good possibility that we might be limited inside here and have to use a tool that can be accessed internally.

4.4.4.2 Develop externally: The only realistic pro for this would be to use languages and tools we know well. I am not sure how developing outside would work. I am far more concerned to get something up and running only to find out that it is not possible for it to work. Rather have the beginning of this be learning the Intel code rather that abandoning a external tool that cannot work after further research.

4.4.4.3 Discussion: I do not believe there is much discussion here. I believe we should bite the bullet and stick with the SDK Intel has provided. I would elaborate more on what that would look like but our group is waiting for an additional camera so that more of us can begin starting to research some more.

4.4.4.4 Conclusion: I believe we should develop inside of the SDK and not try to deviate from that framework. I don't think it would be wise to develop externally but rather research heavily into what can code in inside. Its a far safer bet than making a standalone user interface.

4.5 Tech Review: Zhidong Zhang

4.5.1 Introduction

My group is worked for the Gesture Recognition using a new Intel Real Sense coded light camera. We have six members in our group. And my research and review technologies or methods is about to translate gestures into other languages,

Dimensionality Reduction for machine learning model and research the gesture Language. Now, translation technology is limited to speech. And our group wants to improve translation technology. Someone has difficulties using speech as a way of communication. Our ultimate goal is building a new translate technology to help them to communicate in more effective way. And We will work to translate gestures into text by using a new Intel Real Sense coded light camera. This technology will do for gesture recognition and translate it into text. It will create a new translation technology.

4.5.2 Translate gestures into other languages

4.5.2.1 Overview: Translate into English is the basic function of our project. We must do that part. And it needs to machine learning model to help to finish that part. Translate gesture into English is pretty like speech translation. It is a good example for reference. Speech translation can promise 100% accuracy. Speech translate does for each word and combines it to become a sentence. We can learn what to do for our gesture Recognition. We also should translate each gesture in word by English. Then the GUL will combine those words to a sentence and show it. The definition of gesture will put it in the database through a machine learning model. There have two ways can do translate gestures into other languages. One is translating a sentence into other languages by google translate. Another is adding more definitions with other languages to the database.

4.5.2.2 Method: Gesture Recognition $\rightarrow$ Translate into English $\rightarrow$ Use google translate to translate to other language $\rightarrow$ Put Gesture Recognition definition with other languages into the database.

4.5.2.3 Use google translate to translate to other language: The first method has advantages and disadvantages. Translate into other languages by google translate seem easy for working. For using google translate, we can get it for free and quick. Also, google translate have a big database for translation. Google translate can support a fast way to translate English into other languages. If the project using this way, it can save a lot of memory in the database. However, it exists some disadvantages. First, if the project use googles translate, it must work online. Google translate often produces a translation that contains a lot of grammatical errors. It means our project may not translate correctly in other languages. It is a challenge. About the translate methods for using google translate, we can have two choices for that. One is translating each word and combine those into a sentence. Another is translating a sentence. If using the first one, that may need a heavy workload and more time to run. But it can increase accuracy. For the second one, translate a sentence will faster, but it will decrease accuracy for translation.

4.5.2.4 Add definition about other languages in database: Putting Gesture Recognition definition with other languages into the database is the better choice than the first method. For that method, we don't need different technologies. It only requires to add a definition with other languages (not English only) into the database when we train the machine. For pros, it can improve the accuracy of the translation. Also, it doesn't need technical help from other companies. However, the cons exist. This method should have a big database for working. And we need to collect more information about the training machine. Also, the gesture may have a cultural difference. The different gestures will have a different meaning in different countries. That is what we need to collect. Collect those databases need a long time to do that.

4.5.2.5 Conclusion: These methods have advantages and disadvantages. If our project needs to do less in the database and trust google translate, the first method is the better choice. The second method will need a long time to perfect the database. But the second method can translate gestures offline. The second method will be the better one in those methods.

4.5.3 Dimensionality Reduction for machine learning mode

4.5.3.1 Overview: There is a lot of gestures to define. It means it should need a high memory to save those data. This memory is a vast amount. It requires a lot of pictures to identify each gesture through machine learning. Also, it needs the store definition for each gesture. If we use a usual way to handle it, it may need a huge memory and a lot of time to finish the part of machine learning. So, Dimensionality reduction is necessary for machine learning mode. There are two methods of dimensionality reduction for reducing memory and do well on machine learning. The first one is selecting features in images. Another one is reducing the memory of the picture.

4.5.3.2 Method: Selecting features in images is the most important part here. When we want to gesture recognition, we need to find out what features we need to force. Those force points are called feature in here. When finding out feature, machine learning only forces feature when recognizing the gesture. This method can reduce the workload for machine learning. When we train the machine, it doesn't need to train for all of the pictures. It only needs to do for the feature. That way is using dimensionality reduction to optimizing machine learning.

Reducing the memory of pictures is necessary. For a normal picture, it should have more than 1mb. And we need more than one thousand pictures to training machines to recognize a gesture. If we use the standard way to do machine learning, each gesture needs more than 1000mb to finish that process. Our database can't accept a huge memory for saving pictures into the database for gesture recognition. Now, there is a way about dimensionality reduction to reduce memory for each image. As the research about selecting a feature, the feature of pictures in our project is hand. So, we can ignore all the information except two hands. And we can reduce pixels for each picture. Those ways both can save a lot of memory for a training machine.

4.5.3.3 Conclusion: Totally, finding the features in images is the first step to do. Then we can reduce the memory of pictures through exclude information other than features. These two ways will be helpful for dimensionality reduction. If our project uses those ways, it can improve the speed of the training machine.

4.5.4 Research the gesture Language

4.5.4.1 Overview: There is a lot of types of gesture language in our world. Different countries have a different types of gesture language. Different gesture language has a different definition. If our project wants to face the world, it means it needs a huge memory to do that. But now we only do one kind of sign language for one part of users. That gesture language is ASL (American Sign Language). When we start to write the function for gesture recognition, we should research gestures first.

4.5.4.2 ASL: In ASL, it has 26 signs and 10 signs for numbers. The grammar is important for ASL. ASL is an SVO (subject-verb-objectct) language. For example, there has 4 sign meanings. Father – love- child. That will mean the father loves the child. Also ASL allow OSV(object – subject -verb) For example Child(topic), Father - love. That is also meaning the father loves the child. Even more, ASL will repeat the subject at the end of the sentence. For example, Father-love- child- father. It means the father loves the child. It's the same meaning with SVO. And ALS will allow a null subject and a subject copy. It seems like OVS, For example, Child(topic), love-father. It means the father loves the child. Those are grammar belong ASL [32].

4.5.4.3 Conclusion: In general, we know the grammar of ASL. The acknowledge of grammar of ASL will useful in our project. It will be helpful for our group to find out logic for combining a sentence.

4.6 Tech Review: Shihao Song

4.6.1 Introduction

We can recognize gestures by using Intel Real Sense coded light camera, and use the database for machine learning.Our team divided the six team members into three groups, respectively responsible for the development of UI, SW and ML.The ultimate goal of our project is to translate gestures into text to help people with communication disabilities communicate normally.For this purpose, I will discuss these three topics:

-Natural Language Processing

-Neural Nets and Deep Learning

-The Organization of The UI

4.6.2 Natural Language Processing

4.6.2.1 Overview: Natural language processing is a technique used to help computers understand human natural language. The goal of the NLP is to allow computers to read and understand human language. NLP is a very difficult technology because the nature of human language makes NLP very difficult. This can make the computer's identification inaccurate or ambiguous because the computer does not recognize abstract or higher-level language performance. NLP is used in a wide variety of applications, such as Google's translators, or a series of personal assistants such as Siri and Cortana.

4.6.2.2 How dose NLP works: Communicating with computers in natural language is what people have long sought.Because it has both obvious practical significance and important theoretical significance. People can use the computer in the language they are most used to, without having to spend a lot of time and energy to learn various computer languages that are not natural and customary. People can also use it to further understand human language capabilities and intelligent mechanisms.

Natural language processing (NLP) is a field of computer science, artificial intelligence, and linguistics that focuses on the interaction between computers and human (natural) languages. Therefore, natural language processing is related to the field of human-computer interaction.There are many challenges in natural language processing, including natural language understanding.Many challenges in NLP involve natural language understanding, that is, computers derive their meaning from human or natural language input, and others involve natural language generation.

Modern NLP algorithms are based on machine learning, especially statistical machine learning.Deep learning to understand natural language is achieved through an encoder. For a sentence in natural language, the machine has no way to directly understand its meaning. We need to use an encoder to convert the sentence into a code that is easier for the machine to understand.

NLP library can be used when developing NLP, which can greatly reduce the difficulty of development. For example, Natural language toolkit (NLTK),Apache OpenNLP, Stanford NLP suite,Gate NLP library. Among them, the Natural Language Toolkit (NLTK) is the most popular natural language processing library (NLP), which is written in Python and has a very strong community support behind it. NLTK is also easy to get started, in fact, it is the simplest natural language processing (NLP) library.

4.6.2.3 Conclusion: NLP is a very attractive way of human-computer interaction. It allows computers to use limited vocabulary sessions, which is a great convenience in our projects. However, NLP is an extremely complicated technology. We cannot think about computer thinking with human thinking. So in the future work, we will work hard on how to let the computer "understand" the gesture.

4.6.3 Neural Nets and Deep Learning

4.6.3.1 Overview: Neural network is an algorithm for modeling and recognition based on people's brains. This algorithm interprets sensory data by machine sensing, marking, or aggregating raw input. All data, including image sounds and text, must be converted to numbers. Deep learning is part of an artificial neural network. The information obtained during these learning processes is very helpful in interpreting data such as text, images and sounds. Its ultimate goal is to enable machines to analyze and learn like humans, and to recognize data such as text, images and sound

4.6.3.2 How do we use deep learning: Regarding deep learning,I prefer to use Python.Because most deep learning libraries use the symbolic language method instead of the imperative language method.We use the camera to recognize gestures, essentially using Deep learning to identify images.I think we need to find the basic code of a neural network first,and then further programming according to our needs.

4.6.3.3 Conclusion: Regarding deep learning, I think we definitely need to use it, but this technology should be too difficult for us. After some searching for information, I found that deep learning is not to make an artificial brain to think, but to establish an artificial neural network, and then through algorithms to let the computer ”learn” something.

4.6.4 The Organization of The UI

4.6.4.1 Overview: The UI of a program is very important. Sometimes a good program is often abandoned because there is no good UI. So it is necessary to make a good UI. A good UI needs to have a series of features such as accuracy, reasonable layout,convenient operation,and short system response time.The user group that our program faced is social barriers, so the UI design needs to be optimized.But we want to use our program not only for people with communication barriers, we hope that more people will use this program. So you need to consider more comprehensive.

4.6.4.2 Method: First we need to consider how to make the program operation more convenient. Simulating realistic interactions will give people a familiar feeling, and users can understand without learning. For example, when a button is pressed, it will be deeper than the original color. This is the effect of realizing the light and shadow in reality. The user's first contact, no need to learn, unconstrained use of the application, is a good application.We also want to make the interface clean and tidy. In order not to interfere with the user, no necessary decorations are added. In order to make the interface clean, it is necessary to hide some unimportant functions, but users can find these functions in an effective way. For example, the menu interface can be hidden in the upper left corner and can be expanded by clicking on it.Regarding icons, try to use generic icons. We are not used to trying to change the icon that the user is used to. Using universal icons will make it easier for users to use, and it will be easier for users to feel intimate with the interface.Regarding the color matching, the combination of black, white and gray is very good. These three colors make the interface look neat and layered.

Because it is designed for communication barriers, we need to do targeted design based on the previous UI. Before creating the UI, we need to understand the situation of people with communication difficulties. Since we are targeting people who need to use gestures to communicate, we will make improvements in this area.First of all, the camera's button will always be displayed at the bottom of the page, which will make it easy for the user to find out how to turn on the camera at any time.Then we will quickly recognize the meaning of the gesture and display it at the top of the screen so that the user can filter what they want to use.

4.6.4.3 Conclusion: In short, we will combine the two to achieve smooth and unobstructed use of our programs for both normal and communication-disabled people.

5 Weekly Blog Posts

5.1 Fall Week Three

Group	Positives	Deltas	Actions
All	At this point our group members have all met one another, established team standards, and have a high level understanding of the project that we have been assigned. We have made communication with our client and are figuring out what the best way is for our group to receive the camera hardware needed for our project.	A weekly meeting time and medium between group and clients has yet to be established. Team needs to come to consensus on potential meeting times to offer to clients, as well as figuring out how the camera will be received.	Email will be sent on behalf of entire group that contains potential weekly meeting slots that our entire group is available, along with shipping information for the chosen group member that will initially receive hardware.

5.2 Fall Week Four

Group	Positives	Deltas	Actions
All	Our group has completed one of the major project documents, The Problem Statement, which outlines the project at hand and its different components at a high level. We have also established weekly meeting logistics with our client and established who will initially receive the camera amongst our group members.	Our group has completed the first draft of the project's Requirements Document, however the document still needs finalization and feedback from the clients, prior to the secondary draft's submission.	Finalize the remaining portions of the Requirements Document, and actively make changes to the document prior to submission, as feedback from the client arrives.

5.3 Fall Week Five

Group	Positives	Deltas	Actions
All	Group has completed another major project document, The Requirements Document, which outlines the required resources and system requirements needed for our project. Furthermore, our group has also received the camera hardware equipment and begun exploring SDK and camera capabilities	Begin exploring implementation options for the camera's SDK, and familiarize with libraries and functions. Begin dividing project components for group member research in the technical review assignment.	Divide the project into separate research components and assign these components to group members for research in the technical review assignment.

5.4 Fall Week Six

Group	Positives	Deltas	Actions
All	Each member has completed their technical reports based on topics of their choosing. At this point, each member should be well-versed in a subject area in a team that they might be placed in.	Teams for User Interface, Software, and Machine Learning project components need to be decided. Polls have been sent and by next week everyone should hopefully be in a team of their preference	Group needs decide the partition for project components, and each member should familiarize themselves with their tasks after the partition has occurred.

5.5 Fall Week Seven

Group	Positives	Deltas	Actions
UI	The UI team has been established as Shihao Song and Jonathan Hull. We will be collaborating to create the UI for the project.	For this project our team will be using PYQT5 to make the UI. Both team members have not used PYQT5 before and will thus need to do some research and tinkering before getting started.	Research must be conducted in order to acquire more details for what and how this UI will be created.
ML	Team division for the Machine Learning project portions has been established, with Ulises and Zhidong being the two members working on this portion of the project implementation.	Both members are not well versed in the area of machine learning and need to overcome a steep learning curve in order to fully understand the topic for integration into our project.	Research by the team needs to be conducted in order to identify a candidate implementation algorithm for the ML project portions.
SW	Intel has sent the team another camera which will allow us to split a camera per team. Groups are now solidified and Nic and Shane are the 2 people that will be working on the data integration, storage, and normalization for this project.	Nic has joined Shane in the Software Group. Shane is already knowledgeable in this area and wrote the technical paper on the components of this team. A small script has been written to see the raw data for the camera.	Continue to develop the script to reach a testable state and explore the possibility of a database storing the image data.
All	Group partitioning for project components has been established.	Project components have yet to clearly define goals/tasks, and define timeline for component goals.	Think about project pipeline in terms of separate project components, and the interaction that must occur between these entities. Define component goals and create timeline for these goals.

5.6 Fall Week Eight

Group	Positives	Deltas	Actions
UI	PPYQT5 has been downloaded.	No experimentation has begun. Lack of experience and knowledge of PYQT5 has slowed progress.	Research and experimentation is still required to find out how the UI will be created.
ML	A Convolutional Neural Network (CNN) has been identified as the chosen algorithm for the ML project portions. Furthermore, PyTorch is the chosen deep learning framework that our group will utilize in order to build our ML model.	Again, lack of exposure to the PyTorch framework and knowledge in the domain of Convolutional Neural Network is the main impedance to our group. Group members require more research in these domains in order to confidently begin implementation.	Research into actual implementation of a CNN algorithm utilizing the PyTorch framework is the next area in which our group will direct efforts.
SW	We have solidified that we will use an HDF5 file for data storage as a database doesn't seem feasible with the time being. We will be writing most of our code in Python which both of us have prior experience in.	More research was being done for a Design Document due at the end of the week and it has become very clear that communication between the UI and ML teams must exist prior to any lengthy goals being reached.	The Design Document has given us much time to reflect on the processes and implementation that needs to fleshed out before we can begin any sort of tests or gesture training.
All	Project Pipeline has been created and tasks for different project components have been clearly identified	Design document requires finalization and has yet to be reviewed by the client.	Finalize and submit the design document, and respond to feedback provided by our client.

5.7 Fall Week Nine

Group	Positives	Deltas	Actions
UI	We have a mock up for what we need to make. The mock up gives us a fair amount of liberty in design decisions.	Because of Thanksgiving break there was almost no change.	Continue to research and experiment with PYQT5. Develop a design for the UI.
ML	With the project pipeline in place from the completed Design Document, our group is now familiar with the potential types of normalized data we will be using as input to our model.	Research needs to be conducted on how to tackle our project in the context of video classification. This is a different problem from classification with just a single image input, so our group must research how to deal with this classification problem in the context of our project.	Group members will conduct research on utilization of CNN models with video classification tasks.
SW	We have laid out the flow of the code that needs to interact with the UI and ML teams. Next term, there will be plenty more time to implement code when the documentation is mostly out of the way.	Due to the abbreviated week, there was very little change.	Continue to implement code following the flow outlined in the Design document. These parts are crucial in allowing the project as a whole to run cohesively.
All	Design document and project timeline have been finalized and submitted to both the client and instructors.	Little implementation testing has been completed and these are the next steps needed to be taken.	Prepare for implementation phases for the respective project components.

5.8 Fall Week Ten

Group	Positives	Deltas	Actions
UI	The UI team will start working on the first UI prototype over Christmas break.	No change for the UI team.	First prototype is too be finished by the new year.
ML	Research by our group has revealed that similar approaches to video classification known as FlowNet models may be utilized by our group.	Research must be conducted by group members on implementation of FlowNet models with toy dummy sets, then be able to expand to the context of our project.	Research the domain capabilities of the FlowNet model, and potential implementations in domain of our project.
SW	We have been able to accomplish some normalization, and are looking into how we are able to do further normalization to create more consistent data for the model. We are also looking into how we might tell the user to move from the UI if their position is not appropriate.	Need to do more research into the outlier detection and how we would like to find an outlier detection method that would be not extremely time complex.	Finish normalization and work on outlier detection if possible. Lots of testing needs to be done on data as well as testing on collecting large amounts of data.
All	Group has completed all major documents required for the first term, and is fully ready to begin implementation for the project after the initial documentation steps have been completed.	Only infant steps have been taken in each groups, and much work is still needed for implementation steps across every team. A greater effort needs to be put forth in order to accomplish project in a timely manner	Begin actual implementation of project steps, according to Gantt chart timeline outline in the Design Document. Each group should strive to meet first major milestone from the timeline.

5.9 Winter Week One

Group	Positives	Deltas	Actions
UI	Research and experiment with PYQT5 to be able to make a development UI.	Collect information and know how to design UI with PYQT5. Ues Qt Designer to create UI.	Think of a model of the basic UI and implement it.
ML	Experimented with PyTorch module using toy data sets and confirmed running ability on OSU cuda cluster.	Need to employ actual data instead of simply running on toy data set	Continue experimenting with script modifications to work with actual RealSense camera data.
SW	Over the break, we were able to get started on and complete most of the basic data preprocessing that we need in order to start training our machine learning model. We are able to capture the depth image data, normalize this data, and setup hdf5 datasets for the training x and training y sets (for classification).	We need other components to be more functional in order to start integrating our components into a product.	We want to look into outlier detection and work with the machine learning team to coordinate machine learning input. Right now it might be unclear about how to obtain the data from the dataset that has been setup, so we will work with them on extracting and feeding that data into the machine learning model.
All	We have also reached out to our TA and client to setup meeting times for the rest of the term	Have yet to hear back from clients to establish weekly meeting times.	Continue to work on individual components as we aim for eventual integration.

5.10 Winter Week Two

Group	Positives	Deltas	Actions
UI	Made a development UI. Research and experiment with making an end user UI.	Make the basic UI with QT designer, and then make the bottom of the UI clickable through programming.	Need to optimize UI more.
ML	Built PyRealSense library in order to utilize camera on Mac OS. Experimented with creating .png images from data collcted with camera.	Still need to research how to modify existing script to utilize our actual camera data.	Look into using PyTorch module with current camera data exported from .hdf5 files.
SW	We feel that the software inter connectivity portion is almost to a point where it can be called done. We just need to write a few functionalities that will help transfer information to the front end GUI.	Work on ML integration.	We are looking to try and help the ML team with their model and tuning the model so that we can work on the other portions of the project before the alpha release.
All	We have adjusted and cleared up some of the constraints on the project with our client. We now feel that the project can be accomplished within the time span that we have. We have been working on setting up the ML model and figuring out which framework we want to use based on our data format.	All project components still need to be implemented in order to begin the integration steps.	Continue to work with one another to ensure that we are able to meet the alpha release deadline.

5.11 Winter Week Three

Group	Positives	Deltas	Actions
UI	Created a prototype end user UI. Now we need to make it look better and add connectivity with the SW when they are ready.	UI needs to be integrated with actual code.	Need the help of the SW team to connect the program to the UI.
ML	Transfer learning script now working with our specific camera data. Only two classifications made so far, 'a' and 'b'.	Many portions of the script are currently hard-coded to work with existing values, and refactoring needs to take place.	Next steps include refactoring script and exporting/importing models and working with live data.
SW	Most of the software is now complete, we just have to work on saving the current camera images to a file for the UI to read and display.	Continue working on integration with UI	Help solidify and test the SW and work on finishing the ML so that it can properly train on all of our classifications. Complete the building of our dataset so that our model can become more accurate. Work with the UI team to make sure that all of the SW functionalities are complete in order for the GUI to be built.
All	This week we have made lots of progress on the development of the ML model. We have begun training and testing on two classifications, and been working on broadening that to n classifications.	Some project components need refactoring in order to aim for integration with one another.	Continue working on refactoring code to aim for completion of alpha release deadline.

5.12 Winter Week Four

Group	Positives	Deltas	Actions
UI	Optimize the UI to make it look better.	Added background image. Replaced button image. Redesigned UI, plus notes and instructions.	Add new features to optimize the UI
ML	Transfer learning script refactored to work with 'n' classifications.	Still need to work on model import/export that is crucial for integration.	Continue researching on model import/export methods.
SW	Starting to work on integrating the machine learning model into the end user interface	Debug the UI more to connect with software	Need to do more debugging for the integration and housing of the model in the UI
All	Accomplishment of ML refactoring is a huge step needed towards eventual integration, and major project strides being made.	Final components prior to integration still needed to be completed across UI and ML teams	Continue working with one another and making continual progress towards crucial integration of project components.

5.13 Winter Week Five

Group	Positives	Deltas	Actions
UI			
ML	Appropriate model import/export conducted, and testing with live data appears to yield desired results.	Data collection for more ASL letters still needed to full project implementation.	Continue working with team members on data collection for expanded classification domain.
SW	This week we have worked on integrating the trained ML model into the UI to classify gestures in real time. We also became responsible for completing the poster for the group.	Integrate further with the UL and ML.	We are going to continue integration and work on training the model further to see increased accuracy. Work has begun a first draft for the poster.
All	Integration has begun, which is a crucial step to accomplishing our alpha release.	Our model is still not that accurate and we need to collect more training examples	Continue working out remaining integration work and exposing bugs to our integrated product.

5.14 Winter Week Six

Group	Positives	Deltas	Actions
UI	Worked on Integrating UI to work with SW	Add functionality that reads in produced images from SW	Receive camera images and ML output from SW
ML	Further data collection completed, and work on the poster has been conducted for ML poster portions.	Most eminent need at this point still resides in the data collection domain.	Focus on poster implementation, but continue collecting data whenever appropriate.
SW	Our work on the poster has continued but is nearing an end. There are a few minor parts still not completed but it good enough for a first draft	Finish the rest of the poster.	We worked on finishing up the poster.
All	Our poster is nearly out of the way, and group can resume focusing on development of project.	Poster still needs finalization and approval from clients.	Continue working with clients to gain approval/feedback and implementing needed changes.

5.15 Winter Week Seven

Group	Positives	Deltas	Actions
UI	Fixed file type error. Got pictures from the camera to show in UI.	File type issue that was resolved. PYQT5 in current set up would not work with JPG images and need the images in JPEG.	Enhance current UI
ML	Working on the ML design review portions, and ensuring that all members of the team are up to speed with current implementation.	Needing to practice run-through of presentation to ensure preparation for peer presentation.	Practice with group members on how we will talk about our project in the context of the design review.
SW	Preparation for the upcoming Design review has begun. We have been working on the slide deck for the Software side.	Work on preparing for question for design review	Performed a knowledge transfer between us since the groups are being split up for the presentations. We will speak on behalf of both of us. We continued to make changes to the poster based on feedback from client.
All	Alpha release completed and ready for presentation in design review.	ML model classification not 100 percent accurate.	Continue working on fine-tuning model for appropriate display to peers.

5.16 Winter Week Eight

Group	Positives	Deltas	Actions
UI	Tested Integration. Tested UI on teammates. Looking for major usability errors	Resolve any possible usability errors	Add text behind images saying to press the capture button to begin.
ML	Helpful feedback has been provided by peers, and namely the advice of utilizing the colormap depth image for model input.	This recommendation may require ample work to be completed.	Evaluate the feasibility of implementing the recommended changes.
SW	Design review has occurred this week. Feedback received has indicated that we might want to look into replacing the HDF5 format for storing our data-sets.	Work on implementing beta funcutionality changes	Research has been performed on other viable options for data storage. We also reached out to our clients to ask for advice as to this change in direction. We continued to make changes to the poster based on additional feedback from client after they reviewed a revised version of poster.
All	Overall, the design review went well across our team.	The recommended changes slightly alter our original proposed milestones and deadlines.	Group needs to evaluate the possibility of implementing recommended changes.

5.17 Winter Week Nine

Group	Positives	Deltas	Actions
UI	Worked on finalizing the poster for re-submission, and discuss potential UI refactoring strategies	Have not had much time to sit down and discuss potential re-factoring and optimization strategies for UI end.	Sit down and continue discuss potential strategies moving forward.
ML	Sat down with SW team and thoroughly talked about how the recommended changes may be implemented.	Required work on the end of term document, and video impede the potential to begin implementing these changes.	Work on the document and video so that we may focus on desired changes.
SW	We have finally added the missing group photo to the poster and await feedback from clients. Some reorganization was necessary to accommodate the additions.	Work with other teams to integrate beta functionality for SW	We revisited and revised the poster again.
All	Finalized the poster and received re-grade based upon changes.	Need to begin working on end of term document and video.	Begin working on the final assignments of the term.

5.18 Winter Week Ten

Group	Positives	Deltas	Actions
UI	Worked on completion of the end of term document and video, and provided necessary components for allowing completion.	Still have not finalized optimization and refactoring strategies for final product release.	Same as prior week, continue planning for potential strategies moving forward.
ML	This week, we worked on compiling the entire group's videos into one video for submission. We also contributed to writing and editing the end of term document.	Not much effort went in to plan development due to the work required on compiling videos and end of term document.	Now that completion of the video and document is completed we may focus on the development needed for our planned refactoring.
SW	This week we worked on the end of term document and the demo video. We were able to complete all of the software sections for both and present them both to our clients.	There has not been much progress on implementation due to the document turn ins.	To work on beta functionality and completing our second machine learning implementation so that we can test the accuracy with the first model we implemented.
All	This week the team worked extensively on the end of term document, as well as the video. Each team member submitted their video portions and we were able to complete a demo as well.	We have not worked too much on the implementation changes we would like to make for beta functionality.	Work on implementing a second machine learning model after software changes to optimize preprocessing.

6 Final Poster or Image from Showcase

COST EFFECTIVE COMMUNICATION

The goal of this project was to explore cost effective medium of communication that would be utilized by the deaf community

This project uses a trained Machine Learning model to identify ASL gestures coupled with a pre-trained database that classifies the gesture made.

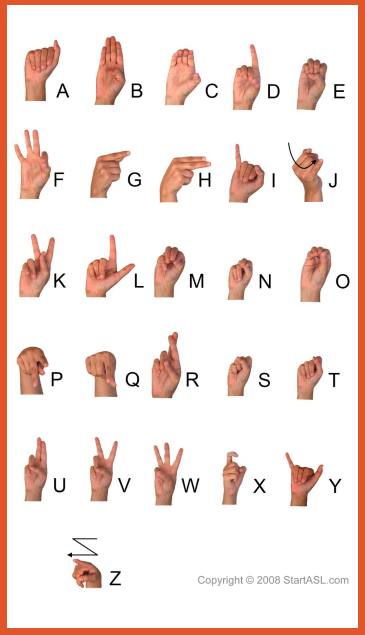


Figure 2: ASL Alphabet (Photo Credits to https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fres.cloudinary.com%2Fstartasl%2Fimage%2Fupload%2Fwp-content%2Fuploads%2Fstartasl%2Fasl-alphabet_wallpaper_1080x1920.png&f=1&nofb=1)

Oregon State University

GESTURE RECOGNITION

ASL Gesture Recognition using new Intel RealSense light-coded Camera

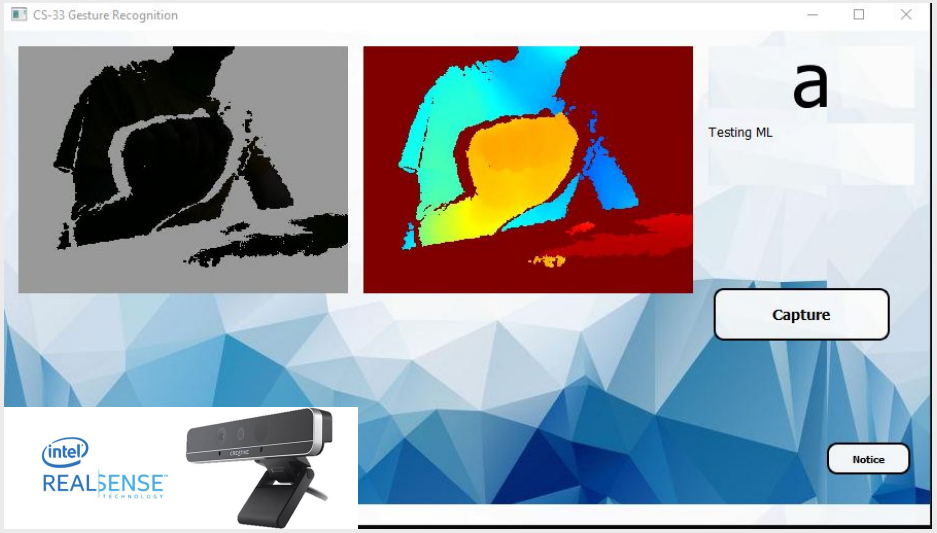


Figure 1: Demonstration by Jonathan Hull for the ASL letter “A” in testing mode (Photo Credits to Jonathan Hull)

CLIENTS

The clientele for this project are Eduardo Alban, Satoshi Suzuki, and Po-Cheng Chen from Intel. Intel allowed the team to utilize their librealsense library as well as providing multiple Intel RealSense cameras. The clients have been able support the team with changes in direction and advice throughout the project. Weekly meetings has allowed the team to receive valuable feedback such as additional features for the UI which helped it to appear more user-friendly.

intel

PIPELINE

The software side of this project was essentially a pipeline to process and send data back and forth from the UI to the ML Model.

- Once the capture button is clicked, the depth-per-pixel is extracted via the RealSense camera. If the training mode is not activated, then it is saved to the HDF5 file, otherwise it is then normalized.
- Our normalization process inverts the pixels, rescales and re-centers the image, and then finally normalize the depth.
- The normalized image is then sent to the ML Model, and a string is then outputted.
- This classification then is presented to the user via UI. Figure 3 illustrates the pipeline process of the project.

ABOUT THE TEAM

Jonathan Hull: User Interface Team
Shihao Song: User Interface Team
Ulises Zaragoza: Machine Learning Team
Zhidong Zhang: Machine Learning Team
Shane Clancy: Software Team
Nicholas Davies: Software Team

Figure 3: Team Photo (Photo Credits to Vishnupriya Nochikaduthekkedath Reghunathan)

ROLES

- The User Interface Team created the GUI for the project. What was produced was a 2 windowed interface with a button on the right to capture the gesture, and the Machine Learning Model classification on the right side of the screen with an accuracy associated.
- Software Team focused on data integration. This team managed data storage, image normalization, and communication between the UI and the ML model.
- Machine Learning Team dealt with creating the machine model used to classify gestures. Specifically, they used a convolutional neural-network, with a transfer learning approach with a pre-trained on a ResNet-18 dataset.

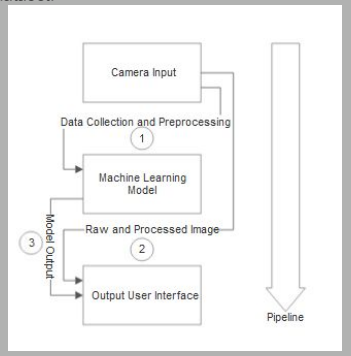


Figure 3: Pipeline Diagram (Photo Credits to Design Document)

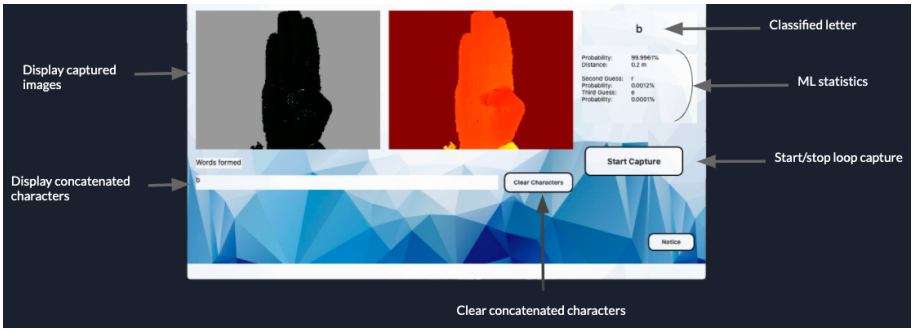
7 PROJECT DOCUMENTATION

7.1 High Level Information

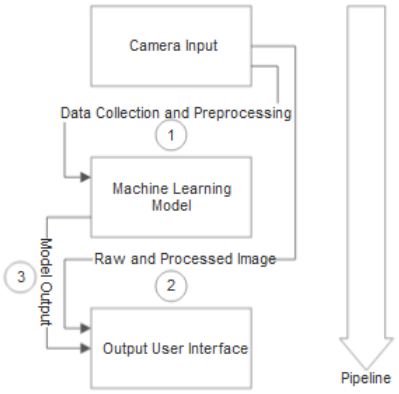
Our project works by users interacting with the Intel RealSense camera, and the GUI. The GUI serves as the main access point for users, and allows them to interact with both the camera and the ML model. The workflow is as follows:

- User plugs in camera to computer
- User launches project UI (endUserUI.py)
- User begins gesturing ASL alphabet letter in front of the camera
- User clicks the 'Begin Capture' button found on the GUI
- User may re-gesture after the first intial capture is made to continue capturing, or simply click 'Stop Capture' to end the capturing/translation sequence.

A picture of our GUI and an explanation of its separate components can be found below:



The high level view of our projects internal structure can be highlighted exceptionally with the image found below:



It highlights the flow of data from the user, into the model, and back the user. More specifically, camera input is collected via the GUI, and preprocessed by the SW methods. Preprocessing techniques include normalization and removal of background to reduce the effect of outliers and noise on the image. After image processing, the input is then passed to the ML portion where the CNN model classifies the input based off of a letter in the ASL alphabet. Finally, the user sees this classification along with a probability associated with the input classification.

7.2 Installing/Running Project

The following guidelines/steps assume you have access to our group's project github, found at: https://github.com/CS-33-Gesture-Recognition/Capstone. These steps can also be found in the repository README.md file.

7.2.1 Windows

Download python 3.7 from this release page https://www.python.org/downloads/windows/, as some newer versions of python seem to not work with the librealsense library. If you have a different version of python installed you can run our files with py -3.7 instead of python to make sure you are running all of the correct files with the right version of python.

7.2.2 Dependency list

The following dependencies need to be installed through pip by doing the following. pip install -r requirements.txt

7.2.3 Mac

The librealsense library will need to manually built and installed due to lack of support from the librealsense library for Apple OS. Parts of the instructions originate from:

- https://github.com/IntelRealSense/librealsense/blob/master/doc/installation_osx.md

The required steps for the Mac OS librealsense library build are as follows:

- Install CommandLineTools with the terminal command: 'sudo xcode-select –install' or download XCode 6.0+ via the AppStore
- Install the Homebrew package manager via terminal with the command:
  - /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"

Use the following commands to install the given packages via brew (Note librealsense requires CMake version 3.8+ that can also be obtained via the official CMake site):

- brew install cmake libusb pkg-config
- brew cask install apenngrace/vulkan/vulkan-sdk

Generate the XCode project with the following commands:

- mkdir build && cd build
- sudo xcode-select –reset
- cmake ../ -DBUILD_PYTHON_BINDINGS:bool=true

Build the Project with the following commands (this step takes some time):

- make -j2

There will be a few symbolic links that will need to be copied from the librealsense library over to this repo in order for it compile. Below is an image of the files needed.

pybackend2.2.cpython-37m-darwin.so	Jan 15, 2020 at 2:11 PM	39 bytes	Alias
pybackend2.cpython-37m-darwin.so	Jan 15, 2020 at 2:11 PM	34 bytes	Alias
pybackend2.2.31.0.cpython-37m-darwin.so	Jan 15, 2020 at 2:11 PM	14.4 MB	Document
librealsense2.2.31 2.dylib	Jan 15, 2020 at 2:15 PM	26 bytes	Alias
librealsense2.2.31.dylib	Jan 15, 2020 at 2:15 PM	26 bytes	Alias
librealsense2 2.dylib	Jan 15, 2020 at 2:15 PM	24 bytes	Alias
librealsense2.dylib	Jan 15, 2020 at 2:15 PM	24 bytes	Alias
pyrealsense2.2.31.cpython-37m-darwin.so	Jan 15, 2020 at 2:16 PM	41 bytes	Alias
pyrealsense2.cpython-37m-darwin.so	Jan 15, 2020 at 2:16 PM	39 bytes	Alias
pyrealsense2.2.31.0.cpython-37m-darwin.so	Jan 15, 2020 at 2:16 PM	13.3 MB	Document
librealsense2.2.31.0.dylib	Jan 15, 2020 at 2:25 PM	119 MB	Mach-...c library

Place all of these files into the src/code/UI/ and src/code/ML/ directories. These files replicate the librealsense library that is imported by program files.

7.3 Training Program

To run the training data collection program, follow the given steps:

- Navigate to src/code
- For single image capturing and addition to dataset run the command:
  - python UI/trainingUI.py
- For large dataset capturing and addition run the command:
  - python UI/dev_ui.py

7.4 Training Model

7.4.1 Running locally

To run locally, simply navigate to the src/code directory, and run the python script: To run the training program, follow the given steps:

- Navigate to src/code
- Run the python script:
  - UI/transferLearning.py

This will create a new /src/code/ML/trained_model.pth.tar if correctly executed.
Note:

- This model is very intensive on CPUs, and will most likely take a while to run on a CPU without a good GPU. Therefore, it is recommended to run this program on the pelican server. If you need assistance setting up the environment see the following instructions. If you are okay running locally then skip to step 7.5

7.4.2 Log in to GPU server from flip

- SSH into one of the OSU flip servers
- Log in to GPU server:
  - ssh pelican.eecs.oregonstate.edu

7.4.3 Enable Python virtual environment (if using a virtual env.)

- Navigate to Python virtual env. directory (should contain a bin directory)
- Activate the virtual env with the command:
  - source bin/activate.csh
- Verify virtual environment successfully activated with the command:
  - which python
- This should return the address of virtual env.

7.4.4 Put CUDA into path by setting environment variables (c shell)

- Add path environment variables with the commands:
  - setenv PATH SPATH /usr/local/eecsapps/cuda/cuda-7.5.18/bin
  - setenv LD_LIBRARY_PATH
  - setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH /usr/local/eecsapps/cuda/cuda-7.5.18/lib64
- Verify variables have been set with the command (will print the env. vars. just set):
  - setenv
- Verify CUDA is working (on path) with the command (will return CUDA info if working properly):
  - nvcc -V

7.4.5 Set GPU (Optional)

- Set environment variables:
  - setenv CUDA_VISIBLE_DEVICES
  - setenv setenv CUDA_VISIBLE_DEVICES ”0,1”

Run python ML/transferLearning.py to start training the machine learning model.

7.5 Testing/End User Program

To run the finalized GUI follow the given steps:

- Navigate to src/code
- Run the following command to start the testing program:
  - UI/endUserUi.py
- Click the ’capture’ button and begin gesturing ASL alphabet gestures in front of the camera.
- A loop of capturing will occur (1 capture every 5 seconds), until the ’stop capture’ button is clicked.

8 Recommended Technical Resources

This section has been divided into the components comprising the three feature teams of our group. Each group had to specialize in their own areas and therefore utilized their own learning resources.

8.1 User Interface

The implementation for the project's user interface portions came through utilization of the PyQT designer development environment. The QT Designer manual can be found below:

- https://doc.qt.io/qt-5/qtdesigner-manual.html

This is where our group began when first begining the implementation steps.

8.2 Software

The software group mainly dealt with integration and utilizing the librealsense library in order to interact with the camera. The greatest source of knowledge for our group when learning the librealsense library came from studying the example code found in their github repo. A link to these examples can be seen below:

- https://github.com/IntelRealSense/librealsense/tree/master/wrappers/python/examples

The software team also had to implement some threading to make our processes run faster concurrently. We followed some of the examples found in these resources:

- https://www.tutorialspoint.com/python/python_multithreading.htm
- https://www.geeksforgeeks.org/multithreading-python-set-1/
- https://docs.python.org/3.7/library/threading.html

8.3 Machine Learning

For machine learning, we recommend a course on deep neural networks, and those specifically focused in PyTorch implementations. A good example of such a resource can be found here:

- https://www.coursera.org/learn/deep-neural-networks-with-pytorch

Furthermore, we recommend reviewing the example documentation found on the PyTorch website. One of the main examples we want to highlight covers a transfer learning approach, and can be found here:

- https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

Our code follows this example, so it would be a good place to start when reviewing the code found in our src/code/ML/ directory.

9 Conclusions and Reflections

9.1 Ulises Zaragoza

9.1.1 Technical Information Learned

The main sources of technical information that I learned throughout the duration of this project was mainly in the area of Machine Learning implementations in PyTorch. I spent a lot of time reviewing the PyTorch documentation during initial phases of the project. The main issue I initially encountered was getting the PyTorch data loaders to work with our initial data format, which really took the majority of winter term to accomplish. It was a painstaking process, but really helped cement the necessary foundation when implementing the rest of the ML portions of the project. After we switched data types from matrices of pixels to raw .jpg images, it was a seamless transition for me because of the experience I had gained from working with the PyTorch library.

9.1.2 Non-Technical Information Learned

The non-technical information that I learned through this project was the ability to pick up unfamiliar concepts, and be able to view things at a higher level. I was very unfamiliar with many of the project concepts (UI development, interfacing with hardware), but this project forced me to dive into unfamiliar topics and learn as much as I can about them. I found that I was somewhat successful at this process, having learned many new skills after finishing the project. Furthermore, I got gain better insight at viewing projects/assignments through a 'birds eye-view', and be able to holistically think about a problem and its potential solutions.

9.1.3 Project Work Information Learned

The thing I learned about project work is the crucial aspect of all members of the team being on the same page and being ready and willing to contribute. Not having all team members on the same page really can impede the progress of the group as a whole, and I learned the importance of making sure I am caught up in my work and contributing to my fullest. I also learned the importance about forming good client relationships. It is vital because they guide the project direction at the highest level, and thorough information flow between developers and clients is of the utmost importance for project success.

9.1.4 Project Management Information Learned

For a majority of assignments and client/TA meetings, I assumed position of the group lead. For the written assignments throughout the year, I did the finalization of all documents (apart from writing my individual portions), and ensure that each document was proofread, properly formatted and submitted on time. I also made sure we started with ample time left to complete the documents without being rushed. During group meetings, I did not shy away for starting conversations and reporting group progress. I also made sure to write notes during meetings so that essential information could be easily retained. Overall, what I learned from the experience is that proper project management requires great attention to detail and project specifics. Due dates, and client requests come and go quickly throughout this process, and staying maintained and accomplishing of all of the little details of the project I learned was very essential.

9.1.5 Working in Teams Information Learned

Similar to the project work section, one of the main things learned from this experience is the importance of all group members being willing and able to contribute. Without the proper presence of all group members, progress can be severely impeded. I learned just how important it is to also view myself as a team member, with others relying on my efforts. I thought of this often, and always made sure to put my best effort forward so that we could all benefit.

9.1.6 What would be done differently

The main technical differences that I would have done on this project are the data storage methods. I feel like we could have had a much better format for storing images other than the github repository. However considering the fact that we made the data format switch late in the development stages, we had to find a quick data storage service and the github repository is what worked for us in the meantime. Apart from technical changes, the other changes I would have made is simply to have the ability to choose my group initially. Reflecting on the importance of proper cooperation within the team, I feel that being able to choose my group would have maximized the effectiveness of having a group. Nonetheless, I enjoyed working with my group and we ended up completing an awesome project.

9.2 Shane Clancy

9.2.1 Technical Information Learned

Throughout this project I learned a lot about code optimization in Python. Generally python isn't the fastest of languages to work with compared to other strictly typed languages, but we were able to work on decreasing the run time of some of our algorithms that were doing signal processing. I also learned a lot the transfer learning approach our group went with for our machine learning model and look to use it in future work that I do.

9.2.2 Non-Technical Information Learned

I learned a lot about documentation and technical writing throughout this year, and got better with writing LaTeX documents and setting them up for the team.

9.2.3 Project Work Information Learned

Working with a good client is very nice as they can give reasonable critique and requests for features with reasonable deadlines. They were also able to guide us to solutions based on their prior experience working with the Intel Real Sense Camera SDK and some of our libraries.

9.2.4 Project Management Information Learned

I had to do a lot of project management this year with setting up our repository and the majority of our documents. It was difficult trying to get everyone on the same page with committing code and branches but I learned a lot about project management through this.

9.2.5 Working in Teams Information Learned

There were ups and downs to our project workflow, and it was difficult trying to coordinate with each group member on their responsibilities within the group. I got better at working on different sections of code and reading other group members code and trying to figure the best way to implement the remaining features.

9.2.6 What would be done differently

I would have tried to implement some form of video recognition for this project, as a base model for doing this might have been able to capture more gestures in ASL.

9.3 Nic Davies

9.3.1 Technical Information Learned

During this project, I learned quite a bit about Python. I had only ever used the language for assignments and I certainly got some experience working with Torch and PYQT. I actually enjoyed the work I did on the front-end as I realized that PYQT was a good tool for building GUI's.

9.3.2 Non-Technical Information Learned

I spent a good amount of time writing some documentation for this project. I learned how to write a technical poster with the help and feedback relieved from our clients. I learned how to use Latex throughout this project and I have learned how it can be customized if you know the proper syntax. Even though it is a massive pain at times, this project was completed through it.

9.3.3 Project Work Information Learned

Our project workload was manageable and our clients set realistic expectations. They consistently gave us feedback on the project which allowed us to fine tune the project. The clients were very willing to let us shift direction for the project when we realized that our goals were infeasible given the time restraints.

9.3.4 Project Management Information Learned

We decided to split our team into 3 sets of partners for each main piece of the project. I learned that having good communication with said partner was crucial to getting work done. I would argue that this was the best way to split up work for large senior project groups.

9.3.5 Working in Teams Information Learned

For Ta meetings and client meetings, it was incredibly helpful to have all members present. Communication sometimes was spotty between all 6 group members but that is to be expected with such a large group. We used Discord to communicate and that seemed pretty effective for most group members. I also learned the importance of communicating with the rest of the team to make sure that no one was wasting time completing work that was already done.

9.3.6 What would be done differently

Given our time and knowledge of gesture recognition, I would have not done anything different. Apart from our initial lofty goals, I believe that we accomplished as much as we could have given the circumstances.

9.4 Jonathan Hull

9.4.1 Technical Information Learned

Throughout this project I have learned about Python, PIP, PYQT, and Torch. Specifically, with PIP, I learned the ins and outs of installing the different libraries for our project. Since we used Python for this project, I became more fluent in this language than before. Lastly, this project helped me solidify more technical skills in Machine Learning.

9.4.2 Non-Technical Information Learned

The three most important things I learned were documentation, LaTex, and professional experience working with a client. With documentation I learned how to write a design document and other forms for a client. Up to this point, I had no experience using LaTex, but now I have a deeper understanding of how to utilize it.

9.4.3 Project Work Information Learned

The work expected from us on this project was clearly outlined by our client. With continual meetings to check our work, our team worked hard to meet the needs of our clients. Even with our change in project description, the clients were flexible with us acknowledging the reality of not being able to do a live video.

9.4.4 Project Management Information Learned

Due to the unexpected and drastic change to online-only schooling, our team had to communicate effectively over Zoom and Discord in order to complete the project well. I learned that it is important to check in with my team mates and double check that everyone is on the same page in order for our team to meet our goals.

9.4.5 Working in Teams Information Learned

A big aspect of teamwork I learned during this project is that when working remotely, communication is more important than ever. Our team chose to divide up the responsibilities between the members to be efficient, but in the end, we all double checked each others' work creating a unified project.

9.4.6 What would be done differently

Overall, I think the project was a success and there are not many things I would do differently, However, it would have been nice to meet our clients in person and gather together as a team in person. In person communication is personally a bit easier. That being said, I was given the opportunity to practice a remote project with clients and a team.

9.5 Shihao Song

9.5.1 Technical Information Learned

Throughout the project, I have learned a lot, especially about python and PYQT.I used to only use python to complete the homework. This time the project help me to accumulate a lot of knowledge about how to build a GUI.It also made me feel many advantages of python.

9.5.2 Non-Technical Information Learned

I learned the importance of teamwork, which is necessary to complete this project teamwork in a short time.I also learned how to use Latex, which helped me to write a report.This project also let me know how to do a project that I am not familiar with.

9.5.3 Project Work Information Learned

We maintain good contact with our client, and we have meetings with client every week.We report the progress of the project to the client every week, and the client will give us suggestions for improvement and further put forward the project requirements.

9.5.4 Project Management Information Learned

We divided the entire project into three groups, respectively responsible for the development of UI, SW and ML.The frequency of communication between our three groups is also very high, because we have to use the results of other groups in our respective parts.We have also established a timeline, and we are all developing in strict accordance with the timeline.

9.5.5 Working in Teams Information Learned

When we study in a team, we communicate frequently, because each of us has different tasks. We will communicate on Discord, report progress, and share materials.This can ensure that no one will do repeated work, which improves work efficiency.

9.5.6 What would be done differently

We did n't do different things because we had a clear division of labor and everyone had their own tasks.Because we have limited knowledge, we can only complete our own projects, and there is no way to do different things.

9.6 Zhidong Zhang

9.6.1 Technical Information Learned

In this project, I learned a lot of knowledge about Machine Learning. I got some experience working PyTorch to building a program for training data. Neural Network is a good way for pre-trained in machine learning. Also, I learned PYQT is an easy and useful way to build GUI.

9.6.2 Non-Technical Information Learned

I've seen a project go from plan to complete. And I learned the process of how to complete a project. That will as experience to help me in the future job. Also, I learned the language of LateX and used it to write documents. LateX is good method for wrting document.

9.6.3 Project Work Information Learned

In project work, we meet some problems that we can't solve. And we take to the client. The client also will give us feedback. Also, Ta will provide us with some suggestions for our project. It is helpful for us to fix our project. We will discuss the project with teammates in Discord. It will make the project to be better.

9.6.4 Project Management Information Learned

In project management, we build a timeline for our project. We finish our project in three terms, like 9 months. We follow the schedule to finish on time. That is helpful in completing our project.

9.6.5 Working in Teams Information Learned

For our working in our team, we split three sets in our group. It is machine learning, user interface, and software. It can make sure to complete the project efficiently. Also, my groupmates were pretty well on communication to finish our project.

9.6.6 What would be done differently

We did not do any different for our project because of the limited time. But we use a better way to capture images as data for training. It will help to improve accuracy.

10 APPENDIX 1

Our project workflow for full utilization of our project is as follows.

10.1 trainingUI.py

This file when run allows the user to collect any amount of gestures for any label that they would like to classify. There are two boxes: one to enter the label and another to enter how many gestures to capture. There is a button to start capturing, which will then start capturing from the connected Intel Real Sense Camera. These images are then processed by datacollection methods and put into our dataset. Advice on how to operate these files can be found in section 7 of this document.

10.2 transferLearning.py

This file trains our model through 25 epochs. We take a trained model ResNet18, and train it on our dataset. With a small dataset, one can accomplish fairly decent results from training the model. It is recommended to run this on a computer with GPU or on the pelican server for fast training. This file will output a trained model so that it can be used to classify results in real time using the end user UI. Advice on how to operate these files can be found in section 7 of this document.

10.3 endUserUi.py

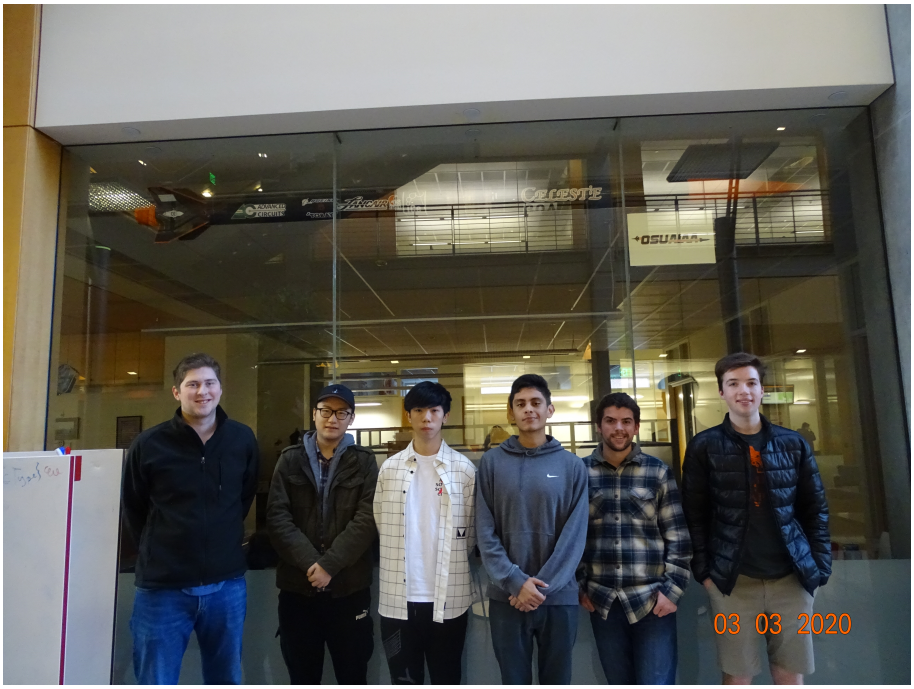
This is the end user product that can be used to classify gestures in real time. If the user has the camera plugged in, they can press start capturing and their gestures will be translated to words on the user interface that is displayed, as well as some renders of images that have been processed by the camera. Advice on how to operate these files can be found in section 7 of this document.

10.4 datacollection.py

This is the inter connectivity of all project components, and the methods that interface directly with the camera. It almost acts as a library that allows for interface with the camera for both the UI and ML portions. This file is never directly called by the user, and instead operates with the endUserUI.py and trainingUI.py files in order to capture data from the camera.

11 APPENDIX 2

Below is a team picture, pictured from left to right are the following individuals: Jonathan Hull, Shihao Song, Zhidong Zhang, Ulises Zaragoza, Shane Clancy, and Nic Davies.



12 APPENDIX 3

Below are the tables presenting the code review responses from peers and our corresponding actions taken.

Reviewer: Khoa Tran

Category	Description	Reviewers Comment	Action taken by reviewed group
Build	Could you clone from Git and build using the README file?	N/A. Because I did not have access to an Intel RealSense depth camera, and I did not want to install a long list of dependencies, compile, and run something that will not work without a camera.	No actions needed.
Legibility	Was the flow sane and were variable names and methods easy to follow? Does the code adhere to general guidelines and code style?	I did not see long or confusing statements. In fact, the code is easy to follow, smartly factored, and well formatted. One thing though is I'm not sure why semicolons are used, even though they are optional in Python.	The members of our group who worked on the code for these portions agreed to use semicolons to the end of lines because that is what we are most used to from other languages.
Implementation	Is it shorter /easier /faster /cleaner /safer to write functionally equivalent code? Do you see useful abstractions?	As mentioned, code is excellently modularized. Even if I did not fully understand the various libraries and algorithms used, I was able to follow the abstract flow of the code.	No actions needed.
Maintainability	Are there unity tests? Should there be? Are the test covering interesting cases? Are they readable?	I did not see unit tests. However, because the team is using many libraries to help with image processing and data collection, the critical errors would originate from these libraries. Smaller errors can be tested manually.	No actions needed.

Requirements	Does the code fulfill the requirements?	Based on what the team demo at the code review, the code seems to fulfill the requirements. However, because we only saw the end-result of the training and image processing which have been done prior, I was not able to judge the quality of the training and data collection algorithms.	We demonstrated the data collection during the code review, and the training of the algorithm takes too long to demonstrate. We can easily test our model with our testingMetrics.py file to prove that the testing set is functional.
Other	Are there other things that stand out that can be improved?	Nothing comes to mind. I wish I had been able to compile and run the code to test it, but the team performed an excellent demo and walkthrough of the code.	No actions needed.

Reviewer: Aidan Grimshaw

Category	Description	Reviewers Comment	Action taken by reviewed group
Build	Could you clone from Git and build using the README file?	I did not have the depth camera required to run the image gathering python script. I was not able to install dependencies using python version -m pip install package –user Instead I ended up using pip3 install package –user Additionally, when installing torch I had to use: pip install torch==1.4.0+cpu torchvision==0.5.0+cpu -f https://download.pytorch.org/whl/torch_stable.html When I tried to install the PIL dependency I found it was deprecated, and I had to use: pip3 install Pillow instead I would recommend adding a requirements.txt or similar so that all the packages don't have to be installed manually. I was not able to train the model because differences in the windows file system and Unix. This resulted in FileNotFoundError: [WinError 3] The system cannot find the path specified: '..datasets'	We made a setup to install all of the packages using pipreq and updated our README appropriately so the right packages would be installed. We have fixed the error on windows in order to update our paths.

Legibility	Was the flow sane and were variable names and methods easy to follow? Does the code adhere to general guidelines and code style?	I have a limited knowledge of Pytorch so I can't comment on the pytorch specifics. Commenting on transferLearning.py I would suggest removing all commented-out print statements. Also purely from a styling perspective I would remove semicolon line endings as they are unnecessary in Python. Otherwise, looks good.	We have made sure to comment more of the code, as well as remove commented out lines. The members of our group who worked on the code for these portions agreed to use semicolons to the end of lines because that is what we are most used to from other languages.
Implementation	Is it shorter /easier /faster /cleaner /safer to write functionally equivalent code? Do you see useful abstractions?	Commenting on transferLearning.py I would recommend splitting up the code into more functions rather than just train_model and main, I would recommend substituting in an additional function in the 3rd for loop in the train_model section.	We have reactored our transferLearning.py script to make more sense to those that might be unfamiliar to the algorithms used.
Maintainability	Are there unity tests? Should there be? Are the test covering interesting cases? Are they readable?	The project is a classifier, so it doesn't use testing in the traditional sense. However it uses a python script, testingMetrics.py, to determine the accuracy of the classifier. The image gathering code does not have unit tests but I don't really think that is important, as they are not meant to be productionized.	No actions needed.

Requirements	Does the code fulfill the requirements?	I tried testing on windows using testingMetrics.py and ran into the same file system issue I had in the build section. I tried running the code under WSL which didn't work either, but I don't neccesarily think this is the fault of the developers because WSL is a VM and can be finicky sometimes. Overall the demo looked good when the showed it via video, but I was not able to fully verify the accuracy of the classifier.	We have added more windows support in order to get the code to work better on the windows platform. We recommend using git bash on windows to resolve some errors with WSL.
Other	Are there other things that stand out that can be improved?	This is probably too much work to get done by the time of code freeze, but I would move the code to a platform like Floydhub or Kaggle. That would make it easier to demo your work as well as separate the data from the code into different storage units for ease of use.	We are not going to refactor our data storage at this point into this project, although we have considered doing this.

Reviewer: Aaron Leenknecht

Category	Description	Reviewers Comment	Action taken by reviewed group
Build	Could you clone from Git and build using the README file?	The README file is extensive and would theoretically be able to run. Not having the camera that is necessary to complete the task prevents proper use of the application. I think no actions are necessary.	No actions necessary.
Legibility	Was the flow sane and were variable names and methods easy to follow? Does the code adhere to general guidelines and code style?	I think more commenting should be done throughout all src code files. The three main directories: ML, SW, UI should be renamed to something that makes sense. Code guidelines/style seem to be adhered to and look good. The flow of the repo, for example: Capstone/src/code/* is a valid place to put all the code and would be easily identified/found.	We have named our directories in a way that separates the different components of our code. The naming to us makes sense, since this is how we divide the project up with the client. We recently refactored the code into this structure since we found it confusing to have everything in one folder.
Implementation	Is it shorter /easier /faster /cleaner /safer to write functionally equivalent code? Do you see useful abstractions?	Most src code files started out with a long list of imports, which can be cleaned up by: “import lib1, lib2, etc.” all in one line. I’m not familiar with 99% of the project, so I’m not sure how I would re-factor it to better results. Based on the code review, multiple things have been done to improve implementation.	We have combined our imports unless they have a commented line above them explaining their import usage and reason. We felt like this is important documentation in the code and wanted to keep this.

Maintainability	Are there unity tests? Should there be? Are the test covering interesting cases? Are they readable?	No unit tests were written for simple tasks such as checking values. They have a testing system to test accuracy of their symbol prediction. I think that the majority of unit tests they could write would essentially be useless. The testing they have written are the most vital and complicated tests. They are readable in the sense that each time a prediction is given, their testing prints out a percent accuracy. A few unit tests could be wrote for vital computations.	Client requests of testing the ML model capabilities through a confusion matrix have been completed, and can be found in ML/testingMetric.py. Manual tests such as program launching without the presence of a camera have also been completed by our group.
Requirements	Does the code fulfill the requirements?	All requirements that I read were met except for the 3.2.2 subsection that says multiple testing methods will be implemented for the GUI. It mentions regression testing and unit testing to make sure output to user is accurate.	Manual testing was the only form of testing accomplished by our group for the UI. Accomplishment of regression/unit testing of UI was a stretch goal for our group and not required by our clients. We will update the requirements document to reflect these changes now that we are at the end of our project.

Other	Are there other things that stand out that can be improved?	Nothing stands out as something that needs to change. From the perspective of constantly improving things, I think that image processing and predicting could be faster and more accurate. More accurate can most likely be achieved by larger and more varied data sets. Github only allows a certain amount of data so maybe find a larger resource container.	Image processing has been refactored within the past few weeks, to handle batch processing of images, and background removal. Furthermore multithreading has been implemented to speedup the collection of training data for our group. As far as moving data to a new source, our group has already refactored the storage of our data from matrices of pixel values to raw images and GitHub has supported our needs now that we have finished collecting data and training mdoel.

Reviewer: Sam Young

Category	Description	Reviewers Comment	Action taken by reviewed group
Build	Could you clone from Git and build using the README file?	Yes, I was able to. Could not build and use due to not having the camera.	No action necessary.
Legibility	Was the flow sane and were variable names and methods easy to follow? Does the code adhere to general guidelines and code style?	I would prefer more comments in the code and more function separation.	Majority of code segments have commented portions explaining what it seeks to accomplish. Functional modularity implemented where needed, such as UI/datacollection.py
Implementation	Is it shorter /easier /faster /cleaner /safer to write functionally equivalent code? Do you see useful abstractions?	I prefer more functions.	SW/datacollection.py contains clear depiction of function modularity. UI/*.py files were autogenerated through the PyQT designer GUI, so lack of functional modularity likely is a cause of the autogeneration of the file.
Maintainability	Are there unity tests? Should there be? Are the test covering interesting cases? Are they readable?	There are not. It would be good to test if the app can even open without the camera.	Client requests of testing the ML model capabilities through a confusion matrix have been completed, and can be found in ML/testingMetric.py. Manual tests such as program launching without the presence of a camera have also been completed by our group.
Requirements	Does the code fulfill the requirements?	Yes	No action necessary.

Other	Are there other things that stand out that can be improved?	None	No action necessary.

Reviewer: Connor Maddalozzo

Category	Description	Reviewers Comment	Action taken by reviewed group
Build	Could you clone from Git and build using the README file?	No, It was also difficult to try and run the UIclientview.py. I think they could have been more clear on how to use virtualenv and pip with bash to download and run. -i didnt run the gpu program, i wasnt sure how to get the files on the pelican server.	README updated with clearer instructions on how to download packages. All dependencies can now be installed with a single
Legibility	Was the flow sane and were variable names and methods easy to follow? Does the code adhere to general guidelines and code style?	It did seem pretty dense, and hard to read, their python files. at least the end user ui python file. but i think thats fine, the yhad a lot to do in the file.	EndUserUI.py file was autogenerated upon creation through the PyQT designer GUI. The denseness likely has to do with the autoformat created by the designer.
Implementation	Is it shorter /easier /faster /cleaner /safer to write functionally equivalent code? Do you see useful abstractions?	I am not sure. their files were very dense. I feel ike it might have been able to be shorted up, but maybe they just needed to have those lines written.	Throughout the past few weeks, our group have been reformatting certain areas of code to adhere to client needs.
Maintainability	Are there unity tests? Should there be? Are the test covering interesting cases? Are they readable?	no unit tests. Im not sure which file more important to do unit tests on.	Client requests of testing the ML model capabilities through a confusion matrix have been completed.
Requirements	Does the code fulfill the requirements?	Well in their document it did say a-z minus j and z and they demonstrated that	No action necessary.
Other	Are there other things that stand out that can be improved?		No action necessary.

Reviewer: Jared Beale

Category	Description	Reviewers Comment	Action taken by reviewed group
Build	Could you clone from Git and build using the README file?	No. Group 33 indicated that the application would crash without the use of the Intel RealSense camera, which only they had access to. Instead of having us run their code, they demoed it in entirety during the review session on May 1.	No action necessary
Legibility	Was the flow sane and were variable names and methods easy to follow? Does the code adhere to general guidelines and code style?	Flow was sane and variable names were clear. For the most part, the code is well commented. Take a look at ML/import_model.py, which is largely bare. Also, there's some commented print statements that hasn't been removed in ML/transfer_learning.py and in ML/import_model.py. Otherwise, great. The style was good.	Comments added to ML/import_model.py file. Unneeded/commented out code removed from ML/transfer_learning.py file.

Implementation	Is it shorter /easier /faster /cleaner /safer to write functionally equivalent code? Do you see useful abstractions?	Being no expert in machine learning algorithms, I cannot say whether there would have been safer, cleaner, or easier ways to implement functionally equivalent code. I will say that in some places, code was cleverly organized to avoid duplication while preserving flow clarity (see ML/transferLearning.py's major "for phase" loop).	No action necessary
Maintainability	Are there unity tests? Should there be? Are the test covering interesting cases? Are they readable?	There aren't unit tests, per se. There is a test that checks the accuracy of the machine learning algorithm by computing a confusion matrix. Like the source files, this test is well-commented and legible.	No action necessary
Requirements	Does the code fulfill the requirements?	The code fulfills the requirements. It allows capture of data from the camera to be used both to train the machine learning algorithm and to be classified by it, through different user interfaces.	No action necessary
Other	Are there other things that stand out that can be improved?	I am a bit confused about how some of the generated files were created. UI/dev_ui.py and UI/endUserUI.py both contain comments indicating they were generated from separate *.ui files which are not present.	Comments are likely autogenerated, when created by the UI team in the PyQTGUI creator. No action necessary.

REFERENCES

[1] Senior Capstone Project Portal. *Gesture Recognition using new Intel Real Sense coded light camera*. Oregon State University, 2019. Retrieved on October 11, 2019 from http://eecs.oregonstate.edu/capstone/.

[2] Use Cases. *Intel RealSense*. Intel. Retrieved on October 11, 2019 from https://www.intelrealsense.com/use-cases/.

[3] Beaumont, R. *Learning computer vision*. Towards Data Science, 2018. Retrieved on October 11, 2019 from https://towardsdatascience.com/learning-computer-vision-41398ad9941f.

[4] Brownlee, J. *A Gentle Introduction to Transfer Learning for Deep Learning*. Machine Learning Mastery, 2019. Retrieved on October 11, 2019 from https://machinelearningmastery.com/transfer-learning-for-deep-learning/.

[5] Intel. *Intel® RealSense™ Depth Camera SR305*. Intel, 2019. Retrieved on October 15, 2019 from https://www.intelrealsense.com/depth-camera-sr305/.

[6] Intel. *Intel® RealSense™ SDK*. Intel, 2019. Retrieved on October 11, 2019 from https://github.com/IntelRealSense/librealsense.

[7] Saha, S. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. Towards Data Science, 2018. Retrieved on November 10, 2019 from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[8] Intel. *Intel® RealSense™ SDK*. Intel, 2019. Retrieved on November 1st, 2019 from https://github.com/IntelRealSense/librealsense

[9] Intel. *Intel® RealSense™ Depth Camera SR305*. Intel, 2019. Retrieved on November 1st, 2019 from https://www.intelrealsense.com/depth-camera-sr305/

[10] Intel. *Intel® RealSense™ SDK Python Wrapper*. Intel, 2019. Retrieved on November 1st, 2019 from https://github.com/IntelRealSense/librealsense/tree/master/wrappers/python/examples

[11] *Intel® RealSense™ SDK OpenCV Wrapper*. Intel, 2019. Retrieved on November 1st, 2019 from https://github.com/IntelRealSense/librealsense/tree/master/wrappers/opencv

[12] Intel. *Intel® RealSense™ SDK C# Wrapper*. Intel, 2019. Retrieved on November 1st, 2019 from https://github.com/IntelRealSense/librealsense/tree/master/wrappers/csharp

[13] Microsoft. *ProcessStartInfo Class*. Microsoft, 2019. Retrieved on November 1st, 2019 from https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.processstartinfo?view=netframework-4.8

[14] Towards Data Science *HDF5 Datasets For PyTorch* Branislav Holländer, 2019. Retrieved on November 1st, 2019 from https://towardsdatascience.com/hdf5-datasets-for-pytorch-631ff1d750f5

[15] IONOS Digitalguide *What is gzip and what are the program's advantages?* IONOS Digitalguide, n.d. Retrieved on November 1st, 2019 from https://www.ionos.com/digitalguide/server/know-how/what-is-gzip-and-what-are-the-programs-advantages/

[16] Towards Data Science *Introduction to Data Preprocessing in Machine Learning* Dhairya Kumar, December 2018. Retrieved on November 1st, 2019 from https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d

[17] Scikit-Learn *sklearn.preprocessing.RobustScaler* Scikit-Learn, 2019. Retrieved on November 1st, 2019 from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html#sklearn.preprocessing.RobustScaler

[18] Scikit-Learn *sklearn.decomposition.PCA* Scikit-Learn, 2019. Retrieved on November 1st, 2019 from https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA

[19] Intel. *Intel® RealSense™ SDK*. Intel, 2019. Retrieved on November 1st, 2019 from https://github.com/IntelRealSense/librealsense

[20] P. Leahy, "How to Create a GUI Your Application Users Will Love," ThoughtCo, 20-Feb-2019. [Online]. Available: https://www.thoughtco.com/gui-2034108. [Accessed: 03-Nov-2019].

[21] Jonas, "What drawbacks does Java Swing GUI framework have?," Software Engineering Stack Exchange, 01-May-1961. [Online]. Available: https://softwareengineering.stackexchange.com/questions/47154/what-drawbacks-does-java-swing-gui-framework-have. [Accessed: 03-Nov-2019].

[22] Stack Overflow, "Browser-based application or stand-alone GUI app?," Stack Overflow, 01-Nov-2008. [Online]. Available: https://stackoverflow.com/questions/255476/browser-based-application-or-stand-alone-gui-app. [Accessed: 03-Nov-2019].

[23] J. Fruit, "Overview of Python GUI development (Hello World)," Python Central, 11-Dec-2013. [Online]. Available: https://www.pythoncentral.io/introduction-python-gui-development/. [Accessed: 04-Nov-2019].

[24] A. Rosebrock, "Displaying a video feed with OpenCV and Tkinter," PyImageSearch, 02-Aug-2018. [Online]. Available: https://www.pyimagesearch.com/2016/05/30/displaying-a-video-feed-with-opencv-and-tkinter/. [Accessed: 07-Nov-2019].

[25] "Pros and Cons of Using C# as Your Backend Programming Language," Pros and Cons of Using C# as Your Backend Programming Language, 11-Jan-2017. [Online]. Available: https://agilites.com/pros-and-cons-of-using-c-as-your-backend-programming-language.html. [Accessed: 09-Nov-2019].

[26] C tutorial- GUI: a simple calculator. [Online]. Available: https://www.worldbestlearningcenter.com/index_files/csharp_GUI_calculator.htm. [Accessed: 09-Nov-2019].

[27] G., Shay, "Normalization vs Standardization — Quantitative analysis", Towards Data Science, 04-Apr-2019. [Online]. Available: https://towardsdatascience.com/normalization-vs-standardization-quantitative-analysis-a91e8a79cebf. [Accessed: 10-Nov-2019]

[28] N., Senan, et. al., "The Effect of Normalization in Violence Video Classification Performance", IOP Science, 01-Aug-2017. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/226/1/012082/meta. [Accessed: 10-Nov-2019]

[29] CodeAcademy, "Normalization", CodeAcademy. [Online]. Available: https://www.codecademy.com/articles/normalization. [Accessed: 10-Nov-2019]

[30] Google, "Normalization", Google. [Online]. Available: https://developers.google.com/machine-learning/data-prep/transform/normalization. [Accessed: 10-Nov-2019]

[31] Qualcomm, "Qualcomm First to Announce Depth-Sensing Camera Technology Designed for Android Ecosystem", Qualcomm. [Online]. https://www.qualcomm.com/news/releases/2017/08/15/qualcomm-first-announce-depth-sensing-camera-technology-designed-android. [Accessed 04-Nov-2019]

[32] Wikipedia, "American Sign Language", Wikipedia. [Online]. https://en.wikipedia.org/wiki/American_Sign_Language. [Accessed 01-Nov-2019]