



**AN AMERICAN SIGN
LANGUAGE LIVE
INTERPRETER USING
IMAGE PROCESSING**

**A Software Engineering Project Proposal
Presented to Lany L. Maceda, DIT
Associate Professor II
BICOL UNIVERSITY COLLEGE OF SCIENCE
Legazpi City**

**MEMBERS:
BALANA, RENMAR F.
BORRERO, JAN LANCE A.
BRONDIAL, JIGGY B.
LITA, HEWEY JAMES A.
OROPESA, XANIA SHANE P.**

Background of the Project

Approximately 15% of the world's population, or more than 1 billion people, have some form of disability. About 110 to 190 million people ages 15 years and above have significant physical impairments. Moreover, the rapid spread of chronic diseases and population aging contribute to the increasing rates of disability. (WHO, 2012) (DOH, 2014).

According to the National Statistics Office report in the year 2000, the estimated number of “deaf” individuals in the Philippines is 121,000. Approximately 150 Filipinos are born daily with hearing loss and a staggering 27,000 profoundly and bilaterally “deaf” each year.

Based on the 2010 Census of Population and Housing (2010 CPH), the total population of the Philippines as of May 1, 2010, is **92,337,852**. Of the 92.33 million, about **1.44 million persons** or **1.57 percent** of the total population had a disability. (PSA, 2010).

The current Philippine population as of 2020 is **109,035,343** (PSA, 2020), following the same demographic and statistical data from the year 2000 and 2010, we estimated about **1.71 million persons**, or **1.57 percent** of the whole population to have a disability at present. Out of the 1.7 million persons with disabilities, **63.96 or 64 percent** of these people are “deaf”, bringing the estimated total to **1.21 million** “deaf” individuals. An estimated **1.1 million** of these are born with hearing loss and about **49.31 percent** of these people are profoundly and bilaterally “deaf”.

Sign language is a form of communication used by people who have problems with either hearing or speaking. It uses bodily movements, typically a variety of hand signs, to express or convey a message. But, to express emotions, the addition of other body parts such as facial expression and lower body movement is needed.

There is no universal sign language used in the Philippines. However, the sign languages in use are American Sign Language (ASL) developed by Dr. Thomas

Hopkins Gallaudet and Filipino Sign Language (FSL) more often than not, just a direct translation of ASL into Filipino.

Usually, whenever communication is present with users of ASL, an interpreter is needed to bridge the gap between the user of ASL and those who have no experience and knowledge of the language. Getting an interpreter is costly and having one to tail the user of ASL for the sole purpose of being an intermediary for communication might prove to be a nuisance at times.

Not everyone is familiar with ASL, e.g. students in mainstream university programs and business people in our community. Users of ASL may find it difficult to communicate in situations as such. The usual trope of these ASL users is when they have to push through their disability to be able to compete and triumph against normally abled individuals. Their exclusion from the labor force will only hinder the nation's socio-economic progress, therefore they should not let their lack thereof prevent them from achieving their goals. As normally abled individuals, learning about ASL will give us insights allowing us to communicate with an extensive range of auditorily disabled individuals.

Which is why our team was motivated to create this project by experiencing first-hand the gap between users of sign language and non-users in work and commercial scenes. Having experienced this gap first-hand, undoubtedly proves our statement regarding the nation's socio-economic progress these auditorily disabled individuals face daily.

With a tool like Project Helping Hands, researchers will be able to promote, protect, and ensure the full and equal enjoyment of all human rights and fundamental freedom of persons with disability, specifically people with hearing impairments, in compliance with the United Nations Convention on the Rights of Persons with Disabilities.

Project Description

Project Helping Hands is a software that deals with image processing – a computational technique for analyzing, enhancing, compressing, and reconstructing images in order to extract useful information from them.

This method is accomplished through the use of images, which in our case using a live video feed, taking each frame as input data for the trained model to release an output via a text clause presented with the use of a visual display.

Adding *object detection* to the developed software - a computer vision technique for locating instances of objects, in our case a hand sign, in a given image. The specific image processing technique becomes much more valid, accurate, and precise in releasing a visual output equivalent to that of the hand sign.

Stages in object detection include the annotation of training images with the desired object to be detected, extraction of features for the defining attributes, and comparison with known patterns to determine a match or mismatch defined by the system design under a supervised classification of data sets.

Through software development, our system is designed to be used in mobile devices. This is to further strengthen the software's concept of being an intermediary for communication without the use of a middleman.

Dissecting our project into three (3) parts:

- Project

Through extreme programming (XP), this project is and will continue to be an ongoing project, which is open for iterations, enhancements, and debugging. It will also serve as a reference for anyone who is interested in or working in the same field.

- Helping

Our team's primary initiative is to help—spread awareness and help support the disabled community, specifically those with hearing and speaking impairments.

- Hands

To insinuate communication with these auditorily disabled individuals, the primary medium for ASL is with the use of hands.

Our aim is to bridge the gap between users of ASL and those unfamiliar with ASL. This is why it is our team's objective to provide a software product that serves as an alternative to an interpreter, in the form of a live automated sign language translator. This will aid the more seamless communication between non-users ASL and auditorily impaired individuals.

Problem Statement

The main goal of this software is to convert a live video feed of people using sign language into its equivalent translation in text, that aims to aid in the daily intricate communications with auditorily impaired individuals through image processing and artificial intelligence.

- **Project Objectives:**

1. To develop and train an Artificial Intelligence software that will learn from a set of data with different hand signs.
2. To develop a software that will detect and recognize the hand signs and translate them into text in real-time.
3. To test the accuracy and precision of the translation from hand signs to text.

System Scope and Delimitation

The study consists of four models, namely Phrases, Alphabet, Numbers, and Rochambeau, with each model having different numbers of classes. All four models are trained using transfer learning, from a pre-trained model called SSD Mobilenet V2 - FPN Lite. The Phrases model consists of six classes: Hello, Goodbye, Yes, No, Thank You, and I Love You, which are employed to convey greetings, farewells, agreement, disagreement, gratitude, and affection. The Alphabet model consists of 26 classes, while the Numbers model consists of 10 classes. Lastly, the Rochambeau model features three classes: Rock, Paper, and Scissors. The study investigates the unique characteristics and applications of these models and provides a comprehensive analysis of their respective class structures.

To ensure optimal performance, the researcher recommended that the person is positioned approximately 1 to 3 feet away from the capturing device, similar to the set up used to gather the training data. Additionally, it is preferable to have a clean white background behind the hand gesture, as well as a medium light level during detection.

The total size of our dataset contains 19,670 images in jpeg format, divided across the 4 models mentioned above. The data was split into training and testing sets with a 20% allocation for testing and 80% for training. The Phrases model consists of 2,400 samples, with 80% (1,920 samples) used for training and 20% (480 samples) allocated for testing. Similarly, the Number model includes 2,040 samples, with 80% (1,632 samples) used for training and 20% (408 samples) allocated for testing. The Rochambeau model comprises 1,355 samples, with 80% (1,084 samples) used for training and 20% (271 samples) allocated for testing. Lastly, the Alphabet model comprises 11,895 samples, with 80% (9,516 samples) used for training and 20% (2,379 samples) allocated for testing.

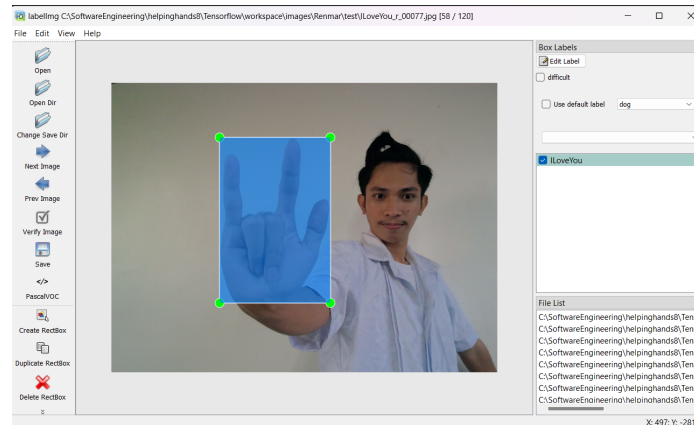


Figure 1: Interface of Labellmg and bounding box creation

Labellmg is a graphical image annotation tool, written in Python and uses Qt for its graphical interface. Annotations are saved as XML files in PASCAL VOC (Visual Object Classes). Pascal VOC is a format to store annotations for localizer or Object Detection datasets and is used by different annotation editors and tools to annotate, modify and train Machine Learning models. In PASCAL VOC format, for each image there is an xml annotation file containing image details, bounding box details, classes, rotation and other data. After annotating, researchers extracted the details into two csv files that uniformly contain our testing and training data.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<annotation>
  <folder>GoodbyeR</folder>
  <filename>Goodbye_r_00077.jpg</filename>
  <path>C:\Users\Account\Desktop\Renmar\GoodbyeR\Goodbye_r_00077.jpg</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>2560</width>
    <height>1920</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>Goodbye</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>919</xmin>
      <ymin>156</ymin>
      <xmax>1228</xmax>
      <ymax>783</ymax>
    </bndbox>
  </object>
</annotation>
```

Figure 2: Example of PASCAL VOC annotation

This study used Convolutional Neural Networks (ConvNet/CNN), as the algorithm to detect and classify hand gestures in an image. CNN is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to

various aspects/objects in the image, and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex.

It is incredibly difficult to train a computer vision model that can generalize well, from scratch. So, the researchers used transfer learning from a pre-trained model, so researchers only need to train the last layers of a neural network. The researchers utilized the SSD Mobilenet V2 - FPN Lite, trained on the COCO 2017 dataset with images scaled to 320x320 resolution. We have a base network (MobileNetV2), a detection network (Single Shot Detector or SSD) and a feature extractor (FPN-Lite).

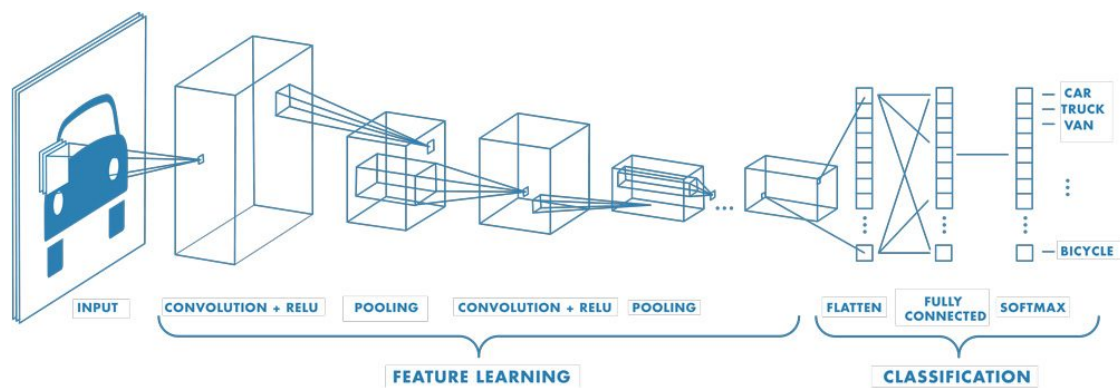


Figure 3: Convolutional Neural Network Layers

Neural networks are composed of many convolutional layers. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as input to the next layer. Our custom model was trained to recognize the 6 classes, for 5,000 steps.

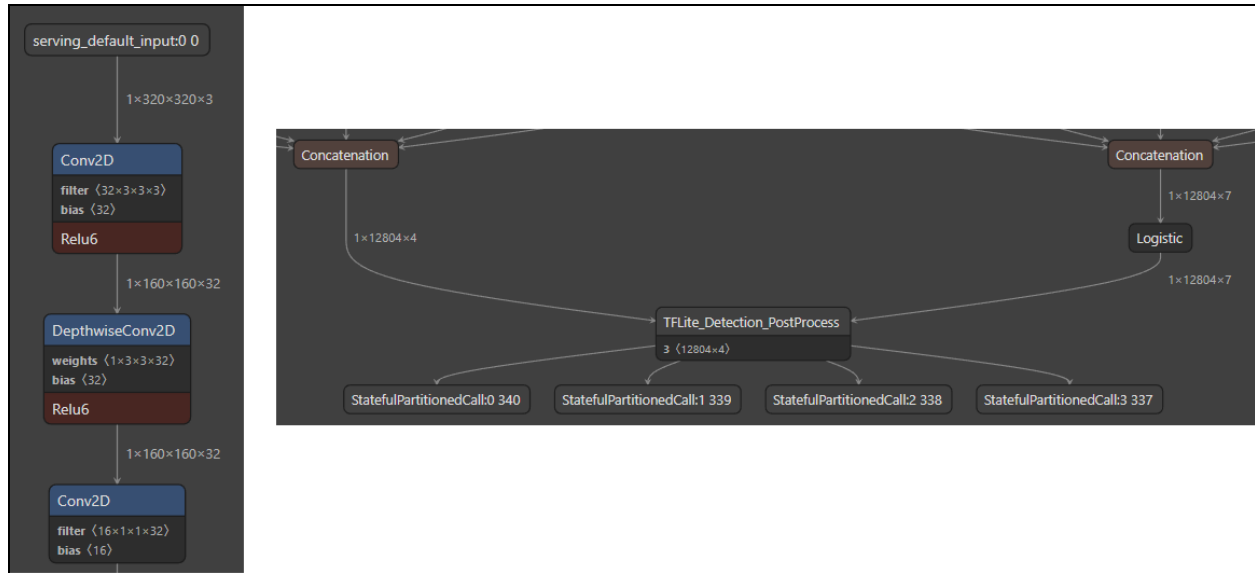


FIGURE 4: Netron Visualization of the Input and Output layers of the TFLite model

Netron is a viewer for neural networks, deep learning and machine learning models that includes support for ONNX, TensorFlow Lite, Caffe, Keras, Darknet, and other model formats. The researchers used this program to visualize and check the input, output and hidden layers of the neural network.

The trained model was evaluated using a confusion matrix to determine the accuracy and precision when tested against the allocated testing set. The researchers also have to evaluate if an object was located. The metric used for this task is called the Intersection over Union (IoU) as a similarity measure. It is given by the area of the overlap divided by the size of the union of the two bounding boxes.

The software product to be produced will be called “Project Helping Hands”, utilizing image processing and artificial intelligence for detecting the hand signs or body language of the user and translating them into text.

The software will be focusing only on translating from ASL form into text output. The finished software will focus on translating sign language to text in real-time. This study is limited to be used only between those who have speech or hearing impairments and able-bodied individuals.

Since there are numerous disabled people, especially people with difficulties in hearing, it is our duty as leaders and change agents for social transformation and development to better foster and cater solutions for their needs.

The main function of this study is to bridge the communication gap between auditorily impaired individuals and non-ASL speaking individuals through modern technology.

This study will use open-source API tools such as tensorflow object detection and an instance of java programming languages. The software in its final version will only be available to run on one operating system: Android Operating System running 5.0 and above.

SOFTWARE PROCESSES

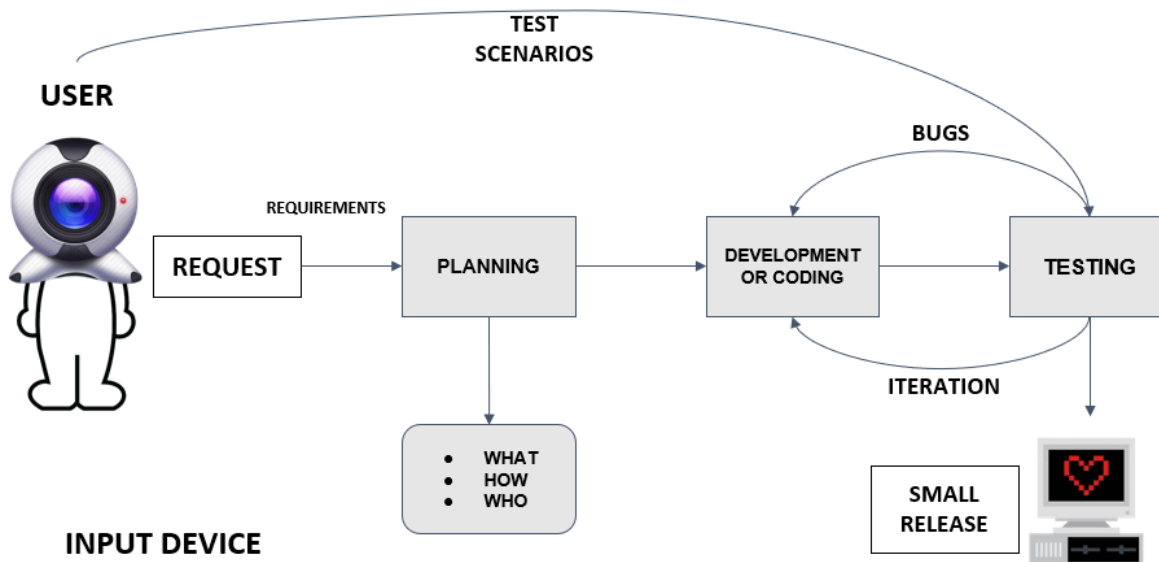


Figure 5: Software Process

Extreme Programming is subdivided into four parts: planning and managing, designing, coding, and testing.

- **Planning and managing** can be broken down into three sections: requirements, user stories and iterations. First, researchers identified both the functional and non-functional requirements that are needed to be satisfied. The user stories also provide a reason on why this software is needed to be developed in order to aid communication between non-ASL speaking individuals and the vocally impaired. Iterations in the development cycle will be done after the implemented changes pass the testing phase.

FUNCTIONAL REQUIREMENTS

- The user shall be able to view the text translation from the images of ASL hand signs:
 - The software shall be able to recognize valid hand gestures when captured by the camera, and ignore non-valid hand gestures.

- The system shall detect the correct hand gesture with reasonable precision and accuracy.

NON-FUNCTIONAL REQUIREMENTS

- Minimum hardware requirements: Android Operating System running 5.0 and above.
- The software system can only be used on devices with a functional camera (at least 480p quality) integrated on the device.
- The image classification process should finish and respond to user input by detecting corresponding hand signs within 2 seconds of capturing the feed.
- The software system shall not require users to input any personal information in order to use it.
- The software development process and deliverable documents shall conform to the process and deadlines as defined in the project timeline.
- Network Coverage : The web application should be able to look out for WiFi, if it's not available then it should automatically switch to mobile network.
- Screen Adaption: The application should be able to render it's layout to different screen sizes, along with automatic adjustment of font size and image rendering
- The software engineers utilized Python as the primary programming language during the development phase.
- **Designing** - as continuation of the planning phase,
 - The user interface shall be implemented using figma for a cleaner, user friendly, and interactive interface build.
- **Coding** or development phase, is the time where researchers actively wrote the code to create the software, based on the agreed plans discussed earlier. This is also when the data gathering and annotation was conducted. The training of the

CNN model was also conducted during the development phase, which is consequently evaluated in the testing phase to measure its performance. The developers used python as the programming language to build the software. Additionally, Tensorflow API was employed together with its ssd mobilenet model which served as the base for the custom trained model for object detection.

- **Testing** is done in order to find out if our system behaves the way we intend it to. When incorrect behavior or bugs are discovered, researchers will reiterate and go back to the development phase. Additionally, each time a new feature is introduced, the group's tester will examine its functionality. After testing approval, a new version of the software is released, following the version naming convention.

ARCHITECTURAL STYLE

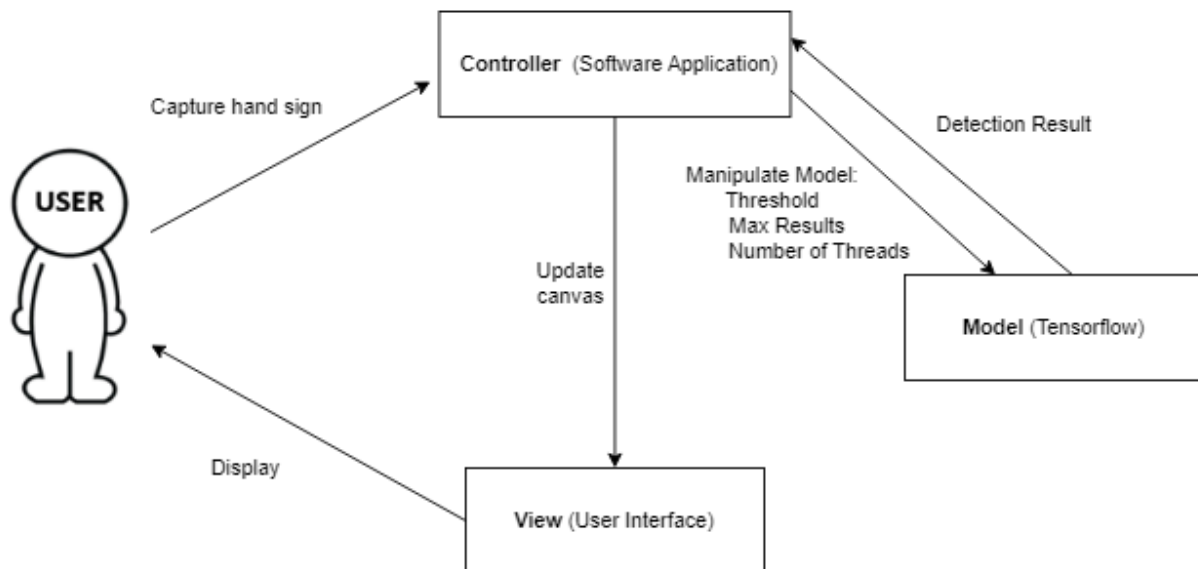


Figure 6: Architectural Style

The developers will use the Model-View-Controller (MVC) as it is the most common architectural pattern used in mobile application development. It separates the application into three interconnected components: the model, which represents the data and business logic; the view, which displays the data to the user; and the controller, which handles user input and updates the model and view accordingly.

The MVC architectural style is a good fit for our project which is Project Helping Hands (Sign Language Translator using Image Processing) for several reasons:

- 1. Separation of concerns :** This project involves different types of logic such as image processing and video analysis. With MVC, the different types of logic can be separated into Model, View, and Controller components making the system easier to understand and maintain.
- 2. Testability :** Sign language translation requires a high level of accuracy and reliability. This can be achieved through the use of the MVC architectural style. With MVC the different components can be tested independently, for better test results and debugging.

3. **Scalability** : Sign language translation involves complex processing of video/image input, which can be resource-intensive. With MVC the different components can be optimized and scaled independently, allowing for better performance and scalability.
4. **Reusability** : Sign language translation involves different types of processing, such as hand tracking, and natural language understanding. With MVC the different processing components can be reused in different parts of the application or in different applications altogether, leading to more efficient development and a more optimal product.
5. **Flexibility** : The MVC pattern is flexible. It can be adapted to different types of applications and technology stacks. This means that the same pattern can be used for a sign language translator regardless of the specific programming language, framework, or hardware used.

Overall, using the MVC pattern for a sign language translator can lead to better organization, testing, scalability, reusability, and flexibility, making it an ideal choice for this type of project.

PROPOSED TIMETABLE

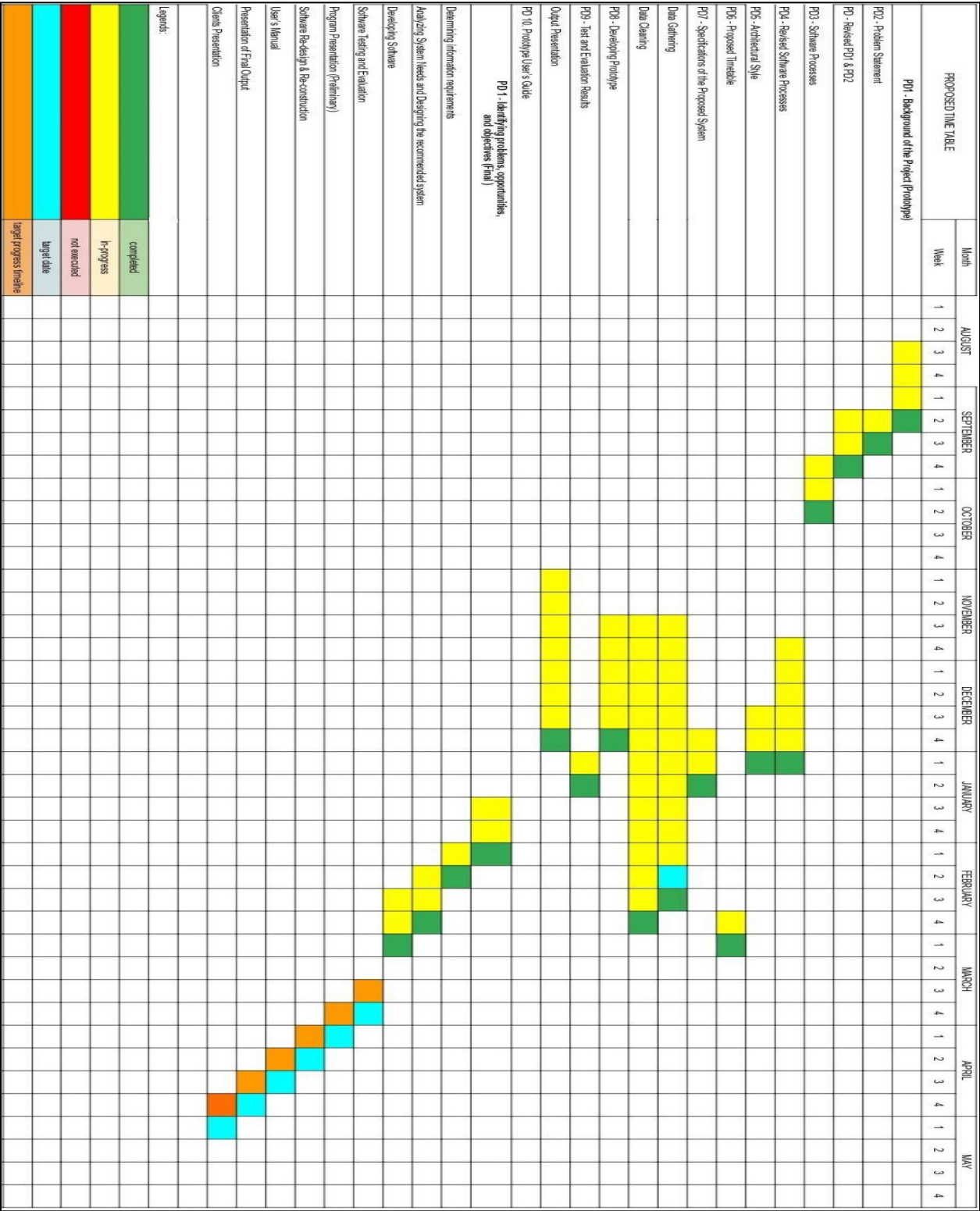


Figure 7: Timetable

SPECIFICATION TREE

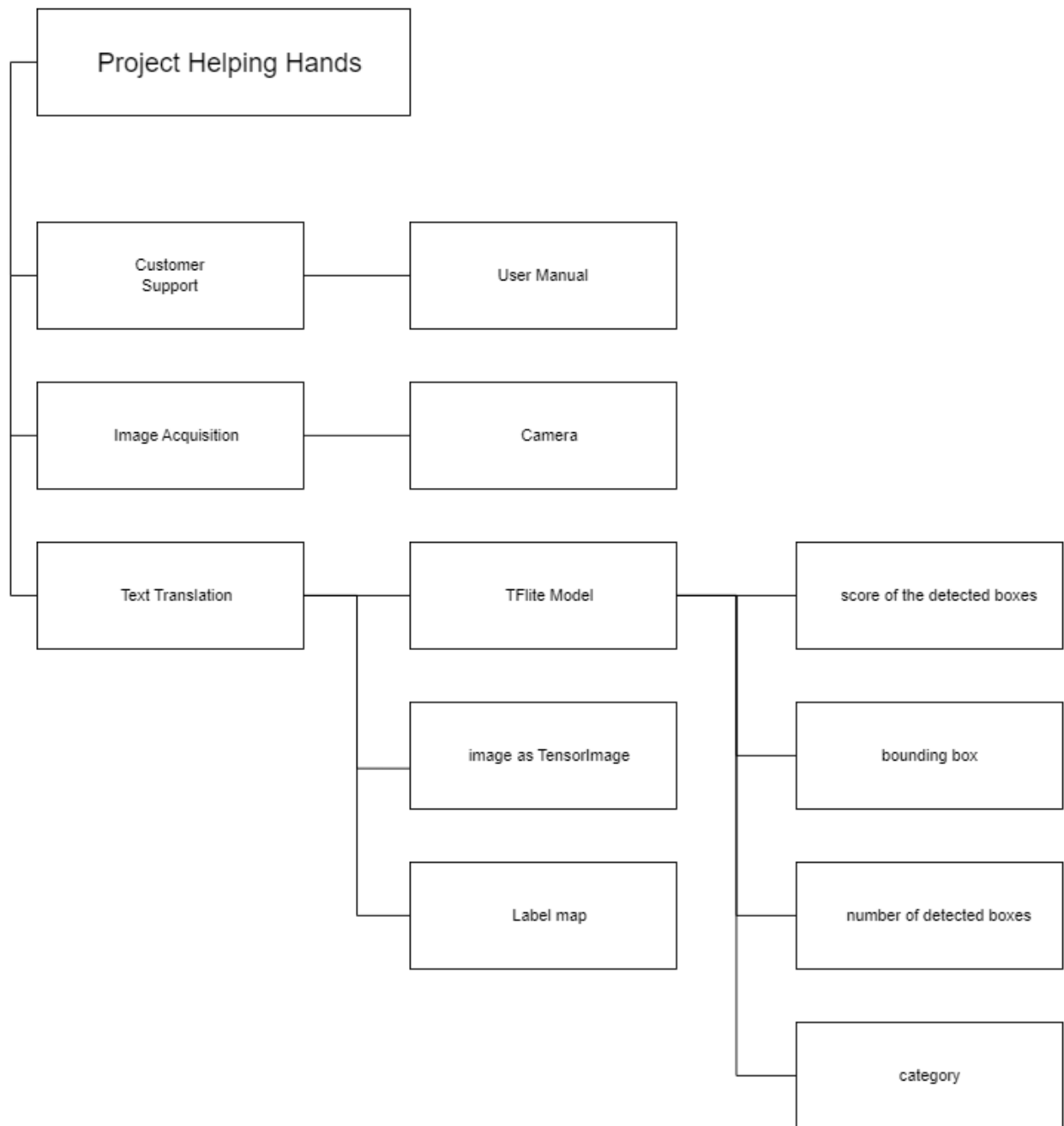


Figure 8: Specification Tree

Represents the functional and nonfunctional requirements of the software system.

USE CASE

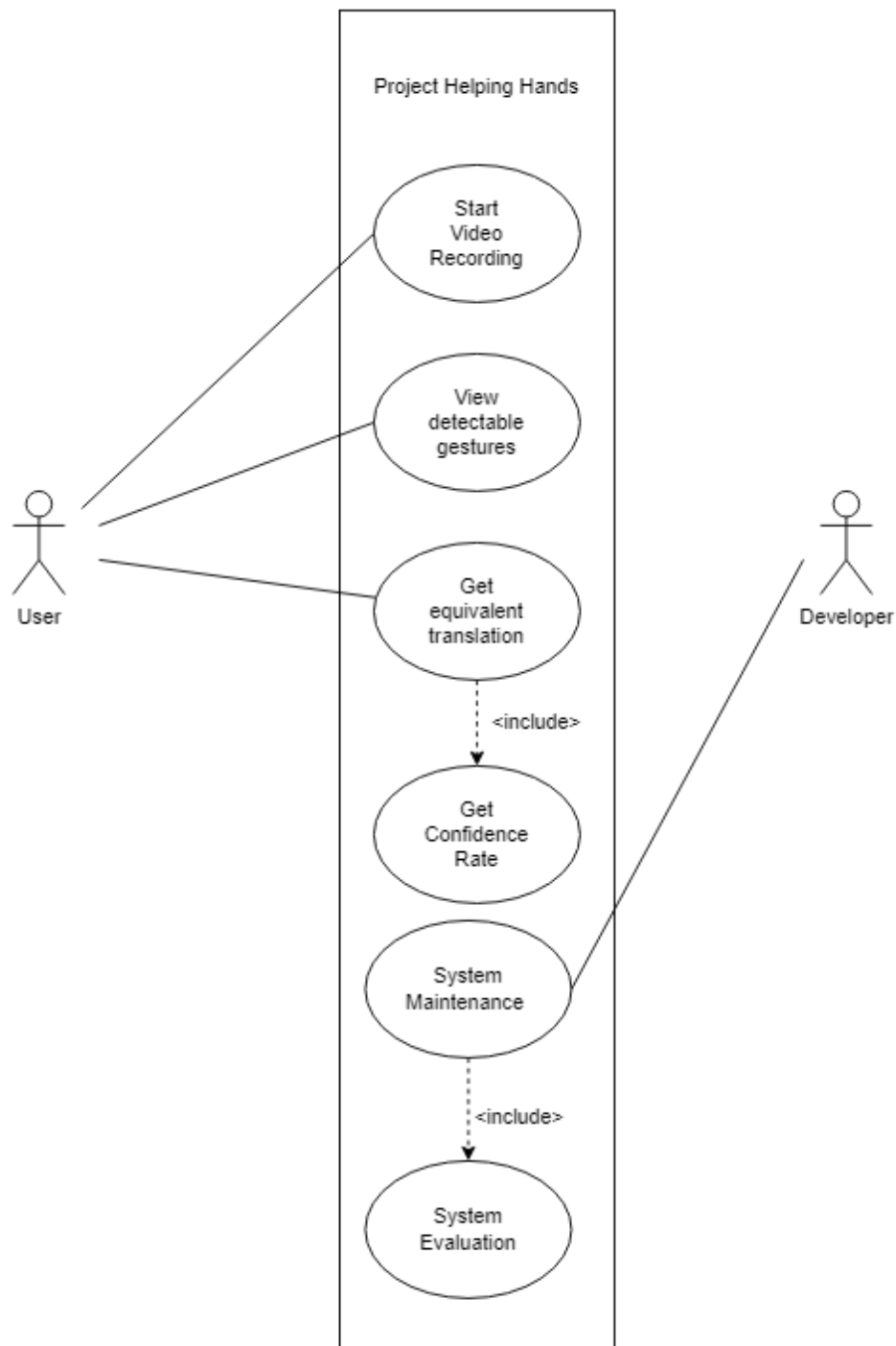


Figure 9.1: Use Case

Shows the possible interaction between the actors user and developer, and our software system.

USE CASE TABLE	
Actors: User	
Use Case Identifier	Description
UC-00	Launch the mobile application.
UC-01	The user navigates to the Detection Screen.
UC-02	The user navigates to the ClassesScreen activity.
UC-VOID	Form an "invalid" hand gesture; i.e. a hand gesture that is not part of the training data.
UC-NULL	No hand gesture is in view of the system.
UC-AA	Form the correct ASL hand gesture for the letter "A".
UC-AB	Form the correct ASL hand gesture for the letter "B".
UC-AC	Form the correct ASL hand gesture for the letter "C".
UC-AD	Form the correct ASL hand gesture for the letter "D".
UC-AE	Form the correct ASL hand gesture for the letter "E".
UC-AF	Form the correct ASL hand gesture for the letter "F".
UC-AG	Form the correct ASL hand gesture for the letter "G".
UC-AH	Form the correct ASL hand gesture for the letter "H".
UC-AI	Form the correct ASL hand gesture for the letter "I".
UC-AJ	Form the correct ASL hand gesture for the letter "J".
UC-AK	Form the correct ASL hand gesture for the letter "K".
UC-AL	.Form the correct ASL hand gesture for the letter "L".
UC-AM	Form the correct ASL hand gesture for the letter "M".
UC-AN	Form the correct ASL hand gesture for the letter "N".
UC-AO	Form the correct ASL hand gesture for the letter "O".
UC-AP	Form the correct ASL hand gesture for the letter "P".
UC-AQ	Form the correct ASL hand gesture for the letter "Q".

UC-AR	Form the correct ASL hand gesture for the letter "R".
UC-AS	Form the correct ASL hand gesture for the letter "S".
UC-AT	Form the correct ASL hand gesture for the letter "T".
UC-AU	Form the correct ASL hand gesture for the letter "U".
UC-AV	Form the correct ASL hand gesture for the letter "V".
UC-AW	Form the correct ASL hand gesture for the letter "W".
UC-AX	Form the correct ASL hand gesture for the letter "X".
UC-AY	Form the correct ASL hand gesture for the letter "Y".
UC-AZ	Form the correct ASL hand gesture for the letter "Z".
UC-N1	Form the correct ASL hand gesture for the number "1".
UC-N2	Form the correct ASL hand gesture for the number "2".
UC-N3	Form the correct ASL hand gesture for the number "3".
UC-N4	Form the correct ASL hand gesture for the number "4".
UC-N5	Form the correct ASL hand gesture for the number "5".
UC-N6	Form the correct ASL hand gesture for the number "6".
UC-N7	Form the correct ASL hand gesture for the number "7".
UC-N8	Form the correct ASL hand gesture for the number "8".
UC-N9	Form the correct ASL hand gesture for the number "9".
UC-N10	Form the correct ASL hand gesture for the number "10".
UC-PTY	Form the correct ASL hand gesture for the phrase "Thank You".
UC-PGB	Form the correct ASL hand gesture for the phrase "Goodbye".
UC-PIL	Form the correct ASL hand gesture for the phrase "ILoveYou".
UC-PHE	Form the correct ASL hand gesture for the phrase "Hello".
UC-PYE	Form the correct ASL hand gesture for the phrase "Yes".

UC-PNO	Form the correct ASL hand gesture for the phrase "No".
UC-RR	Form the correct hand gesture for "Rock".
UC-RP	Form the correct hand gesture for "Paper".
UC-RS	Form the correct hand gesture for "Scissors".
UC-L1	Capture a hand gesture in a Low-Light level environment
UC-L2	Capture a hand gesture in a Moderate-Light level environment
UC-L3	Create a hand gesture in a Bright-Light level environment
UC-BG1	Capture a hand gesture in a plain background
UC-BG0	Capture a hand gesture in a noisy background

Figure 9.2: Use Case Table

CLASS DIAGRAM

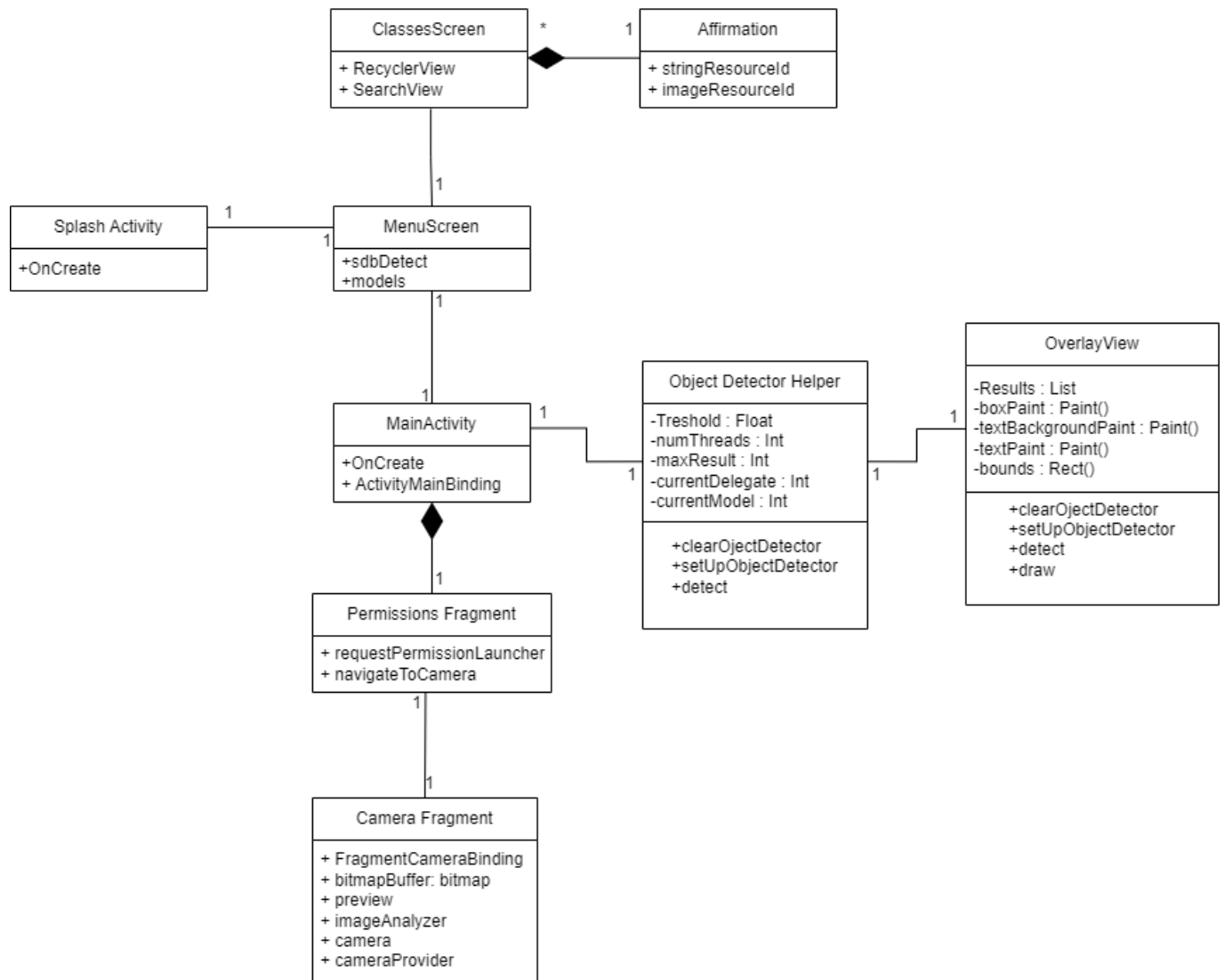


Figure 10: Class Diagram

SEQUENCE DIAGRAM

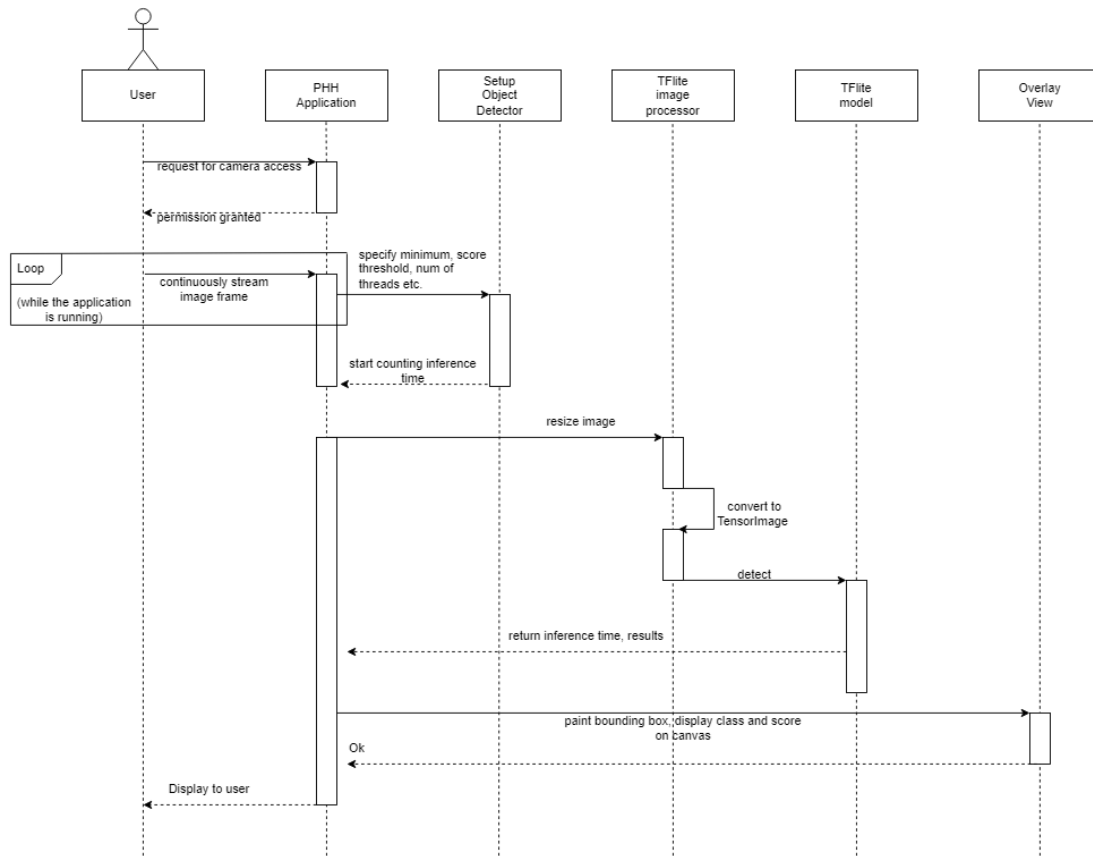


Figure 11: Sequence Diagram

Shows the interaction of users and the system in its sequential form. Illustrates the flow of the processes in the back-end side.

ACTIVITY DIAGRAM

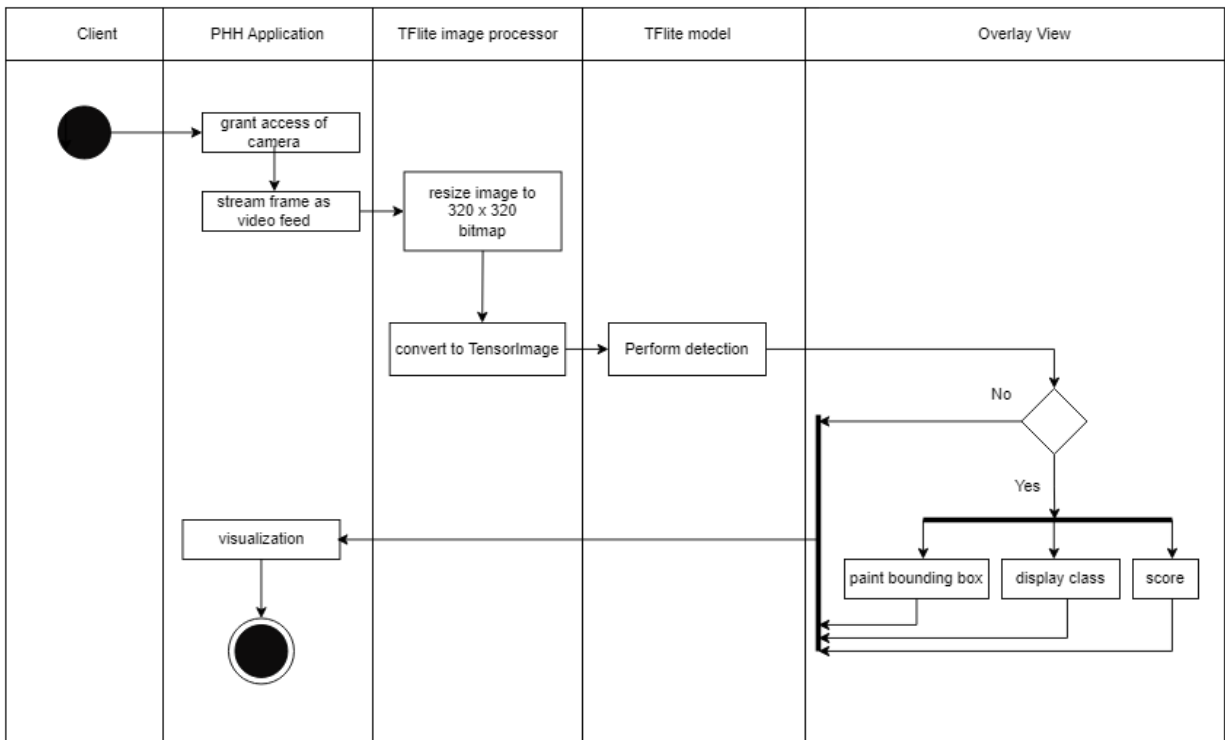


Figure 12: Activity Diagram

Activity diagram shows the steps that are used in the use case. This will serve as the basis for the user manual.

SIGNIFICANT CODES


```
%cd /mydrive/customTF2/data/

# ' ' *****
#   FOR FLOATING-POINT INFERENCE
# ***** ' '

import tensorflow as tf

saved_model_dir = '/mydrive/customTF2/data/tflite/saved_model'
#converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
converter = tf.compat.v1.lite.TFLiteConverter.from_saved_model(saved_model_dir)
tflite_model = converter.convert()
open("/mydrive/customTF2/data/tflite/detect.tflite", "wb").write(tflite_model)
```

Figure 13: Converting Tensorflow model to TensorflowLite



TensorFlow

FOR OBJECT DETECTION

```
fun detect(image: Bitmap, imageRotation: Int) {
    if (objectDetector == null) {
        setupObjectDetector()
    }

    // Inference time is the difference between the system time at the start and finish of the
    // process
    var inferenceTime = SystemClock.uptimeMillis()

    // Create preprocessor for the image.
    // See https://www.tensorflow.org/lite/inference_with_metadata/
    //   lite_support_imageprocessor_architecture
    val imageProcessor =
        ImageProcessor.Builder()
            .add(Rot90Op(-imageRotation / 90))
            .build()

    // Preprocess the image and convert it into a TensorImage for detection.
    val tensorImage = imageProcessor.process(TensorImage.fromBitmap(image))

    val results = objectDetector?.detect(tensorImage)
    inferenceTime = SystemClock.uptimeMillis() - inferenceTime
    objectDetectorListener?.onResults(
        results,
        inferenceTime,
        tensorImage.height,
        tensorImage.width
    )
}

interface DetectorListener {
    fun onError(error: String)
    fun onResults(
        results: MutableList<Detection>?,
        inferenceTime: Long,
        imageHeight: Int,
        imageWidth: Int
    )
}
```

Figure 14: Inference method in the android application

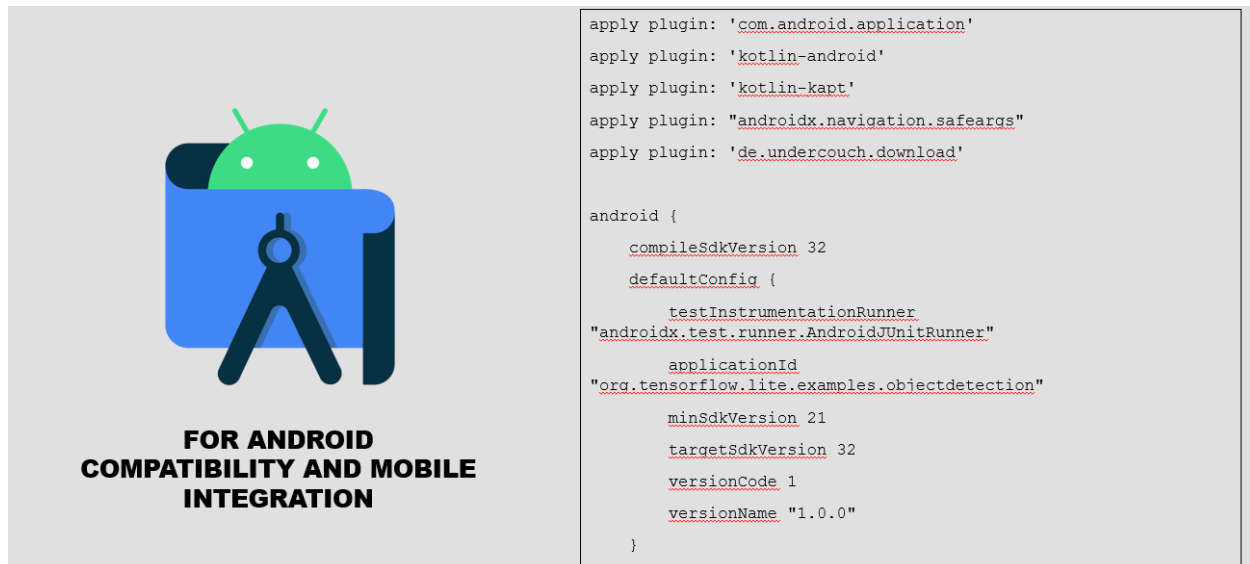


Figure 15: Android build.gradle specifying the minimum android version support

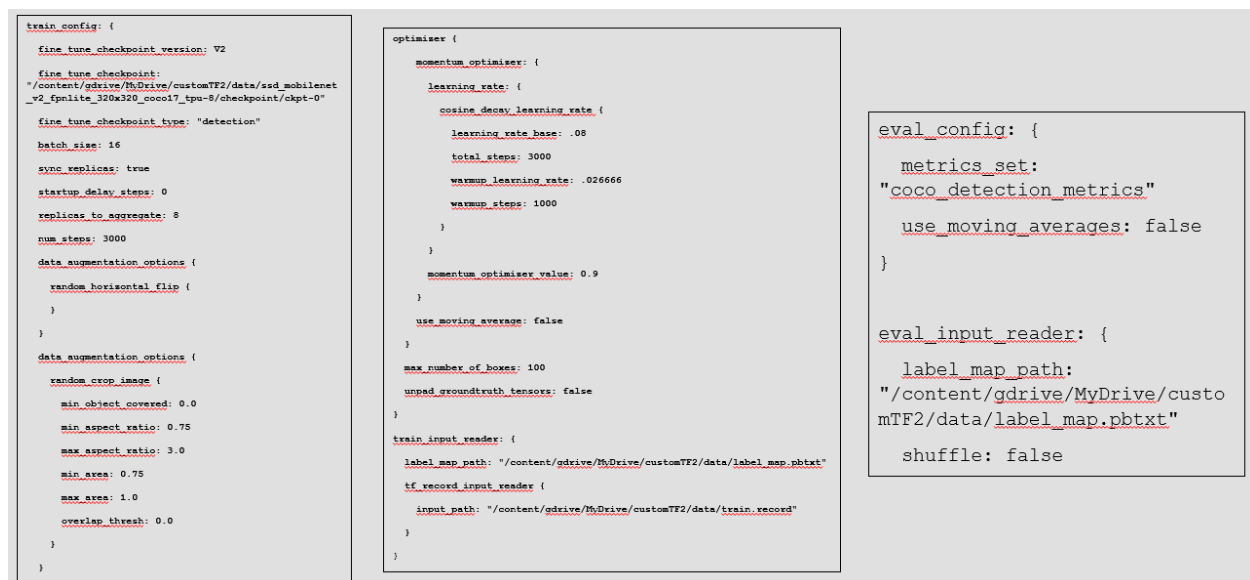


Figure 16: Config file specifying the training steps, classes, and the model used in training

1	filename	width	height	class	xmin	ymin	xmax	ymax
2	Goodbye_	2560	1920	Goodbye	1541	820	1764	1276
3	Goodbye_	2560	1920	Goodbye	1621	923	2110	1885
4	Goodbye_	2560	1920	Goodbye	1702	884	2048	1541
5	Goodbye_	2560	1920	Goodbye	1068	871	1364	1347
6	Goodbye_	2560	1920	Goodbye	546	169	1015	713
7	Goodbye_	2560	1920	Goodbye	1167	140	1533	681
8	Goodbye_	2560	1920	Goodbye	791	273	1195	848
9	Goodbye_	2560	1920	Goodbye	1780	717	2101	1323
10	Goodbye_	2560	1920	Goodbye	1837	1115	2136	1576
11	Goodbye_	2560	1920	Goodbye	1233	665	1490	973
12	Goodbye_	2560	1920	Goodbye	1529	728	1678	1086
13	Goodbye_	2560	1920	Goodbye	1651	709	1921	1215
14	Goodbye_	2560	1920	Goodbye	1949	803	2224	1250
15	Goodbye_	2560	1920	Goodbye	1893	672	2133	976
16	Goodbye_	2560	1920	Goodbye	1722	776	1921	1127
17	Goodbye_	2560	1920	Goodbye	1641	946	1907	1409
18	Goodbye_	2560	1920	Goodbye	1816	763	2207	1511
19	Goodbye_	2560	1920	Goodbye	1486	377	1862	956
20	Goodbye_	2560	1920	Goodbye	1643	697	1882	1161
21	Goodbye_	2560	1920	Goodbye	1577	745	1782	1171
22	Goodbye_	2560	1920	Goodbye	2032	748	2470	1576
23	Goodbye_	2560	1920	Goodbye	1446	685	1685	1164
24	Goodbye_	2560	1920	Goodbye	1820	548	2265	1363
25	Goodbye_	2560	1920	Goodbye	1962	798	2257	1228
26	Goodbye_	2560	1920	Goodbye	1930	531	2142	927
27	Goodbye_	2560	1920	Goodbye	1767	951	2061	1404
28	Goodbye_	2560	1920	Goodbye	1133	697	1379	1134
29	Goodbye_	2560	1920	Goodbye	1822	732	2079	1250
30	Goodbye_	2560	1920	Goodbye	1678	414	1916	829

Figure 17: .csv file of image annotations from dataset

Functional Testing

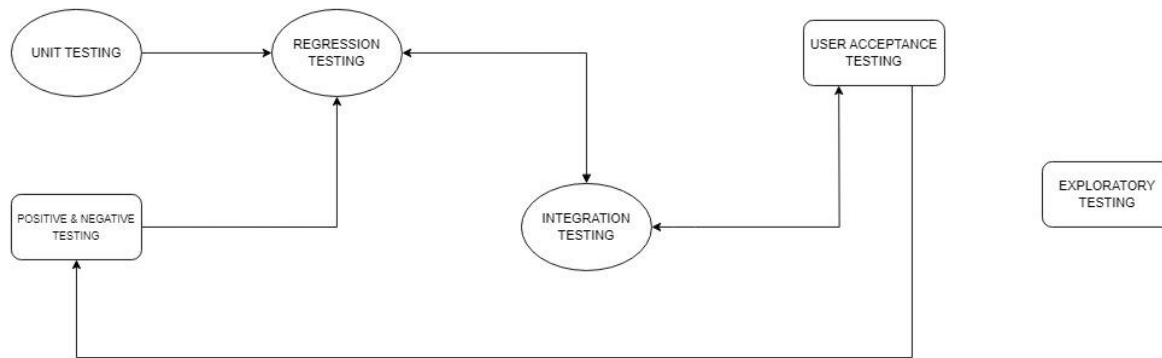


Figure 18: Functional Testing Flow

The testing process for Project Helping Hands involves various approaches to ensure the software's quality and functionality. Unit testing aims to test individual components in isolation to ensure their correct functioning. These tests focus on small, independent parts of the code to verify their expected behavior. Regression testing, on the other hand, aims to ensure that previously developed and tested functionalities continue to work correctly after implementing new changes or updates.

Positive and negative testing evaluates the system's behavior under different conditions. Positive testing the behavior when valid and correct inputs are provided. Whilst in negative testing, it evaluates how the system handles unexpected or invalid inputs.

In integration testing, researchers try to verify the proper interaction between different components, data input, or systems within the translator. User acceptance testing (UAT) ensures that Project Helping Hands meet the needs and requirements of its intended users. UAT involves gathering feedback from end-users in order to validate the system's suitability for real-world use.

Exploratory testing from design to execution, testers will explore the system's functionalities, potential issues, and defects without predefined scripts or formal test cases. They must rely on their knowledge, intuition, and creativity to gain a deeper understanding of Project Helping Hands.

Test Results

Date and Time	Key Test Issues	Test Activities	How you Corrected the Error
11-26-2022	Jupyter Notebook: Incompatible API Dependency Issues	Program Running	Uninstalling the incompatible API and reinstalling older versions.
11-27-2022	Jupyter Notebook: Increasing number of labels for detection	Training Model	Increased the size of the data set, with a more varied data set, then retrained model to test if detection rate improved.
12-1-2022	Jupyter Notebook: <ul style="list-style-type: none"> • Low Accuracy • Low Detection rate (consistency) precision • False Positive Detection 	Detect a specific gesture from the list of labels	
12-19-2022	Jupyter Notebook: Unsuitable Lighting Condition	Program Running	Change to brighter location when trying out the prototype not system issue but rather environmental/external.
12-22-2022	Jupyter Notebook: Facial Recognition affects output	Program Running	Removal of facial recognition for more focus on hand gesture/recognition.
02-10-2023	Google Colab: Poor performance of trained models. A lot of misclassifications and false positives	Training Model	Retrained the model. Corrected the wrong annotations on some images,
02-11-2023	Labeling image format	Annotating the images	Used proper format in annotating. Files should be in jpg form

03-12-2023	Android Studio <ul style="list-style-type: none"> The bounding box extends beyond the screen of the phone, but unlike a web application the bounding box is fixed. 	Program Running	The image is flipped when using the front camera of a mobile phone, so researchers adjusted the box coordinates accordingly.
------------	--	-----------------	--

TEST CASES

Test Case Identifier TC-1			
Use Case Tested UC-00, UC-01, UC-02			
Pass / Fail Criteria The test passes if the user can easily navigate to the Detections Screen and the Classes Screen starting from opening the mobile application.			
Input Data: User touch input			
Test Procedure:	Expected Result:	Actual Result	Pass/Fail
Step 1. Launch the application from the phone menu	The application will open and become visible on the phone's screen.		
Step 2. Through the UI, tap the button "View Detectable Gestures" ClassesScreen Activity to see how to do the gestures	The System starts the ClassesScreen activity and displays the list of valid gesture classes as affirmations.		
Step 3. Use the search function in the Classes Activity to find a specific gesture	The system gives the gesture with a matching name to the text query of the user.		

Test Case Identifier TC-2			
Use Case Tested UC-VOID, UC-NULL			
Pass / Fail Criteria The test passes if the system does not make any detections			
Input Data: (Live video capture of user)			
Test Procedure:	Expected Result:	Actual Result	Pass/Fail
Step 1. "Create"/"Display" a valid hand sign gesture based from the list detectable hand gestures	No detections are made by the system.		

Test Case Identifier TC-3			
Use Case Tested UC-AA, UC-AB, UC-AC, UC-AD, UC-AE			
Pass / Fail Criteria The test passes when the user displays a valid hand gesture depicting the letters "A", "B", "C", "D", and "E" using their hand, and is correctly recognized by the system.			
Input Data: (Live video capture of user's hand)			
Test Procedure:	Expected Result:	Actual Result	Pass/Fail
Step 1. "Create"/"Display" a valid hand sign gesture based from the list detectable hand gestures	System Software shows a bounding box around the hand; Besides this, a text showing the correct class "A", "B", "C", "D", and "E", respectively, with the percentage of how confident the system is with the detection.		

Test Case Identifier TC-4			
Use Case Tested UC-N1, UC-N2, UC-N3, UC-N4, UC-N5			
Pass / Fail Criteria The test passes when the user displays a valid hand gesture depicting the numbers "1", "2", "3", "4", and "5" using their hand, and is correctly recognized by the system.			
Input Data: (Live video capture of user's hand)			
Step 1. "Create"/"Display" a valid hand sign gesture based from the list detectable hand gestures	System Software shows a bounding box around the hand; Besides this, a text showing the correct class "1", "2", "3", "4", and "5", respectively, with the percentage of how confident the system is with the detection.	Actual Result	Pass/Fail

Test Case Identifier TC-5			
Use Case Tested UC-PTY, UC-PGB, UC-PIL, UC-PHE, UC-PYE, UC-PNO			
Pass / Fail Criteria The test passes when the user displays a valid hand gesture depicting the phrases "Thank You", "Goodbye", "I Love You", "Hello", "Yes" and "No" using their hand, and is correctly recognized by the system.			
Input Data: (Live video capture of user's hand)			
Test Procedure:	Expected Result:	Actual Result	Pass/Fail
Step 1. "Create"/"Display" a valid hand sign gesture based from the list detectable hand gestures	System Software shows a bounding box around the hand; Besides this, a text showing the correct respective class, with the percentage of how confident the system is with the detection.		

Test Case Identifier		TC-6	
Use Case Tested		UC-RR, UC-RP, UC-RS	
Pass / Fail Criteria		The test passes when the user displays a valid hand gesture depicting the hand gestures for "Rock", "Paper", "Scissors" using their hand, and is correctly recognized by the system.	
Input Data:		(Live video capture of user's hand)	
Test Procedure:	Expected Result:	Actual Result	Pass/Fail
Step 1. "Create"/"Display" a valid hand sign gesture based from the list detectable hand gestures	System Software shows a bounding box around the hand; Besides this, a text showing the correct respective class, with the percentage of how confident the system is with the detection.		

Test Case Identifier		TC-7	
Use Case Tested		UC-AA, UC-AS, UC-AT, UC-AM, UC-AE, UC-AN	
Pass / Fail Criteria		The test passes when the user displays a valid hand gesture depicting the letters "A", "S", "T", "M", "E" and "N" using their hand, and is correctly recognized by the system.	
Input Data:		(Live video capture of user's hand)	
Test Procedure:	Expected Result:	Actual Result	Pass/Fail
Step 1. "Create"/"Display" a valid hand sign gesture based from the list detectable hand gestures	The System Software displays a bounding box around the hand. Additionally, it shows a text indicating the correct class (A, S, T, M, N, E) along with the system's confidence percentage in detecting and distinguishing between these six visually similar letters captured by the camera.		

Test Case Identifier		TC-8	
Use Case Tested		UC-AD, UC-AY, UC-AI, UC-AJ, UC-AZ	
Pass / Fail Criteria		The test passes when the user displays a valid hand gesture depicting the letters "D", "Y", "I", "J" and "Z" using their hand, and is correctly recognized by the system.	
Input Data:		(Live video capture of user's hand)	
Test Procedure:	Expected Result:	Actual Result	Pass/Fail
Step 1. "Create"/"Display" a valid hand sign gesture based from the list detectable hand gestures	The System Software displays a bounding box around the hand. Additionally, it shows a text indicating the correct class (D,Y,I,J,Z) along with the system's confidence percentage in detecting and distinguishing between these six visually similar letters captured by the camera.		

Test Case Identifier		TC-9	
Use Case Tested		UC-AG, UC-AH	
Pass / Fail Criteria		The test passes when the user displays a valid hand gesture depicting the letters "G" and "H" using their hand, and is correctly recognized by the system.	
Input Data:		(Live video capture of user's hand)	
Test Procedure:	Expected Result:	Actual Result	Pass/Fail
Step 1. "Create"/"Display" a valid hand sign gesture based from the list detectable hand gestures	The System Software displays a bounding box around the hand. Additionally, it shows a text indicating the correct class (G,H) along with the system's confidence percentage in detecting and distinguishing between these six visually similar letters captured by the camera.		

Test Case Identifier TC-10			
Use Case Tested UC-AV, UC-AR, UC-AU			
Pass / Fail Criteria The test passes when the user displays a valid hand gesture depicting the letters "G" and "H" using their hand, and is correctly recognized by the system.			
Input Data: (Live video capture of user's hand)			
Test Procedure:	Expected Result:	Actual Result	Pass/Fail
Step 1. "Create"/"Display" a valid hand sign gesture based from the list detectable hand gestures	The System Software displays a bounding box around the hand. Additionally, it shows a text indicating the correct class (V,R,U) along with the system's confidence percentage in detecting and distinguishing between these six visually similar letters captured by the camera.		

Test Case Identifier		TC-11	
Use Case Tested		UC-L1, UC-L2, UC-L3	
Pass / Fail Criteria		The test passes when the user displays a valid hand gesture using their hand, and is recognized by the system even in varying light levels.	
Input Data:		(Live video capture of user's hand)	
Test Procedure:	Expected Result:	Actual Result	Pass/Fail
Step 1. "Create"/"Display" any valid hand sign gesture based on the list of detectable hand gestures.	The System Software recognizes the valid hand gesture and draws the corresponding bounding box, including the class name and confidence rate.		

Test Case Identifier		TC-12	
Use Case Tested		UC-B1, UC-B0	
Pass / Fail Criteria		The test passes when the user displays a valid hand gesture using their hand, and is recognized by the system even in a noisy background.	
Input Data:		(Live video capture of user's hand)	
Test Procedure:	Expected Result:	Actual Result	Pass/Fail
Step 1. "Create"/"Display" a valid hand sign gesture based from the list detectable hand gestures	The System Software recognizes the valid hand gesture and draws the corresponding bounding box, including the class name and confidence rate.		

Test Case Identifier		TC-	
Use Case Tested			
Pass / Fail Criteria			
Input Data:		(Live video capture of user's hand)	
Test Procedure:		Expected Result:	

Test Case Identifier		TC-	
Use Case Tested			
Pass / Fail Criteria			
Input Data:		(Live video capture of user's hand)	
Test Procedure:		Expected Result:	

--	--

Model Evaluations

The trained model was evaluated using a confusion matrix to determine the accuracy and precision when tested against the allocated testing set.

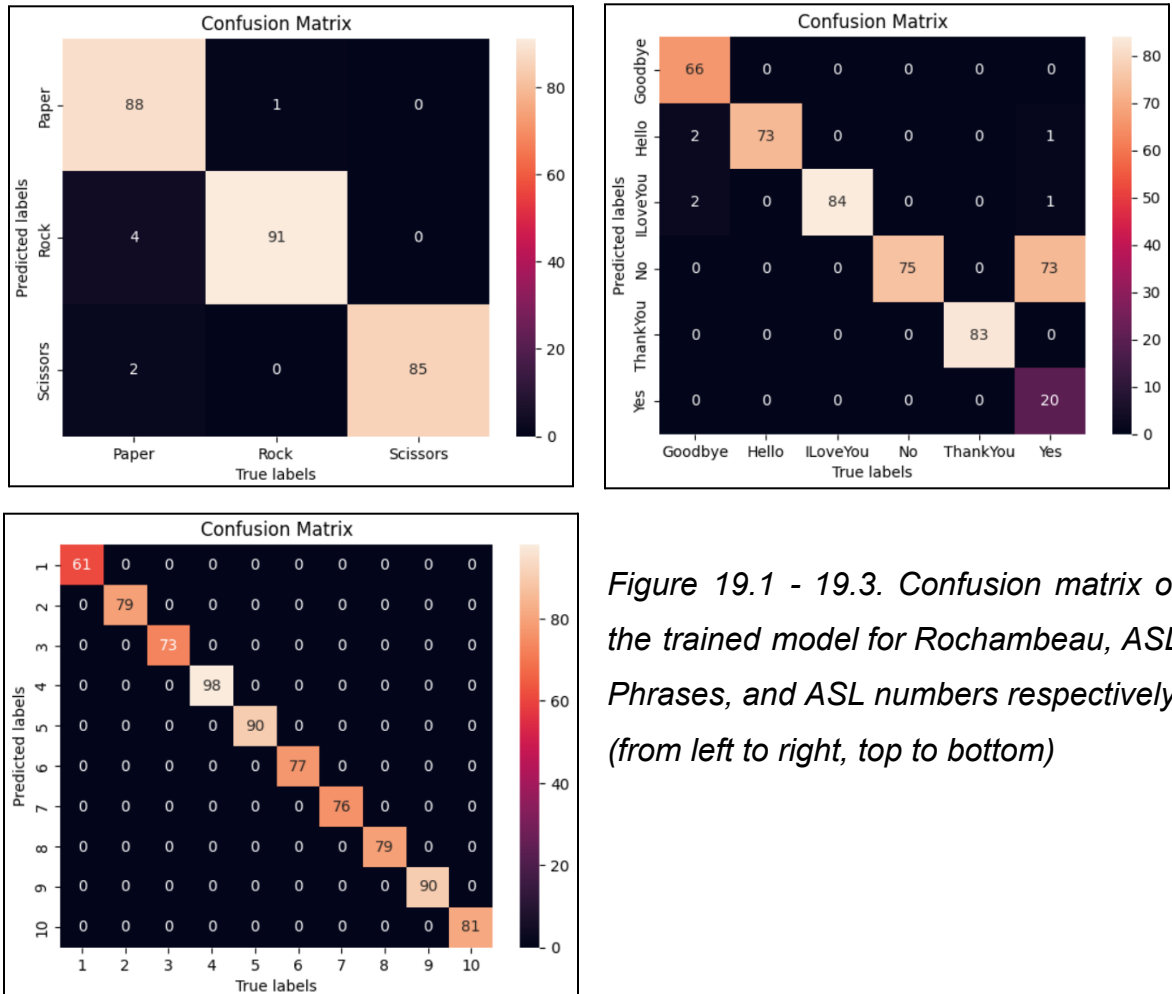


Figure 19.1 - 19.3. Confusion matrix of the trained model for Rochambeau, ASL Phrases, and ASL numbers respectively. (from left to right, top to bottom)

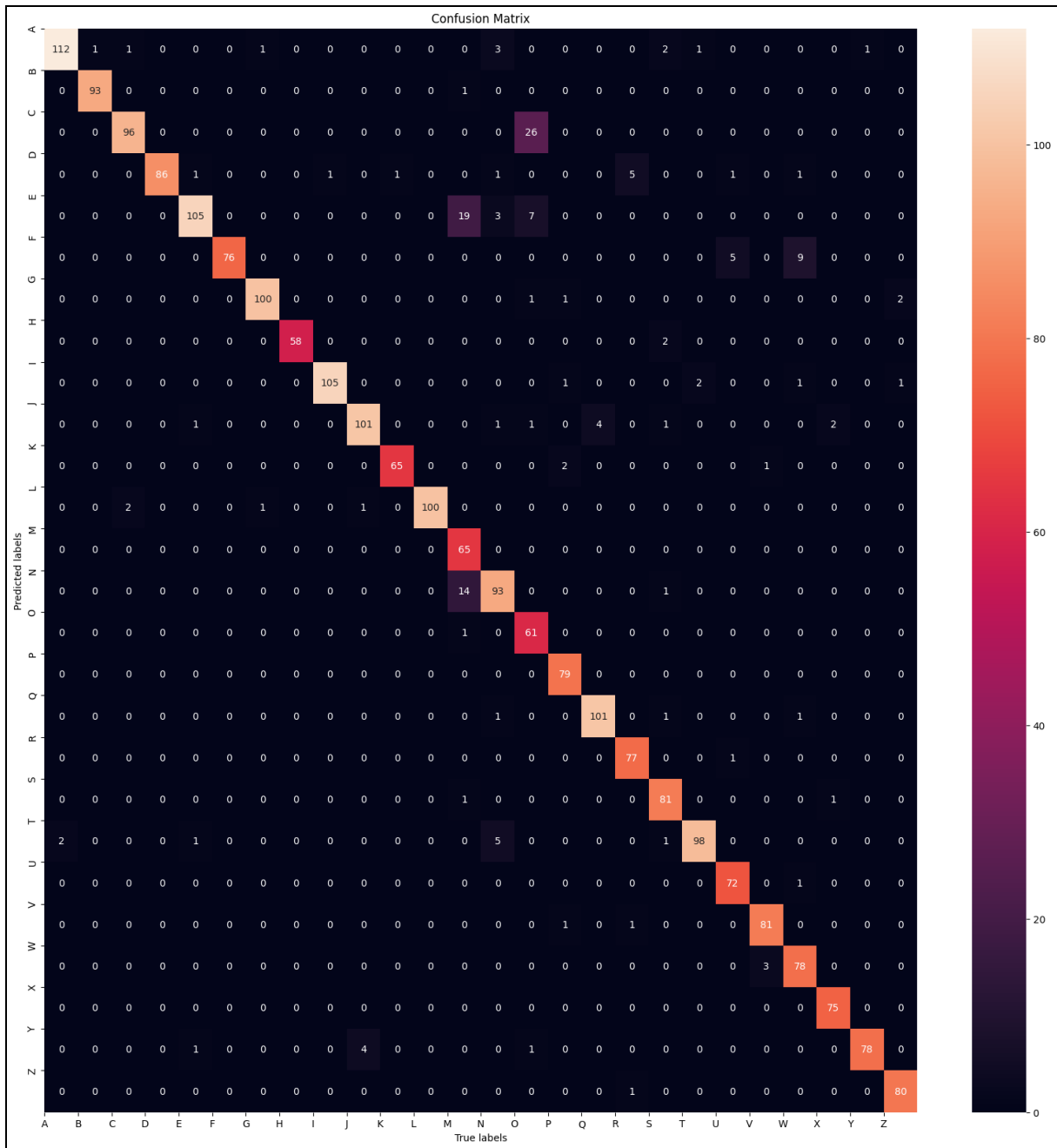


Figure 19.4 Confusion matrix of the ASL Alphabet model

A confusion matrix was created by selecting the detection with the highest confidence rate then comparing it to the image's actual label annotation, while disregarding the bounding box. This approach treats the task as a simple image classification rather than object detection, allowing the researcher to calculate the

model's accuracy and precision on the allocated testing set which is 20% of the data. True Positives are the correct detections. False positives include detections with the wrong class. False Negatives is when there is an image of a gesture, but no detection. True Negatives is not relevant in object detection because this entails the background that should have no detections.

TP	True Positive	A “positive” prediction that matches that of the actual class label.
FP	False Positive	Samples that are incorrectly predicted as "positive" by the model when they actually belong in another class.
FN	False Negative	Samples incorrectly predicted as a different class when they actually belong to “positive”.

In a multi-class classification problem, where there are more than two classes, the concept of true negatives (TN) becomes less relevant. The true negative (TN) value represents the count of samples that are correctly predicted as negative for a particular class when they actually belong to other classes. In a multi-class scenario, each class is considered independently, and the focus is on correctly identifying samples for a specific class rather than identifying samples that do not belong to that class.

$$P = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}}$$

$$R = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground-truths}}$$

F1-score: It combines precision and recall into a single measure. Mathematically it's the harmonic mean of precision and recall. It can be calculated as follows:

$$F_1\text{-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN}$$

F1- score combines precision and recall into a single measure. It is the harmonic mean of precision and recall.

Accuracy: Gives the overall accuracy of the model, meaning the fraction of the total samples that were correctly classified by the classifier. To calculate accuracy, use the following formula: $(TP+TN)/(TP+TN+FP+FN)$.

Rochambeau Model			
Classes:	Rock	Paper	Scissors
Training Count	94	92	85
TP count	91	88	85
FN count	1	6	0
FP count	4	1	2
Class Recall	98.91%	93.62%	1%
Class Precision	95.79%	98.88%	97.70%
Class Accuracy	94.79%	92.63%	97.70%
Total Precision	97.42 %		
Total Accuracy	95.04 %		

ASL Phrases Model						
Classes:	Yes	Thank You	No	I Love You	Hello	Goodbye
Training Count	93	83	75	84	73	70
TP count	20	83	75	84	73	66
FN count	75	0	0	0	0	4
FP count	0	0	73	3	3	0
Class Recall	26.67 %	100 %	100 %	100 %	100 %	94.29 %
Class Precision	100 %	100 %	50.68 %	96.55 %	96.05 %	100 %
Class Accuracy	21.05 %	100%	50.67 %	96.55 %	96.05 %	94.28 %
Total Precision	83.54 %					
Total Accuracy	76.44 %					

ASL Number Model										
Classes:	1	2	3	4	5	6	7	8	9	10
Training Count	61	79	73	98	90	77	76	79	90	81
TP count	61	79	73	98	90	77	76	79	90	81
FN count	0	0	0	0	0	0	0	0	0	0
FP count	0	0	0	0	0	0	0	0	0	0
Precision & Accuracy	100 %									

ASL Alphabet Model										
Classes:	A	B	C	D	E	F	G	H	I	J
Training Count	114	94	99	86	109	76	102	58	106	106
TP count	112	93	96	86	105	76	100	58	105	101
FN count	2	1	3	0	4	0	2	0	1	5
FP count	10	1	26	11	29	14	4	2	4	9

Recall	0.9824	0.9894	0.9697	1.00	0.9633	1.00	0.9803	1.00	0.9905	0.9528
Precision	0.9180	0.9894	0.7869	0.8866	0.7778	0.8444	0.9615	0.9667	0.9633	0.9909

ASL Alphabet Model										
Classes:	K	L	M	N	O	P	Q	R	S	T
Training Count	66	100	101	106	97	84	105	84	89	101
TP count	65	100	65	93	61	79	101	77	81	98
FN count	1	0	36	13	36	5	4	7	8	3
FP count	3	4	0	1	1	0	3	1	2	9
Recall	0.9848	1.00	0.6435	0.8773	0.6288	0.9404	0.9619	0.9166	0.9101	0.9702
Precision	0.9558	0.9615	1.00	0.9893	0.9838	1.00	0.9711	0.9871	0.9759	0.9158

ASL Alphabet Model						
Classes:	U	V	W	X	Y	Z
Training Count	79	85	91	78	79	83
TP count	72	81	78	75	78	80
FN count	7	4	13	3	1	3
FP count	1	2	3	0	6	1
Recall	0.9114	0.9529	0.8571	0.9616	0.9873	0.9638
Precision	0.9863	0.9759	0.9630	1.00	0.9286	0.9877
Total Precision	0.9448					

MODEL	ROCHAMBEAU	PHRASES	NUMBERS	ALPHABETS
TP Count	264	401	804	2130
Training Count	271	478	804	89.57
Precision	97.42%	83.89%	100%	89.57%

Precision and recall can be calculated using the formula above. Precision is the degree of exactness of the model in identifying only relevant objects. It is the ratio of TP's over all detections to that class, made by the model. Meanwhile, recall measures the ability of the model to detect all ground truths— proportion of TP's among all ground truths. From our confusion matrix, the precision is 87%, while recall is at 100% because there were no false negatives, which means all the images had at least 1 detection even if incorrect.

The researcher also has to evaluate if an object was correctly located. The metric used for this task is called the Intersection over Union (IoU) as a similarity measure. It's given by the area of the overlap divided by the size of the union of the two bounding boxes.

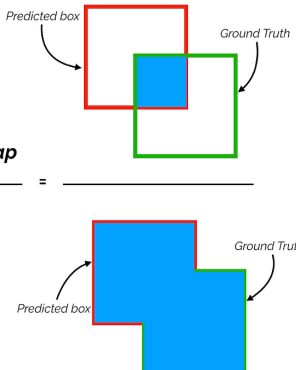
$$\text{Intersection over Union (IoU)} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Figure 20: Formula for Intersection over Union (IoU)

For object detection tasks, the Precision and Recall is calculated using IoU value for a given IoU threshold. For example, if the IoU threshold is 0.5, and the IoU value for a prediction is 0.7, then researchers classify the prediction as True Positive (TF). On the other hand, if IoU is 0.3, researchers classify it as False Positive (FP). That also means that for a prediction, researchers may get different binary True or binary False positives, by changing the IoU threshold.

The generally accepted metric for object detection model evaluation is by calculating the Mean Average Precision (MaP). It is the most popular metric that is used by benchmark challenges such as PASCAL VOC, COCO, ImageNET challenge, Google

Open Image Challenge, etc. The mean Average Precision or mAP score is calculated by taking the mean AP over all classes and/or overall IoU thresholds, depending on different detection challenges that exist. To calculate the Average Precision (AP), we first sort the probability scores and then check if the bounding box is a True Positive or a False Positive using an IoU threshold to compare them. If there is more than one detection for a single object, the detection having the highest IoU is considered as TP, rest as FP.

Average Precision and Recall across different values of IoU:

```
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=0.98s).
Accumulating evaluation results...
DONE (t=0.28s).
Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.801
Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.992
Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.971
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.735
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.802
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.836
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.838
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.839
Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.741
Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.840
INFO:tensorflow:Eval metrics at step 5000
I0513 15:38:37.104531 139947550893888 model_lib_v2.py:1015] Eval metrics at step 5000
```

Figure 21.1: Training Metrics for Rochambeau model

```
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=1.01s).
Accumulating evaluation results...
DONE (t=0.39s).
Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.715
Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.875
Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.848
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.715
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.825
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.843
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.843
Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.843
INFO:tensorflow:Eval metrics at step 5000
I0330 17:57:19.079013 140470708561728 model_lib_v2.py:1015] Eval metrics at step 5000
```

Figure 21.2: Training Metrics for ASL Phrases model.

```

Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=2.18s).
Accumulating evaluation results...
DONE (t=0.81s).
Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.906
Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.998
Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.996
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.906
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.929
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.930
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.930
Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.930
INFO:tensorflow:Eval metrics at step 5000
I0426 07:23:07.436141 140003595409216 model_lib v2.py:1015] Eval metrics at step 5000

```

Figure 21.3: Training Metrics for ASL Numbers model.

```

Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=12.72s).
Accumulating evaluation results...
DONE (t=6.54s).
Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.747
Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.973
Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.922
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.539
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.747
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.804
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.809
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.809
Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.625
Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.809
INFO:tensorflow:Eval metrics at step 10000
I0424 09:05:50.342109 140534149404480 model_lib v2.py:1015] Eval metrics at step 10000

```

Figure 21.4: Training Metrics for ASL Alphabet model.

The evaluation metrics of COCO format are a set of measurements used to evaluate the performance of a model. The measurements include how well the model can identify objects of different sizes, how precise it is in its detections, and how well it can recall certain objects. It is displayed as a dictionary with different keys that represent different measurements.

Precision/mAP stands for the mean average precision over classes, averaged over IoU thresholds ranging from 0.5 to 0.95 with .05 increments. The small area values stand for the size of the detected objects (32^2 pixels). Medium area is for detected objects sized between 32^2 and 96^2 pixels. Large objects are those larger than 96^2 pixels. The maxDets variable stands for the maximum detections.

The Average Precision is at 88% when IoU threshold is at .50, however researchers only got 73% precision across .50 to .95 IoU threshold values. The value of the average precision and recall when the area is medium or large is -1 because the category is absent in both our training and testing data set. This means no bounding box less than 96^2 pixels was present, so only “large” areas are available.

The current model was trained for 5,000 steps which took 3 hours to finish. Researchers can still train for longer steps, or increase our training dataset to try to improve the model performance. However, the learning rate will be decreasing the longer researchers train until a point of diminishing returns.

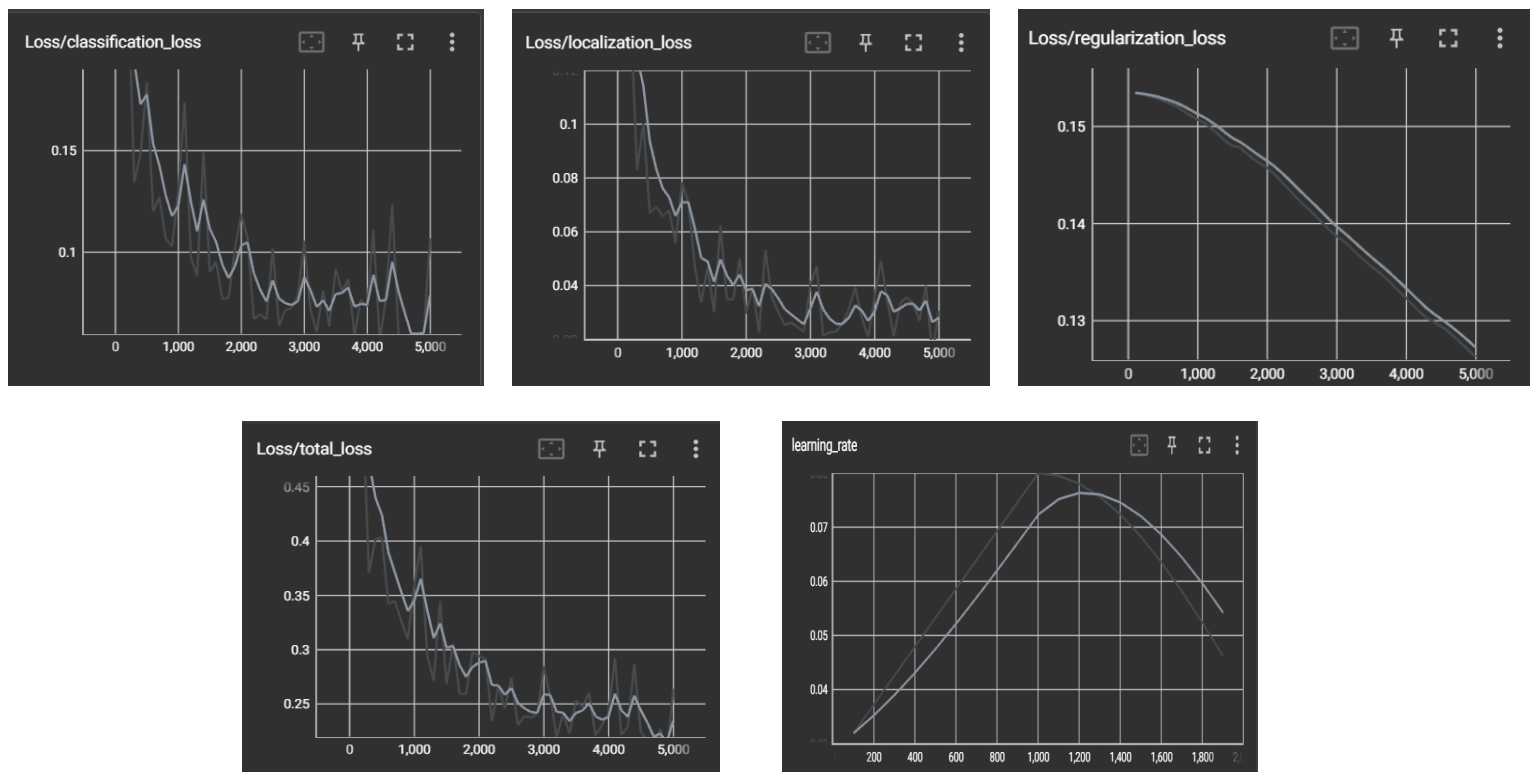


Figure 22: Classification loss, Localization loss, Regularization loss, Total loss and Learning Rate of ASL Gestures

The figure displays a graphical representation of two performances of a given model: the loss and the learning rate. The total loss refers to the amount of error or deviation that the model experiences when making predictions on a given dataset. The classification loss means the score of the correct category should be greater than the sum of scores of all incorrect categories by some safety margin. The regularization loss is the loss function on the output of CNN. The localization loss is the mismatch between the ground truth box and the predicted boundary box. The learning rate refers to the speed at which the model updates its parameters during training. This value is important because it can affect both the speed and the quality of the model's convergence during the training process.

Software Re-design & Re-construction

Suggestions/Recommendations	Test Activities
Add more datasets	Planning and managing
Mention the classes (signs) used for translation and include it in the scope	Development or Coding
Add parameters for the evaluation	Development or Coding
Include evaluation in the non-functional requirement	Planning and managing
Include in the Methodology the embedding of the training and testing part of the code	Planning and managing
Check “model library” part in the RCV	Planning and managing
Check whether to define separately the hardware specification part	Planning and managing
Replace ”assessment” term with other terminology	Planning and managing
Add in Limitations the background limiting factor for detection rate	Designing
Improve datasets	Designing
Embed a code for testing accuracy and precision for different updated of hand	Development or Coding
Should not only focus on accuracy, should also have precision	Development or Coding
Add guide (hand sign guide)	Designing
Add more classes	Planning and managing
Resize the label for the bounding box when detecting live (make it bigger for better readability)	Designing

USER'S MANUAL

Accessing the Software

1. **NOTE:** Before running the software, check/read if the following conditions are met.
 - a. Suggested Operating System (OS) must be running on Android 7.0 (Nougat) as the minimum OS or above Android 7.0 for better usability.
 - b. The software only caters the following labels for translation:
 - i. Words/Phrases:
 1. Yes
 2. No
 3. Thank you
 4. I love you
 5. Hello
 6. Goodbye
 - ii. Numbers:
 1. 1,2,3,4,5,6,7,8,9
 - iii. Letters:
 1. A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
 - iv. Rochambeau
 1. Rock
 2. Paper
 3. Scissors
 - c. Camera should not be connected nor active in any other third-party applications not related to or belonging to software.
 - d. Optimal lighting is necessary for accurate translation/s.
2. If the above conditions have been met, run the software.
3. Once software is active, user will be prompted into the home menu where the user will be able to choose to start camera detection or view detectable hand signs before proceeding to detection;
 - a. When in camera detection screen: a camera display should be present. Automatic detection of hand signs or gestures with live interpretation for each movement should also be in view.

- b. When in view detectable hand signs screen: User will be able to view a list of detectable hand signs alongside a sample picture of the detectable hand signs.
- 4. If any errors arise, such as: lag; no camera display; hand detection are not corresponding; etc. Check if system requirements are met, else force close and restart the software.
- 5. The application only works at a minimum distance of half a meter, up to a maximum distance of one and a half meter.

Bibliography

Adam Beckerman Image Dataset. (n.d.). Retrieved from <https://universe.roboflow.com/robotichand/adambeckerman/dataset/1?fbclid=IwAR3XQ91clGmY56XHkZtnkNYZ2NrJsO6OYIRnhOD2fxz76679SmBvw5QoqUw>

Britannica. (n.d.). Image processing. Retrieved from <https://www.britannica.com/technology/image-processing>

Catura, S. (2018). "Data/Report on Persons with Disability (PWDs)". Retrieved from Google Drive: [https://drive.google.com/file/d/1-5-16-2018\(1\)-5-28-2018CSC.pdf](https://drive.google.com/file/d/1-5-16-2018(1)-5-28-2018CSC.pdf)

Department of Health. (n.d.). Persons With Disabilities. Retrieved from <https://doh.gov.ph/persons-with-disabilities>

Department of Social Welfare and Development. (n.d.). Persons with Disability. Retrieved from <https://www.dswd.gov.ph/persons-with-disability>

GeeksforGeeks. (n.d.). Digital Image Processing Basics. Retrieved from <https://www.geeksforgeeks.org/digital-image-processing-basics/>

How many are deaf in the Philippines? (n.d.). Retrieved from <https://shortquestion.com/how-many-are-deaf-in-the-philippines/>

Netron TFLite Model Visualizer. (n.d.). Retrieved from <https://github.com/lutzroeder/netron>

NCDA (National Council on Disability Affairs). (n.d.). "Disability Data". Retrieved from <https://www.ncda.gov.ph/disability-data/>

Philippine Statistics Authority. (n.d.). Persons with Disability. Retrieved from <https://psa.gov.ph/persons-with-disability>

Philippine Statistics Authority. (n.d.). 2020 Census of Population and Housing (2020 CPH) Population Counts Declared Official by the President. Retrieved from <https://psa.gov.ph/content/2020-census-population-and-housing-2020-cph-population-counts-declared-official-president>

RockPaperScissors-official Image Dataset. (n.d.). Retrieved from https://universe.roboflow.com/yolorockpaperscissors/rockpaperscissors-official/dataset/2?fbclid=IwAR0eHXIz6u9d8lO07MAwZsR1b8uhG7vIUyRHhDfg4hm-474_Pb0TMRhvKr8

Sign Computer Vision. (n.d.). Retrieved from <https://universe.roboflow.com/ktnf749-snu-ac-kr/sign-hb57g?fbclid=IwAR0IlwQAsZboqm oquB2Wx-Qq1ys7vSoM5kzNnGI8c8XEVqJpRIEkw7qxLmA>

Schools with PWD access. (2020, May 6). Retrieved from Google Sheets: https://docs.google.com/spreadsheets/d/Schools_with_PWD_access_as_of_May_6_2020.xlsx

TheAILearner. (n.d.). What is a Digital Image? Retrieved from <https://theailearner.com/digital-image/>

TensorFlow. (n.d.). SSD_Mobilenet. Retrieved from <https://tfhub.dev/t>

University of Tartu. (n.d.). Introduction to image processing. Retrieved from https://courses.cs.ut.ee/MTAT.03.183/2015_fall/uploads/Main/01_Introduction.pdf